Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering (CS 319)

Fall 2009

Billiard Project

Final Report

Group 8 Members

Enes TAYLAN

Hüseyin GÜLER

Alperen ERASLAN

Ömer DURMUŞ

Table of Contents

1. INTRODUCTION

We are assigned to design and implement an object-oriented programming project which is named Billiard (pool) game as we know as Bilardo. Billiard is a table game which is played with balls and a cue stick. Generally, aim is to put the ball into holes in 8-ball and touching 2 balls in 3-ball game. Rules of the game and conditions in game (such as table, balls) change depending on the game type.

There is a general definition of and brief information about Billiard in "Introduction" part of this report. After short description of the game, there is an explanation of why we have chosen to work on this game. In this report, requirements of the project are coming under two headings: Functional Requirements and Non-functional Requirements. System models of the game will be described in detail. System models section includes domain analysis, use-case analysis, sequence diagrams and state diagrams. After System models section, detailed description of the game features will be deliberated. Then the report is concluded with evaluation of Game Project Analysis Period.

2. PROBLEM STATEMENT

Billiard is a well-known entertaining arcade game. It is a real time game at the same time. Current versions of billiard game is consists of one type of game each. For instance, 8-ball and 3-ball billiard game cannot be played in same software, only one type of game is provided in games that have been done. Besides, player cannot adjust power as s/he wants in current versions of billiard.

Based on these current versions, a new billiard game should contain 3-ball and 8-ball type of games together. Moreover, there must be a more user-friendly system for adjusting the power to hit the ball.

Every system that Java Runtime Environment is installed on it should be able to run billiard game.

When the project finished, a complete billiard game with 2 game type including 8-ball and 3-ball will be delivered. Complete game will be completed by December,17 2009.

3. REQUIREMENTS ANALYSIS

3.1. Game Overview

There are hundreds of pool games. Some of the more well known include eight-ball, nine-ball, straight pool, and one-pocket. In this project we have decided to design and implement eight-ball game which is the most popular one in amateur field.

Eight-ball is played with sixteen balls: a cue ball, and fifteen object balls consisting of seven striped balls, seven solid balls and the black 8 ball. After the balls are

scattered on a break shot, the players are assigned either the group of solid balls or the stripes once a ball from a particular group is legally pocketed. The ultimate object of the game is to legally pocket the eight ball in a called pocket, which can only be done after all of the balls from a player's assigned group have been cleared from the table.

Players should be careful about not to pocket the black 8 ball before pocketing all the other balls because that will cause to game over.

3.2. Functional Requirements

3.2.1. Help

Help option should provide the necessary information about the game. Users should be able to find the rules of the game. The keys and their assigned functions in the game should be expressed in help clearly. The properties of the important parts of the game should be able to be accessed from here.

3.2.2. Game Type Selection

We have 2 game types in this project first is the 8-ball game and the other one is the 3-ball game. User should be able to determine the type of the game which he wants to play in order to start the game. Depending on the selection of the user; proper game screen should be displayed.

3.2.3. Game Mode Selection

Apart from game type, game mode selection should be able to be selected in order to start the game. There should be 2 types of game mode, one is Player vs. Computer and the other one is Player vs. Player. Second option will not exactly be like a multiplayer game since it will not be played through network. Users should be able to play on the same computer turn by turn.

3.2.4. Start Game

After selecting the game type and game mode, game screen should be displayed. Players should be able to play the game after game screen is displayed.

3.2.5. Pausing the Game and Continue Game

Players should be able to pause the game at any time. By this way, players should be able to do their other works and then, they should be able continue the game from the level that the game was interrupted.

Continue Game option is a basic requirement of the game since the game can be interrupted by the player at any time after the game took start. This feature should be able to be used if and only if there was a paused game beforehand.

Then the user should be able to continue to his/her game from where he was playing at.

3.2.6. Settings

Settings option is the part of the main menu where players should be able to change the settings of the game manually. For example, a user should be able to play the game without sounds, thus he/she should be able to change the sound settings accordingly. Also there should be some songs integrated in the game that should be able to be selected by the user to listen throughout the game. Gameplay settings (background picture, cue stick type, billiard chalk type, etc.) and sound effects of the game should be able to be set here.

3.2.7. Hall of Fame

Most computer games have a part that shows the highest points recorded at that game. Players should be able to look at their scores to compare themselves to other players. Highest points and the record holders should be able to be registered to the Hall of Fame window. By this way, the quality of the competition and the real rivalry may be achieved by many players.

3.3. Non-functional Requirements

3.3.1. Performance

Billiard Game is played by just one player (Computer vs. Player) or by two players (Player vs. Player). It is very important to give smooth user experience to players. To do so in both modes, the game should response quickly. In Computer vs. Player mode, the game should decide which action it should take in a little time. Also, in Player vs. Player mode transitions between player's turn should not take much time

Apart from playing, the game should also be able to transit from one screen to another screen (for example, from playing screen to settings screen).

3.3.2. Easy to Learn

The graphical user interface (GUI) is not too complicated therefore the players of the game will be enabled about getting familiar with the game view for new users. Experienced users in real Billard Game (not software) can play the game very easily, already.

3.4. Constraints

The implementation language of Billiard game will be Java.

3.5. Scenarios

S1.

Scenario Name: Play 8-Ball Billiard Game vs. Computer

Participating Actor Instances: Hüseyin: Player, Computer: ComputerPlayer

Flow of Events:

Hüseyin chooses the 8-ball game vs. Computer.
Hüseyin starts to game by hitting the white ball.
Balls are spread out and one smooth ball goes in a hole.
Hüseyin hits the white ball again and gets a point again.
Hüseyin wishes to change the color of cue stick into blue.
Hüseyin shots again, but it is not successful.
Computer hits the white ball and 2 non-smooth balls goes into hole.
Hüseyin wishes to exit and closes the game without saving.

S2.

Scenario Name: Play 3-Ball Billiard Game vs. Computer

Participating Actor Instances: Enes: Player, Computer: ComputerPlayer

Flow of Events:

Enes chooses the 3-ball game vs. Computer.
Enes starts to game by hitting the white ball.
White ball hits one other ball only. (Unsuccessful shot)
Computer hits the white ball and gets a point.
Computer hits the white ball again and gets another point.
Enes wishes to change the color of cue into red.
Computer hits the white ball and misses.
Enes gets 10 successful shot.
Enes wins the game.
Enes score saves into the Hall of Fame.
Game ended.

S3.

Scenario Name: Play 2 Player 8-Ball Billiard Game

Participating Actor Instances: Alperen: 1st Player, Mali: 2nd Player

Flow of Events:

Alperen chooses the 8-ball game vs. Mali.
Alperen starts to game by hitting the white ball.
Balls are spread out and one smooth ball goes in a hole.
Alperen hits the white ball again and gets a point again.
Alperen hits again but it is unsuccessful.
Mali shots and 2 non-smooth balls go into hole.
Alperen hits the again and 2 other non-smooth balls go into hole.
Alperen exits from the game without saving.

S4.
Scenario Name: Play 2 Player 3-Ball Billiard Game
Participating Actor Instances: Kemal: 1st Player, Buğra: 2nd Player
Flow of Events:

Kemal chooses the 3-ball game vs. Buğra.
Kemal starts to game by hitting the white ball.
White ball hits both of two other balls.
Kemal gets another point.
Kemal misses the next shot.
Buğra wishes to change the color of the table into blue.
Buğra hits the white ball and misses.
Kemal reaches 10 successful shot.
Kemal wins the game.
Kemal's score saves into the Hall of Fame.
Game ended.

3.6. Use Case Models

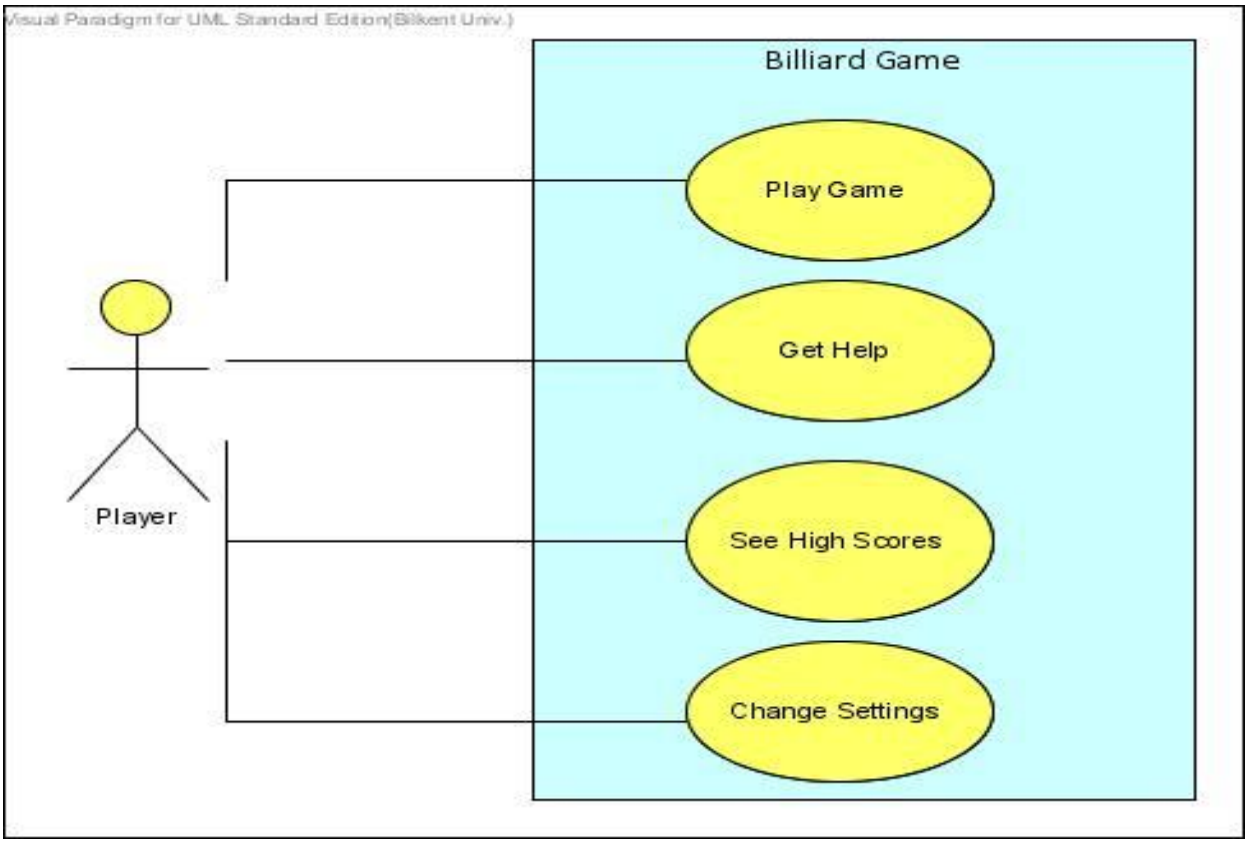


Figure 1: Use Case Diagram

Use case diagram of the Billiard game system is above. Our system basically consists of 4 use cases. Play game, change settings, get help and see high scores. Play

game use case is the one for the user to play the game. User is able to select game type (8-ball or 3-ball) and game mode (player vs. player or player vs. computer) before starting the game. Change settings is the option for the user to set some game options like game music, sound effects, table (background) picture, cue stick type, etc. Get help is for user to get some information about the usage of the system and how to play the game. Also some information about the billiard terms can be gained. See high scores use case is the one for the user to see his/her score in the Hall of Fame list. In the remaining of the report, these use cases will be explained further by analyzing them.

Use Case UC1: Play Game

Actors: Player: User who wants to play the game
Preconditions: Player has opened the game, and then s/he has selected to start a new game, and then selected the game type and game mode accordingly.
Flow of Events:

1. Player selects "New Game" option from the main menu.
2. Game type is selected.
   a. 3-ball game type is selected.
   b. 8-ball game type is selected.
3. Game Mode is selected.
   a. Player vs. Computer game mode is selected.
   b. Player vs. Player game mode is selected.
4. Game is started with the selected options.
5. Game is ended successfully.
6. The score s/he has made is taken and saved into the Hall of Fame.
   a. If Player vs. Computer mode is selected the score of the player is saved.
   b. If Player vs. Player mode is selected the score of the winning player is saved.

Exit Conditions:

- At any time, in the 8-ball game if the ball number 8 is pocketed, thus game ended.
- All the balls are in hole and one of the players won the game, thus game ended.
- Player exits the game.

Use Case UC2: See High Scores

Actors: Player: User who wants to see high scores.
Preconditions: Player has opened the game and the main menu is showed up.
Flow of Events:

1. Player selects "Hall of Fame" menu from the main menu.
2. In "Hall of Fame" menu, a list of highest scores is displayed.
3. Player observes the list and compares him/herself with top scores.

Exit Conditions:

- Player returns to main menu.
- Player exits the game.

Use Case UC3: Get Help

Actors: Player: User who wants to get help about the game.
Preconditions: Player has opened the game and main menu is showed up.
Flow of Events:

1. Player selects “Help” menu from the main menu.
2. In “Help” menu, controls and instructions of the game is introduced.
3. Player reads and comprehends the instructions and rules, s/he clicks the “Back to Main Menu” in order to return to main menu.

Exit Conditions:

- Player returns to main menu.
- Player exits the game.

Use Case UC4: Change Settings

Actors: Player
Preconditions: Player has opened the game and main menu is showed up.
Flow of Events:

1. Player enters to “Options” menu from the main menu.
2. A menu containing all options and settings for the game is showed up.
3. Player arranges the settings and options according to his/her choice.
4. Player wants to return to Main Menu and a confirmation is asked.
   4a. Player prefers cancelling.
      1. Player stays in Options menu.
   4b. Player prefers to save settings and return to main menu.
      1. All changes in settings are saved.
      2. Player is returned back to main menu.
   4c. Player prefers to go back to main menu without applying changes.
      1. Player is returned back to main menu.

Exit Conditions:

- Player returns to main menu.
- Player exits the game.

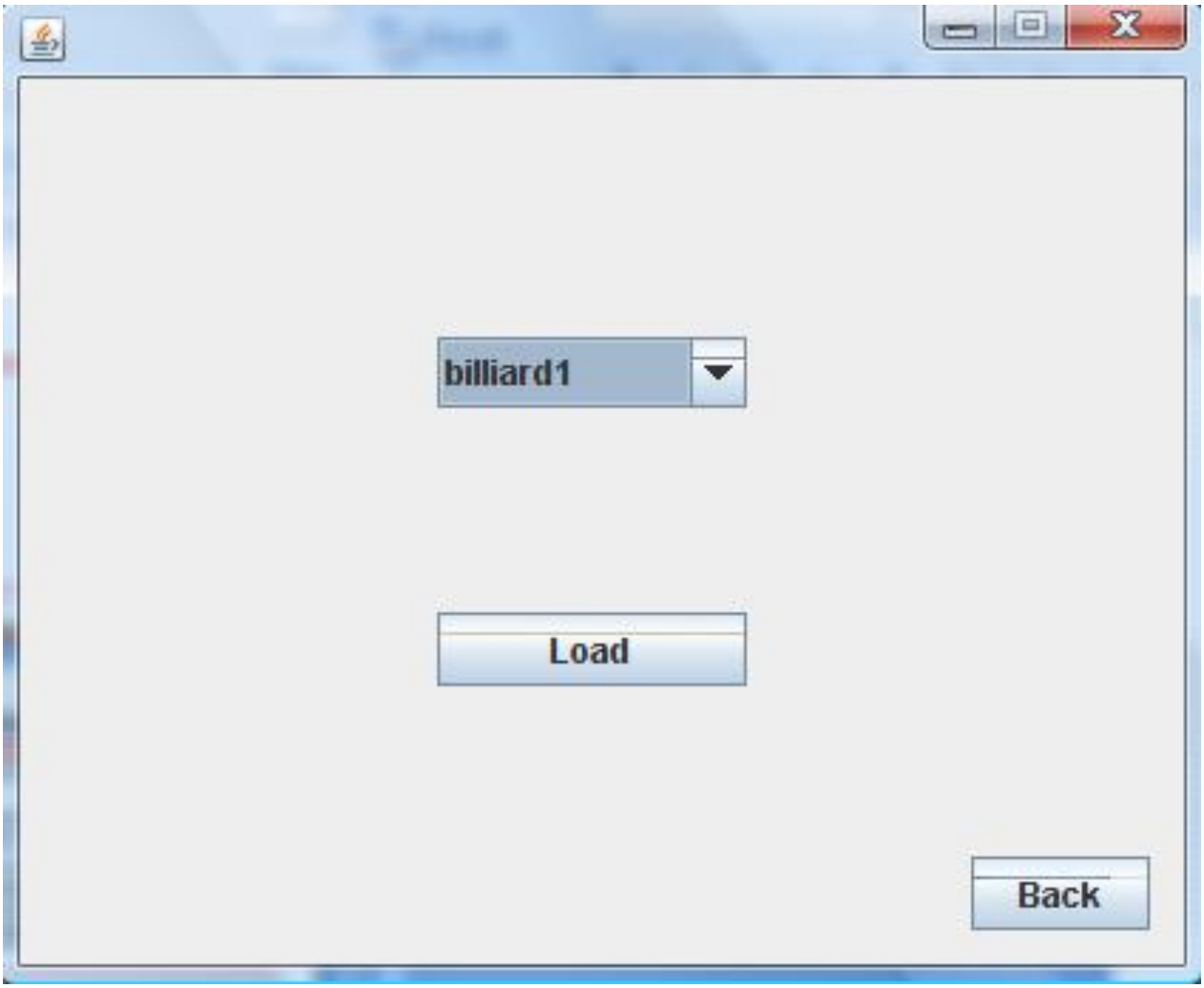
3.7. User Interface

1.Start Menu

Start menu is consisting of New Game, Load **G**ame, Preferences, Credits, Help and Exit buttons. Player can start to play game by clicking New Game button.

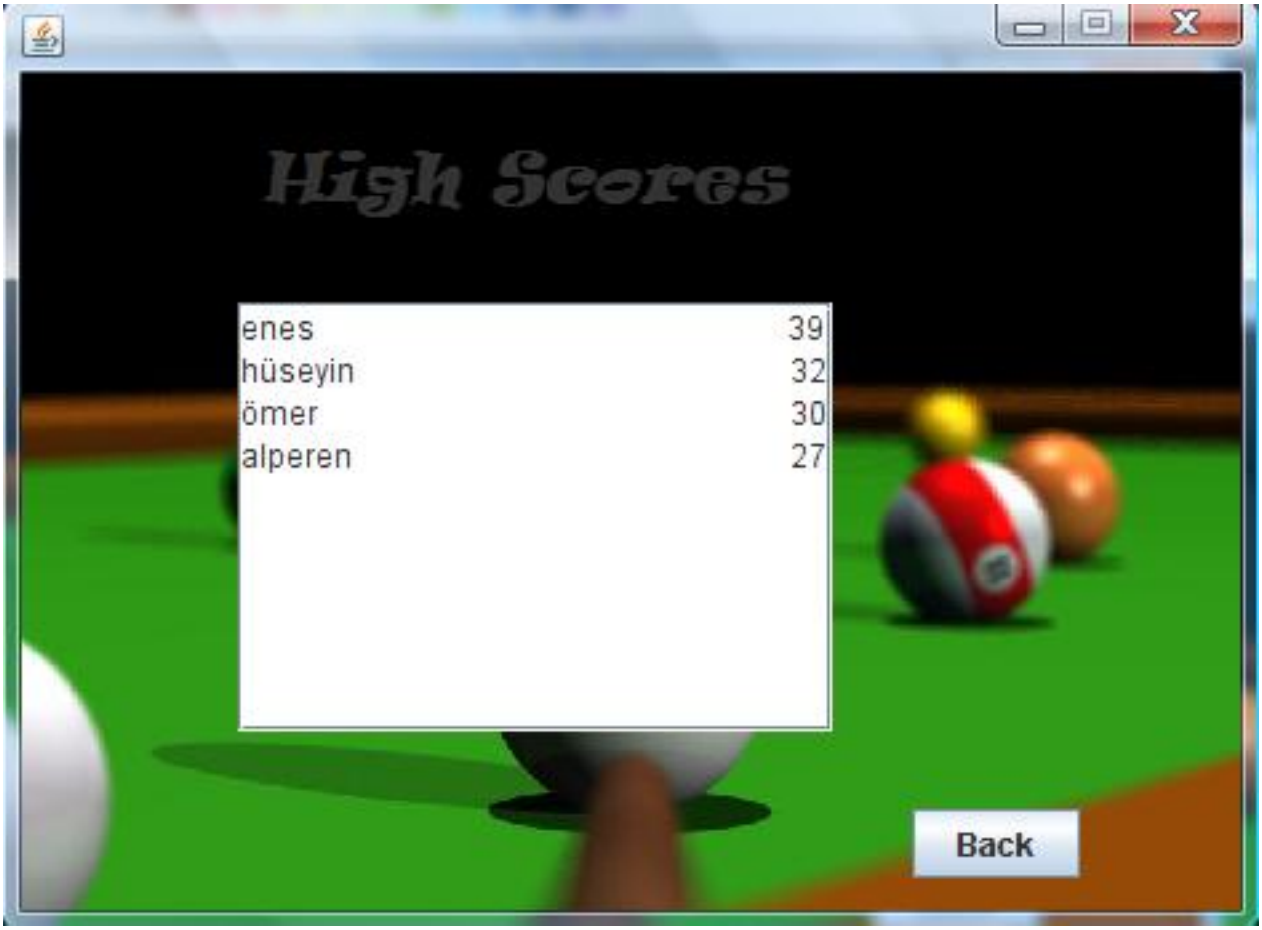
2. Load Menu

If player wishes to continue from last saved game, he/she is able to do it by clicking Load Game button. In the coming screen, player may start the game by clicking Load button or he/she can exit by clicking Exit button.



3. High Scores Menu

In High Scores Menu, player can see the highest scores. Player can exit anytime by clicking exit button.



4. Game Screen

It can be seen in this screen that, a billiard game is playing between two players. Player aims to push colored balls into holes by hitting white ball by the help of cue. Player can adjust the hit power of the cue and also can adjust the direction of the cue.

3.8. DETAILED INFORMATION ABOUT GAME

3.8.1. 8-Ball Game

RACK

To start the game, the object balls are placed in a triangular rack. The rack is positioned so the apex ball of the rack is located on the foot spot. The balls in the rack are ideally placed so that they are all in contact with one another. This is accomplished by pressing the balls together from the back of the rack toward the apex ball. The placement of the balls, for a legal rack according to World Standardized Rules is that the 8 ball is placed in the center, while the two lower corners must be a stripe and a solid and balls alongside of the triangle are alternated between stripes and solids (see image below). The cue ball is placed anywhere the breaker desires inside the "kitchen".

OBJECT OF THE GAME

In Eight ball, the object of the game is to pocket the Eight ball after having legally pocketed either all the Stripes (#9 to #15) or the Solids (#1 to #7). Once all the player's object balls are pocketed, he/she may attempt to sink the 8 ball. To win, the player must first call which pocket they plan to sink the Eight ball into. If the Eight ball is shot into the wrong pocket or a foul (see below) occurs, the player loses. Otherwise, the player's turn is over.

OPENING BREAK

One person is chosen (by a predetermined method, e.g., coin flip, win or loss of previous game, lag) to shoot first and break the object ball rack apart. If the shooter who breaks fails to make a legal break (usually defined as at least four balls hitting cushions or an object ball being pocketed), then the opponent can demand a re-rack and become the breaker, or elect to play from the current position of the balls.

If the breaker pockets a ball, it is still that player's turn and the table is considered "open" (meaning the breaker can still make any object ball to determine if he/she will only shoot solids or stripes throughout the game). If the breaker fails to make another ball after the break, the table is still considered "open" until someone legally pockets a ball.

According to World Standardized Rules, if the 8-ball is pocketed on the break,

breaker may ask for a re-rack or have the 8-ball spotted and continue shooting. If the breaker scratches while pocketing the 8-ball on the break, the incoming player has the option of a re-rack or having the 8-ball spotted and begin shooting with ball in hand behind the head string.

WINNING AN EIGHT BALL GAME

***** The player has legally pocketed the 8 ball, after all his/her object balls have been pocketed
***** The opposing player illegally pockets the 8 ball (e.g. before clearing all of his/her object balls, in the same shot as the last such object ball, or into a pocket other than the one that was called)
***** The opposing player scratches the cue ball into a pocket, or knocks it off of the table, when the eight ball is pocketed. A scratch or foul is not loss of game if the 8-ball is not pocketed or jumped from the table
***** The opposing player commits any foul on the shot that pocketed the 8 ball (in non-tournament situations, non-cue-ball fouls may be excused from this requirement)
***** The opposing player knocks the 8 ball off of the table

RULES FOR PLAYING EIGHT BALL

American-style eight-ball rules are played around the world by professionals, and in many amateur leagues. The rules for eight-ball may be the most contested of any billiard game. There are several competing sets of "official" rules. The non-profit World Pool-Billiard Association (WPA), with national affiliates such as the Billiard Congress of America (BCA), promulgates the World Standardized Rules for amateur and professional play.

The for-profit International Pool Tour has also established an international set of rules for professional and semi-professional play, used in major tournaments broadcast on television. Meanwhile, many amateur leagues, such as the American Poolplayers Association (APA) / Canadian Poolplayers Association (CPA), and the Valley National Eight-ball Association (VNEA) / VNEA Europe, use their own rulesets as their standard (most of them at least loosely based on the WPA/BCA version), while millions of individuals play informally using colloquial rules which vary not only from area to area but even from venue to venue.

FOULS

***** The shooter fails to strike one of his/her own object balls (or the 8 ball, if all of said object balls are already pocketed) with the cue ball, before other balls (if any) are contacted by the cue ball. This includes "split" shots, where the cue ball strikes one of the shooter's and one of the opponent's object ball simultaneously.
***** No ball comes into contact with a cushion or is pocketed, after legal cue ball contact with the (first) object ball (or 8 ball).
***** The cue ball is pocketed.

***** The shooter does not have at least one foot on the floor (this requirement may be waived if the shooter is disabled in a relevant way, or the venue has not provided a mechanical bridge)
***** The cue ball is shot before all balls have come to a complete stop from the previous shot.
***** The cue ball is struck more than once during a shot.
***** The cue ball is jumped entirely or partially over an obstructing ball with an illegal jump shot that scoops under the cue ball.
***** The cue ball is clearly pushed, with the cue tip remaining in contact with it more than momentarily.
***** The shooter touches the cue ball with something other than the tip of the cue.
***** The shooter touches any other ball (with body, clothing or equipment), other than as necessary to move the cue ball when the player has ball-in-hand ***** The shooter knocks a ball off of the table.
***** The shooter has shot out-of-turn.
***** On the break shot, no balls are pocketed and fewer than four balls reach the cushions (in which case the incoming player can demand a re-rack and take the break or force the original breaker to re-break, or may take ball-in-hand and shoot the balls as they lie)

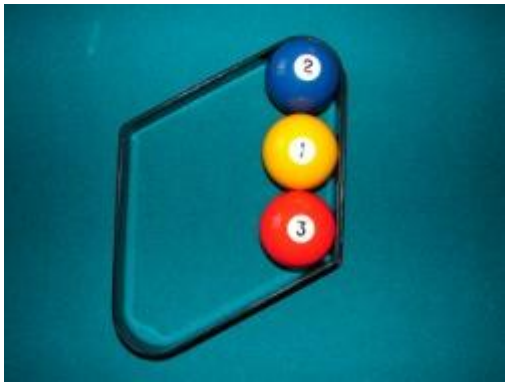
3.8.2. 3-Ball Game

TYPE OF GAME

Three ball is a pocket billiards folk game played with three standard pool object balls and a cue ball. The goal is to pocket the three object balls in as few shots as possible. Theoretically, any number of players can participate, in rotation, but more than five can become unwieldy. The game involves a somewhat more significant amount of luck than either nine-ball or eight-ball, because of the disproportionate value of pocketing balls on the break shot.

THREE BALL RACKS

Three object balls (conventionally the 1, 2 and 3 balls) are racked either in a triangle — like a miniature eight-ball or snooker rack — with the apex ball on the foot spot, or in a straight line, again with the lead ball on the foot spot, and the other balls behind it, lined up toward the center of the foot rail. No particular arrangement is necessary, as there is no specific order in which the balls must be pocketed, nor do any of them have specific point values. (see pictures below)



OPENING BREAK

In three ball, players' turn order is decided at random at the beginning of the game or match, as in other several-player pool games. The cue ball is placed anywhere behind the head string ("in the kitchen") and a typical hard break (as in nine-ball or eight-ball) is performed. The break is the first stroke of a player's game, and thus counts toward his or her score. Any balls pocketed on the break are considered to be legally pocketed and the player now only has to sink the remaining balls.

Very good players can sink all three object balls on the break with surprising frequency, resulting in the perfect (but still tieable) score of one point, especially if the balls are triangle-racked; this feat is achieved using an adaptation of the instant-win break technique from eight-ball and nine-ball; the straight rack was introduced to make this more difficult, as it does not provide the contact point and angles that the well-known technique requires.

FOULS IN THREE BALL

Every shot costs one point, and a foul of any kind costs the player an additional one-point penalty. Fouls consist of: pocketing the cue ball; knocking the cue ball off the table; a double hit on the cue ball with the cue stick (including illegal "scoop-under" jump shots); push shots; and (possibly, depending on how serious the game is) accidentally (or otherwise) moving a ball with a hand, the butt of the cue, etc. A shot in which the player pocketed one or more object balls but also fouled incur a one point penalty - a foul always results in a penalty of 1 point. Thus, a break shot that sank all three object balls plus the cue ball is a score of two (one for the actual shot, plus one for the foul), unless the "instant loss" rule (see below) is in effect.

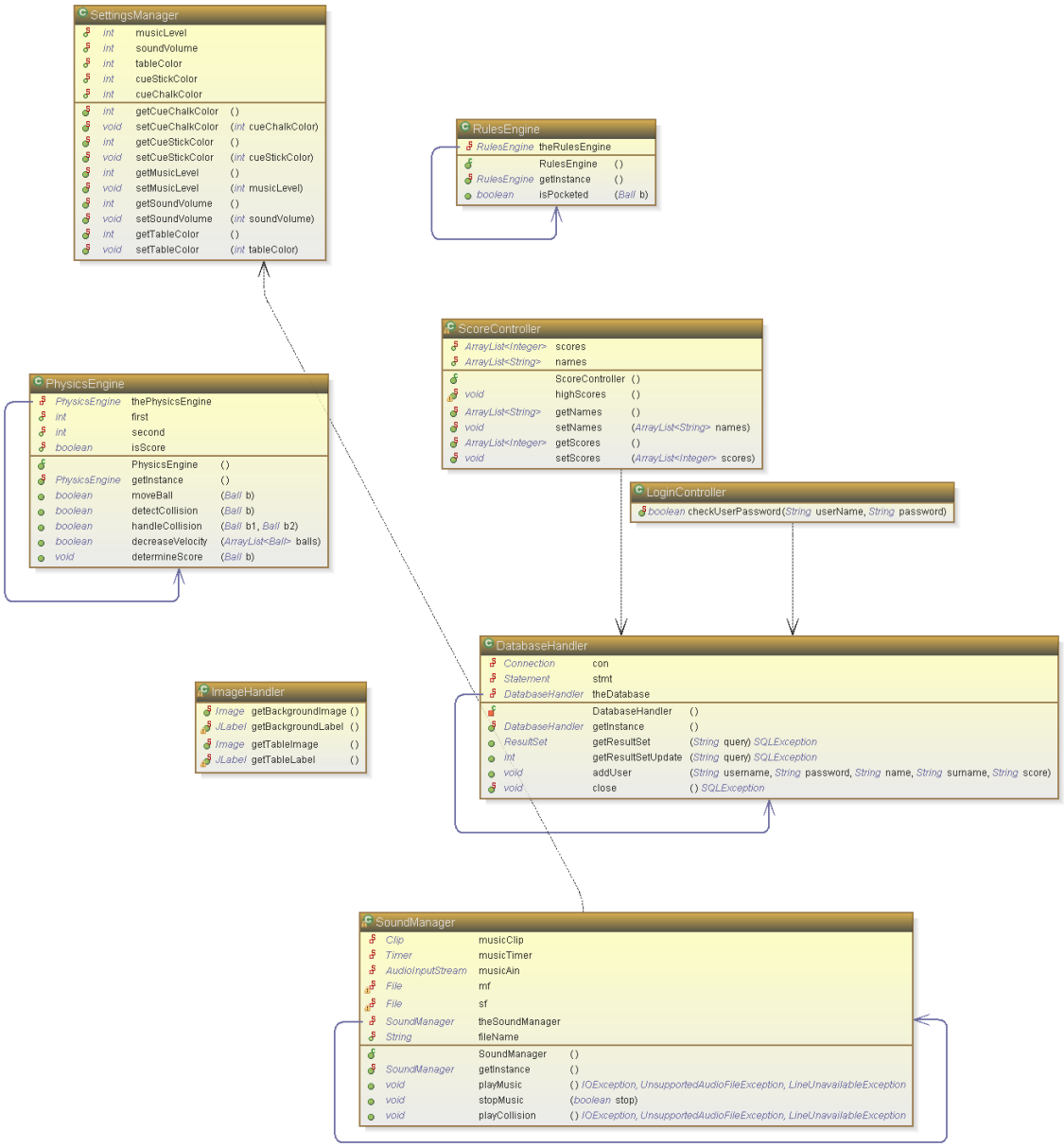
It is not a foul to make a weak shot that does not pocket a ball or contact a cushion, since, again, these mistakes are effectively self-punishing, by costing the player a stroke.

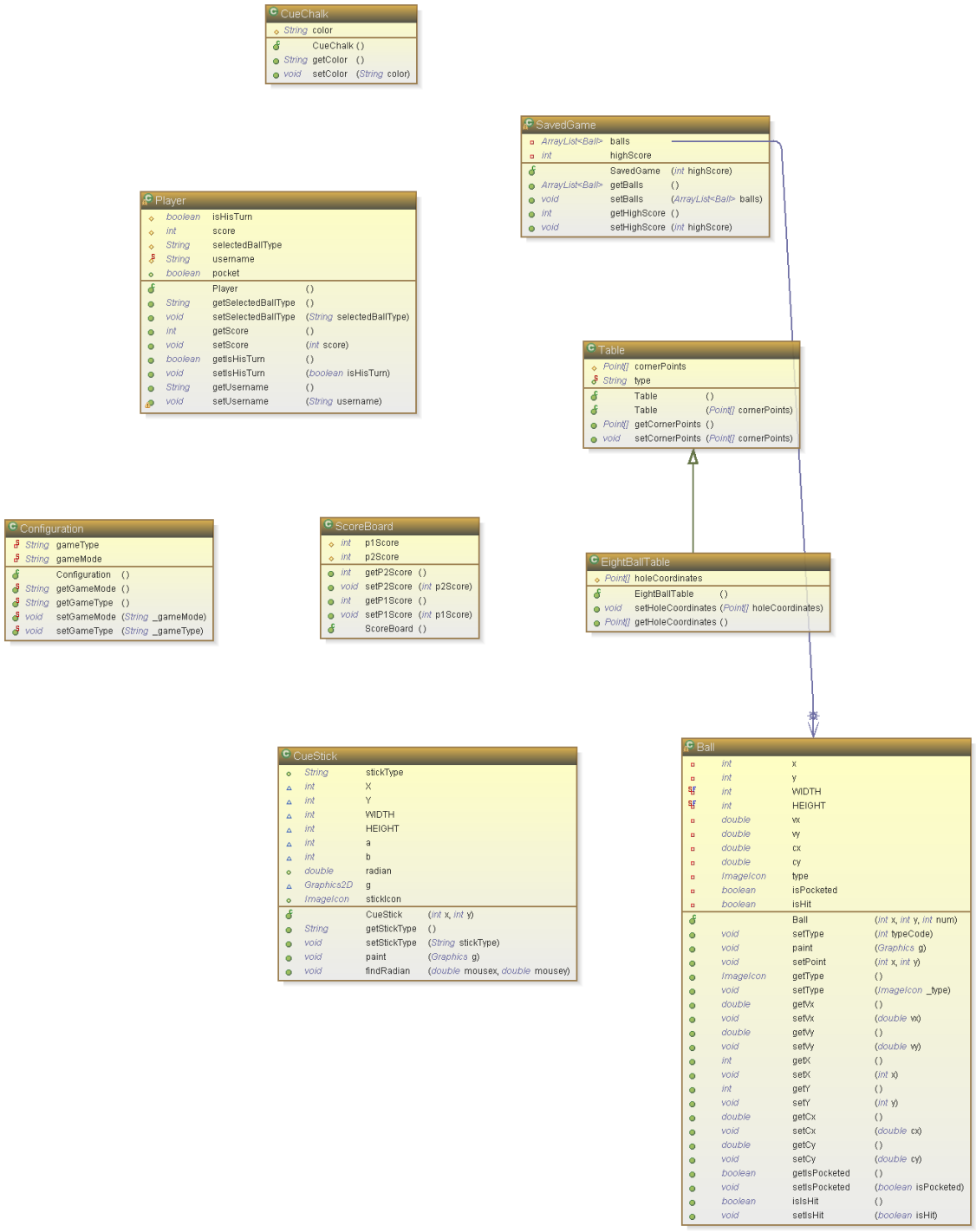
4. ANALYSIS MODELS

4.1. Object Model

4.1.1. Class Diagram

Billiard game consists of a table (3BallsTable or 8BallsTable), balls (3 or 8), a player, a cue chalk, a cue stick and a score board.




CueChalk
String color
CueChalk ()
String getColor ()
void setColor (String color)
SavedGame
ArrayList<Ball> balls
int highScore
SavedGame (int highScore)
ArrayList<Ball> getBalls ()
void setBalls (ArrayList<Ball> balls)
int getHighScore ()
void setHighScore (int highScore)
Player
boolean isHisTurn
int score
String selectedBallType
String username
boolean pocket
Player ()
String getSelectedBallType ()
void setSelectedBallType (String selectedBallType)
int getScore ()
void setScore (int score)
boolean getIsHisTurn ()
void setIsHisTurn (boolean isHisTurn)
String getUsername ()
void setUsername (String username)
Table
Point[] cornerPoints
String type
Table ()
Table (Point[] cornerPoints)
Point[] getCornerPoints ()
void setCornerPoints (Point[] cornerPoints)
Configuration
String gameType
String gameMode
Configuration ()
String getGameMode ()
String getGameType ()
void setGameMode (String _gameMode)
void setGameType (String _gameType)
ScoreBoard
int p1Score
int p2Score
int getP2Score ()
void setP2Score (int p2Score)
int getP1Score ()
void setP1Score (int p1Score)
ScoreBoard ()
EightBallTable
Point[] holeCoordinates
EightBallTable ()
void setHoleCoordinates (Point[] holeCoordinates)
Point[] getHoleCoordinates ()
CueStick
String stickType
int X
int Y
int WIDTH
int HEIGHT
int a
int b
double radian
Graphics2D g
ImageIcon stickIcon
CueStick (int x, int y)
String getStickType ()
void setStickType (String stickType)
void paint (Graphics g)
void findRadian (double mousex, double mousey)
Ball
int x
int y
int WIDTH
int HEIGHT
double vx
double vy
double cx
double cy
ImageIcon type
boolean isPocketed
boolean isHit
Ball (int x, int y, int num)
void setType (int typeCode)
void paint (Graphics g)
void setPoint (int x, int y)
ImageIcon getType ()
void setType (ImageIcon _type)
double getVx ()
void setVx (double vx)
double getVy ()
void setVy (double vy)
int getX ()
void setX (int x)
int getY ()
void setY (int y)
double getCx ()
void setCx (double cx)
double getCy ()
void setCy (double cy)
boolean getIsPocketed ()
void setIsPocketed (boolean isPocketed)
boolean isIsHit ()
void setIsHit (boolean isHit)

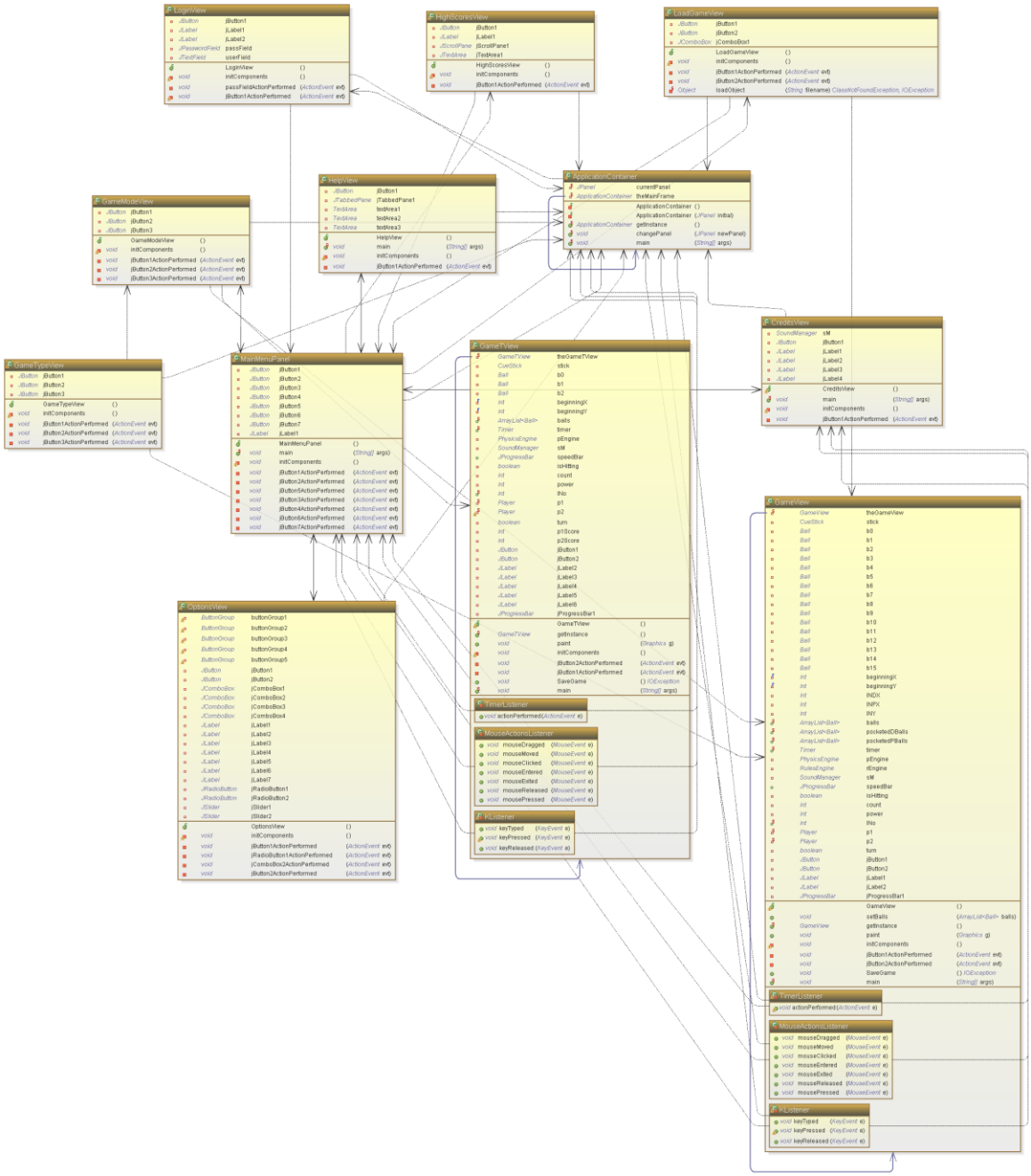


Figure 2: Domain Analysis

Billiard is a table game; it is playable on the table. All balls are stays on the table and player tries to send balls into holes by using cue stick. Player can change the color of the table as s/he wishes. Table property can be chosen by the player on the settings. Table has some differences depending on the type of the game. If 8-ball game has chosen by player, table has 6 holes. If 3-ball game has chosen by player, table has no holes. There are not any more differences for different types of games.

Ball numbers and types are changes according to chosen game type. If 8-ball game is selected, game consists of 16 balls. 14 of them are colored balls and categorizes as follows; 7 of them are marked with lines, 7 of them is smooth, other one is black ball and last one is white ball. If 3-ball game is selected, game consists of 3 balls. Ball colors and types can be changed by the player as he wishes on the settings menu.

Score board is shown while playing is continuing. When player gain a score, score is updated on the score board. Both players' scores are shown during the game.

Player is the one who plays the game. Player can score on the game by using cue stick. Player can adjust the properties of game table and balls. Colors can be selected as player wishes. There might be one or two players depending on the game mode. If Computer vs. Player mode is selected, player can play against computer. Otherwise; two players can be play against each other.

Cue Stick is the main term of the Billiard game. Player hits balls by using Cue Sticks. Cue Stick color can be chosen by the player. Cue stick angle can be adjusted by the player to send towards specific ball. After adjusting angle, player can adjust the hit power. Player can increase the cue stick power if player wishes to send balls faster or vice versa. Cue stick accuracy can be fall down while hit number is increasing.

Cue chalk is helps player to increase his/her cue stick's accuracy. Cue chalk color can be adjusted by the player on settings.

4.1.2. Domain Lexicon

- GuiManager(Display): displays the states of the game to the user
- CueStick: Entity that enable the player to hit the ball
- Ball: Entity that exists on the table and hit by cue stick in the aim of pocketing them
- Table: Entity that keeps the ball on
- ScoreBoard: Entity that keeps the current scores of the player(s)
- GameManager: Data which coordinates the other components of the system
- PhysicsEngine: Data which handle the movement of the balls
- RulesEngine: Data which controls the rules of the game

4.2. Dynamic Model

4.2.1. Sequence Diagrams

Sequence Diagram 1:

Enes opens up the game and selects “Options” from the main menu. Then options screen is showed. In this screen Enes selects a music for listening that song during the game. Then change the background picture from default picture to amother one. Finally saves the options and returns to main menu.

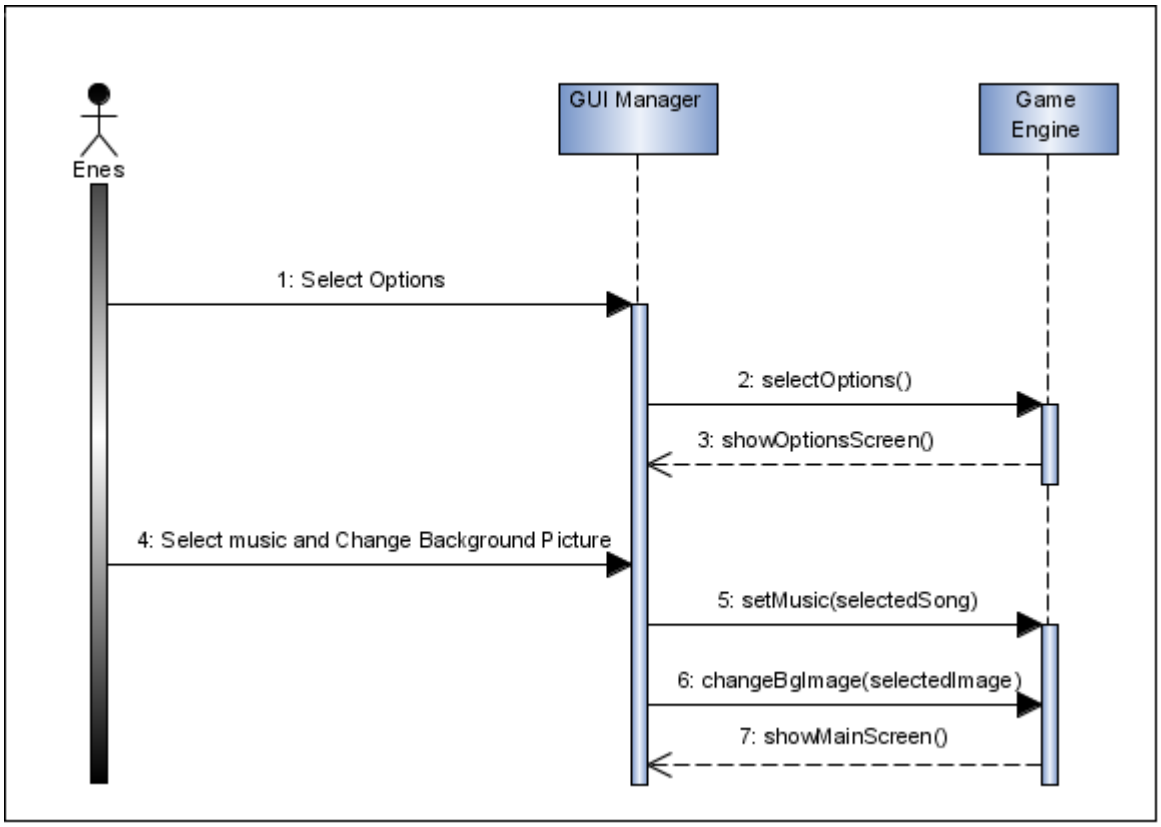


Figure 3. Sequence Diagram for changing some game options

In this sequence diagram, player executes the game. Then he requests to make some changes about game options. GUI Manager contains the main window of the game. It also contains the other important windows and their components visually. Game Engine is the crucial part of the system which handles almost every event. It takes necessary information from the GUI Manager and processes them accordingly. In here it is just adjusting the music and background picture of the system.

Sequence Diagram 2:

Omer opens up the game and selects “Play Game” from the main menu. Then he selects the type and mode of the game as 8-ball pool and Player vs. Player respectively and starts the game.

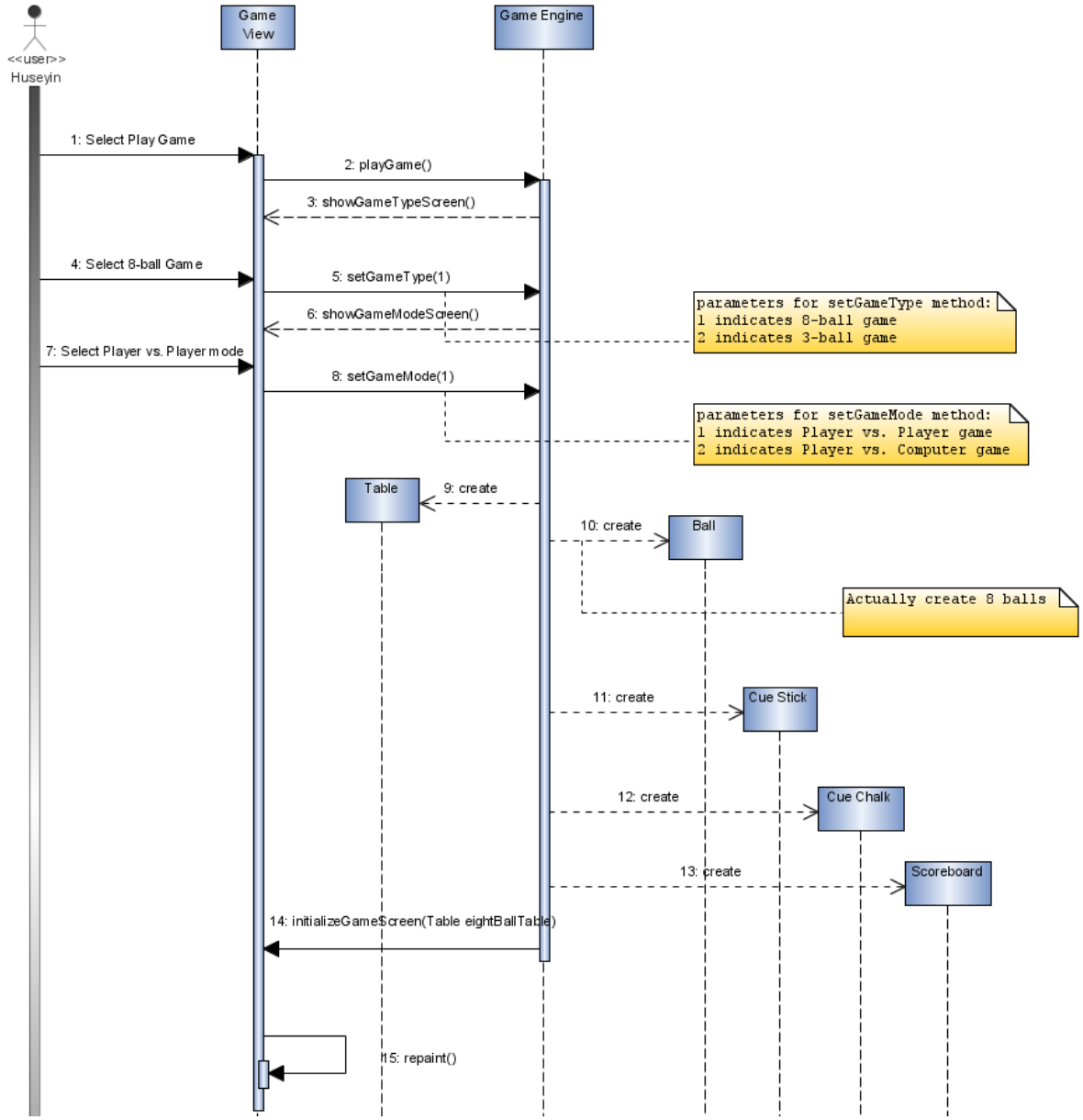


Figure 4. Sequence Diagram for starting a billiard game

In this sequence diagram, player executes the game and selects the type and mode of the game and starts a new billiard game by sending requests to GUI Manager which will open the Game Engine.

Player selects 8-ball type as the game type and based on this selection remaining of the system will be adjusted accordingly such as pool table will have pockets and there will be 8 standard balls and a black ball and a cue ball which is white. Before game starts, cue stick and cue chalk is created also.

Sequence diagram for playing Player vs. Player 8-ball game and Player vs. Computer 3-ball game are given in the next diagrams. This scenario only includes starting the game.

Sequence Diagram 3:

Huseyin and Alperen decide to play the 8-ball billiard game with each other. Huseyin starts to play and makes the first shot. Then they hit the cue ball turn by turn and tries to pockets all balls. And in the end the score of the winner is saved into the hicg scores. (**Figure 5.**)

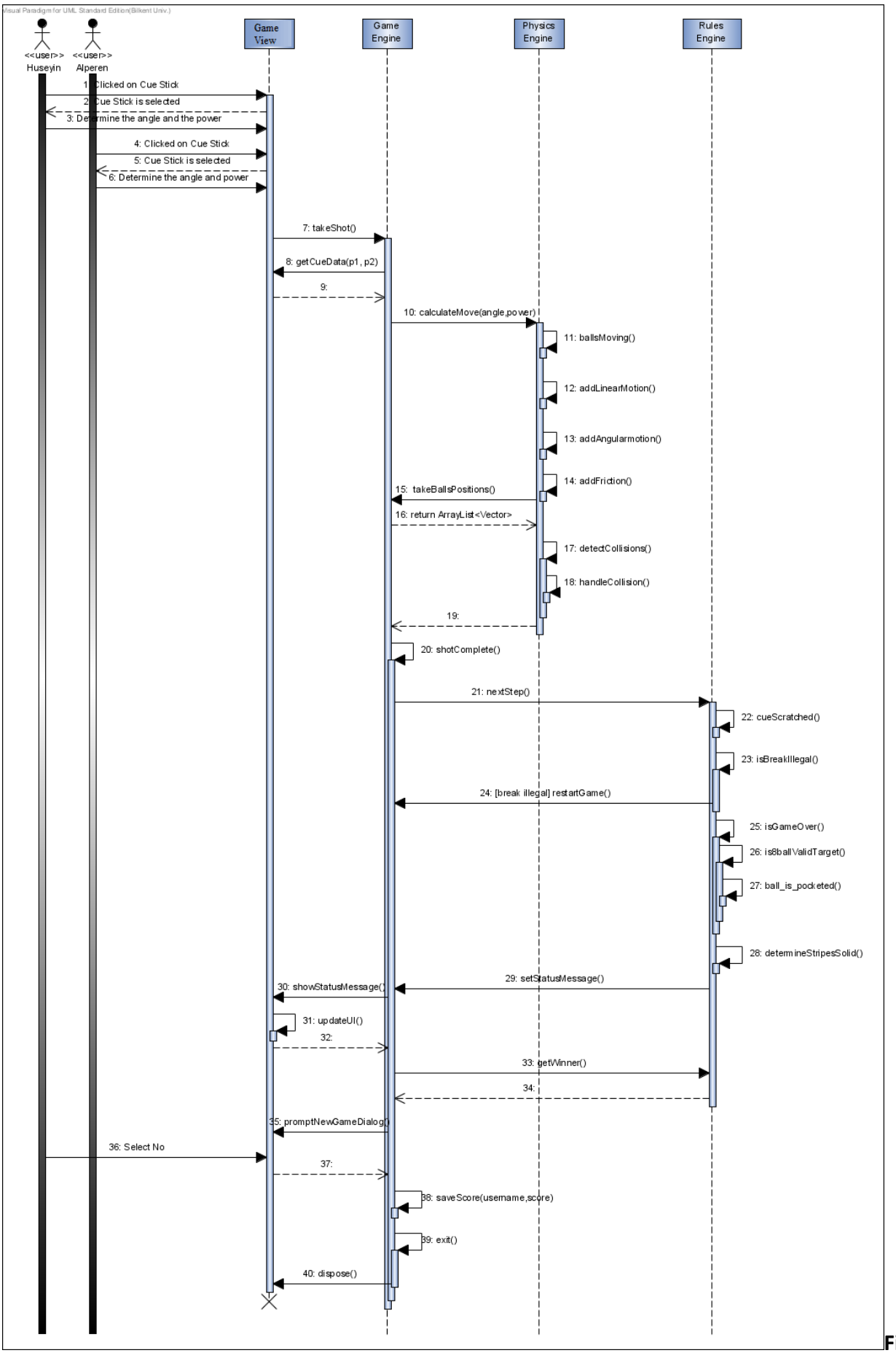


Figure 5. Sequence Diagram for playing 8-ball game

In this sequence diagram, players request to play 8-ball game with each other and starts the game. In this diagram we assume that we have the shots that players made at the beginning for ease of understand, i.e. we have the angle and the power of the shot and makes the shots recursively as seen on the diagram. Huseyin makes the first shot and Game Engine is warned about the shot. It takes the necessary parameters and passed them into the Physics Engine in order to calculate the movement of the balls. Physics Engine also takes the positions of the other balls for detecting and handling the collisions. After the shot is completed, we need to check some rules in order to be sure that the shot is legal, valid or not, if it is not game will be restarted. If the shot is valid then, Rules Engine determines the type of the balls, stripes or solids, that Huseyin is going to pocket in the remaining of the game. Rules Engine always check the pocketed ball if it is a wrong ball, like Huseyin should pocket stripes but he pockets solid, turn is changed or if the pocketed ball is the black ball, number 8, which causes the game over. After every shot, components are repainted by the GUI Manager. When the game ends user is prompted whether he wants to play again or not and Game Engine takes score of him and adds him to the hall of fame.

Sequence Diagram 4:

Huseyin starts the game and decides to play the 3-balls billiard game against computer. He starts the game and makes the first shot. If he is unable to make his ball collides with the other two balls, it becomes computer's turn and vice versa. When the game ends, the score of Huseyin is saved into the hall of fame.

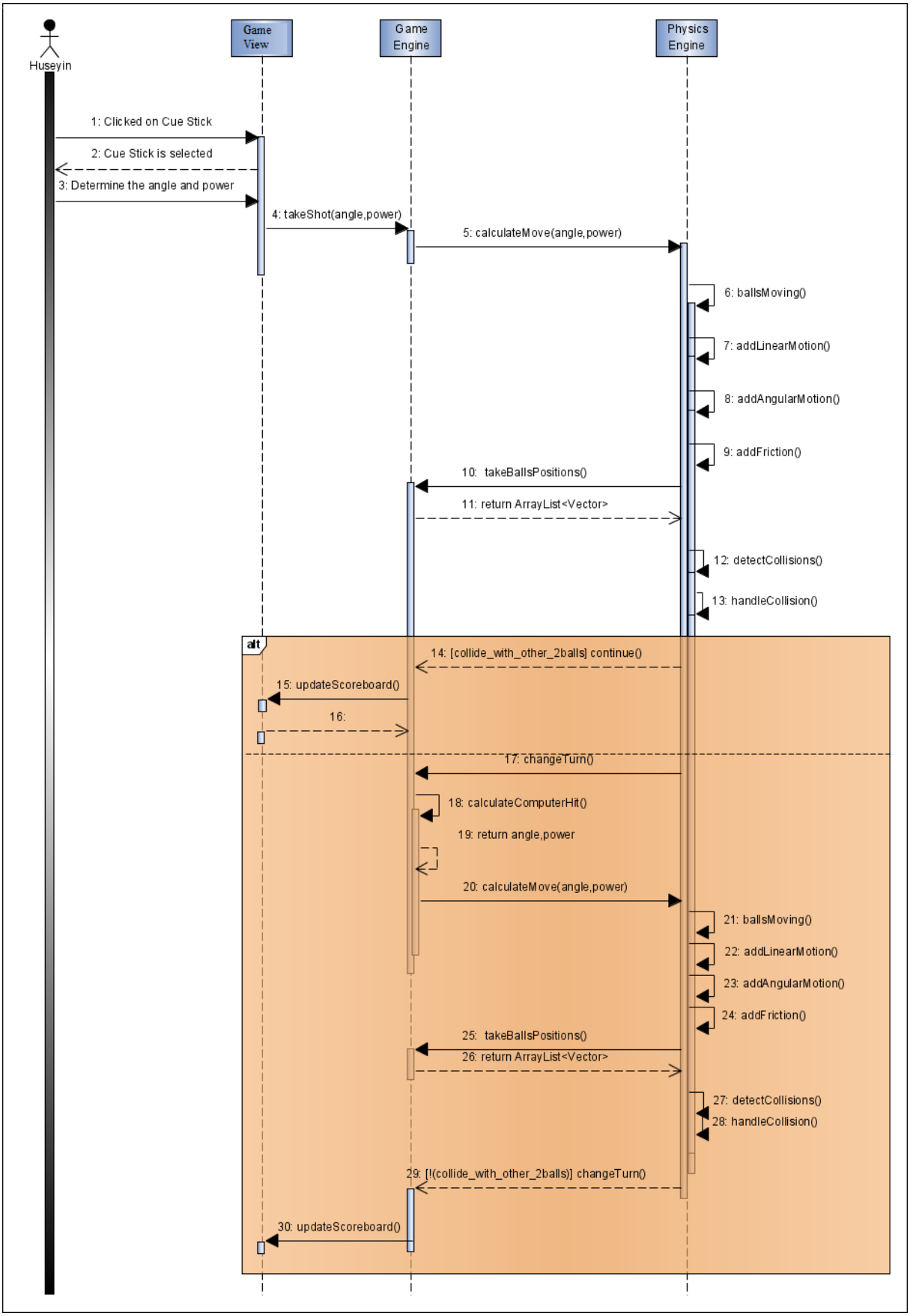


Figure 6. Sequence Diagram for playing 3-ball game

During the game, Huseyin clicks on Cue Stick and selects it. Then by arrow keys or mouse, he adjusts the angle and power of Cue Stick. After Huseyin makes the shot, GUI Manager forwards that to Game Engine. Game Engine sends a calculate request to Physics Engine with power and angle parameters that player decided. After balls are started to move, linear and angular motion and friction are added by Physics Engine. Current locations of balls are taken from Game Engine to Physics Engine and it calculates and handles the collisions.

If white ball touches both of other two balls, score board is updated and Huseyin continues to play. If it does not touch, then it is computer's turn, Game Engine decides power and angle that computer will strike. Then, again Game Engine sends a calculate request to Physics Engine with power and angle parameters that computer decided. After balls are started to move, linear and angular motion and friction are added by Physics Engine. Current locations of balls are taken from Game Engine to Physics Engine and it calculates and handles the collisions.

4.2.2. State Diagrams

4.2.2.1. Ball State Diagram

Changes in the states of the ball objects in game will occur only when white ball is struck. After white ball is struck, there are two possibilities, either it will touch other ball/s or it won't touch. Besides, after striking or missing, white ball can either go into a hole or stop without touching any of the balls.

Except white ball, we have a black ball and colored balls. If white ball strikes to these balls, there are again two possibilities. Balls can go into a hole or stop after moving for a time.

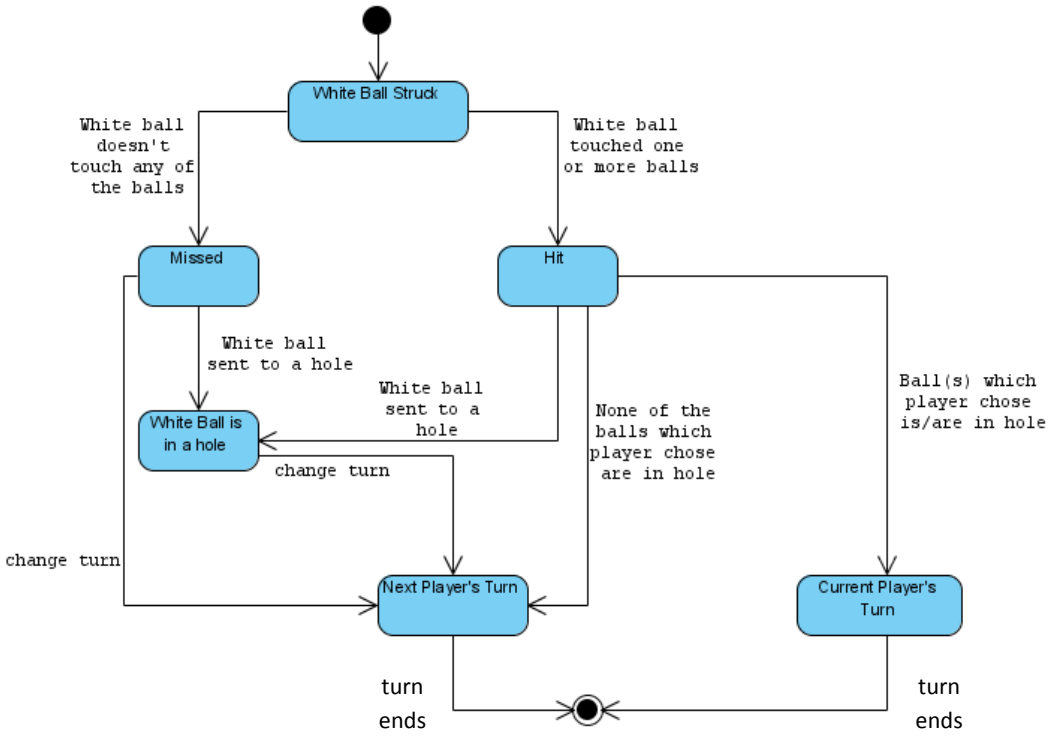


Figure 7: Ball State Diagram

4.2.2.2. Table State Diagram

Table contains objects like ball and cuechalk in game. Only the color property of table object can be changed.

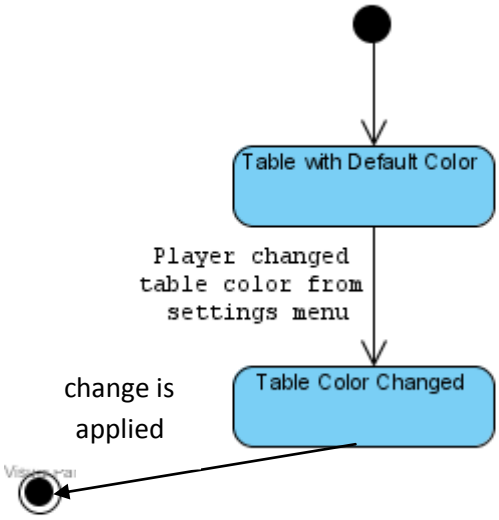


Figure 8: Table State Diagram

4.2.2.3. Score Board State Diagram

When a new game starts, all players have 0 points. When a player makes a successful shot, score board is updated. If the player cannot make a successful shot, score board remains the same.

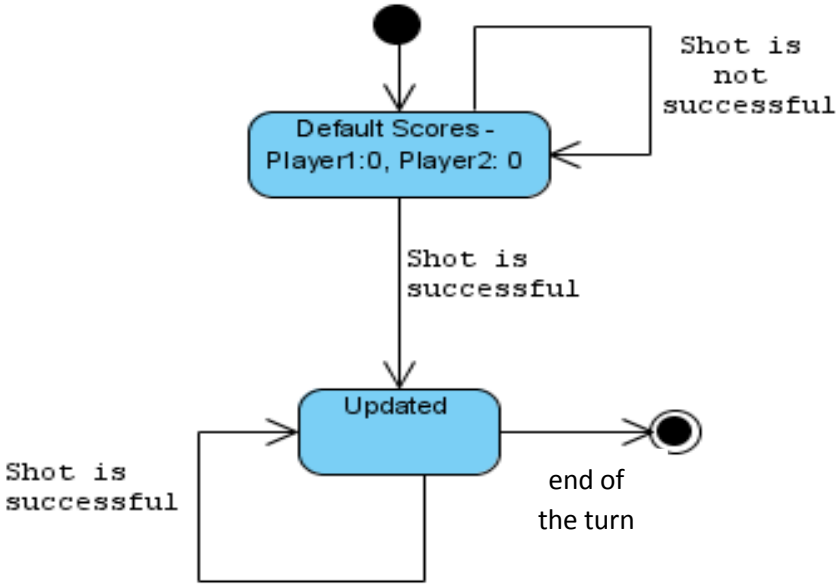


Figure 9: Score Board State Diagram

4.2.2.4. Player State Diagram

After a player started the game, when first colored ball is sent into a hole, player must choose the ball type s/he will try to put into holes. For every player, his/her score is hold and updated according to flow of the game. If s/he scores, his/her score is increased, otherwise, in other words if s/he can't put any of his/her balls into a hole, turn will pass to the other player.

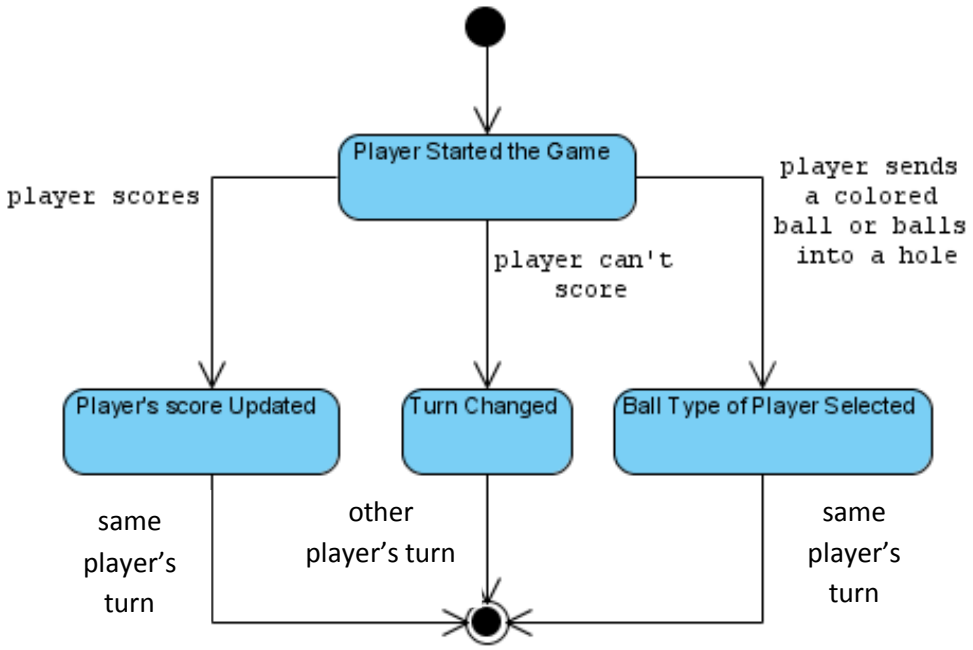


Figure 10: Player State Diagram

4.2.2.5. Cue Stick State Diagram

Player starts the game with a default cue stick which color is brown and has a 100% accuracy. From the settings in main menu, player is able to change the color of cue stick. During the game, player is able to set the direction of the cue stick, that is to say, where s/he wants to send the white ball. Furthermore, since accuracy of cue stick will decrease from turn to turn, player can use cue chalk to increase the accuracy of cue stick. At the end of every turn, accuracy of cue stick is decreased.

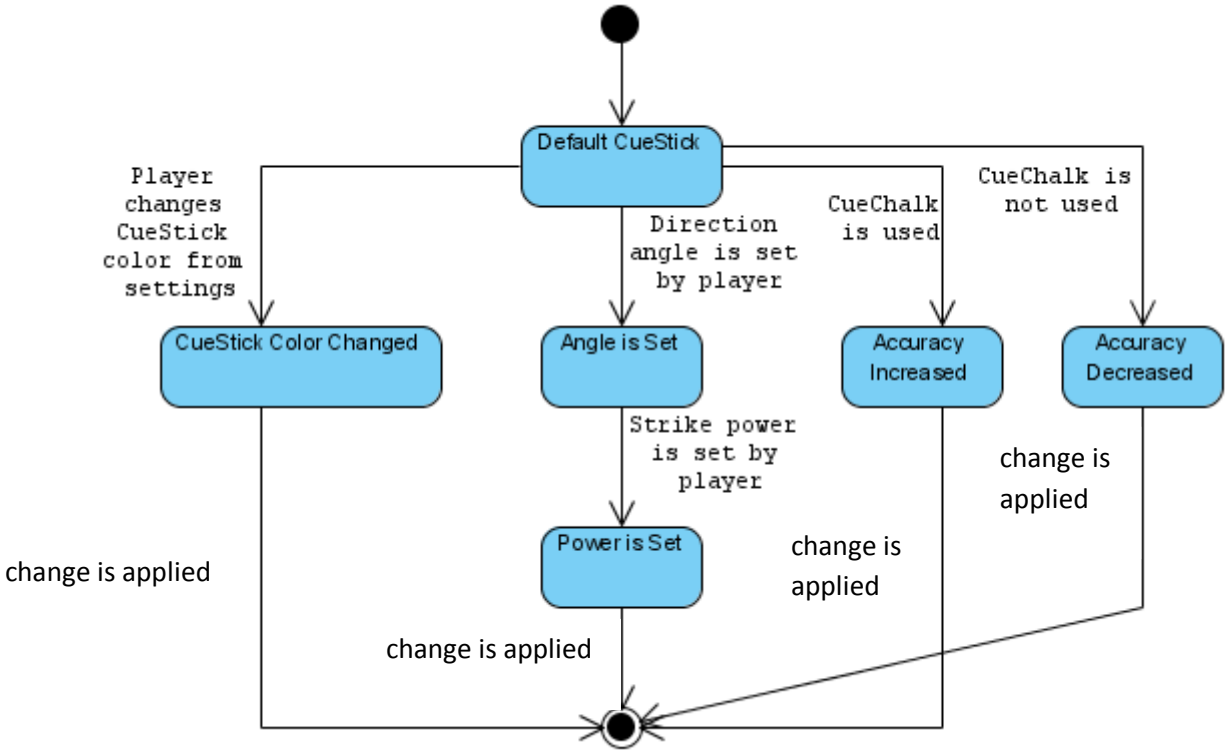


Figure 11 : Cue Stick State Diagram

4.2.2.6. Cue Chalk State Diagram

Cue chalk is an object for increasing the accuracy of the cue stick. Only the color property of cue chalk object can be changed.

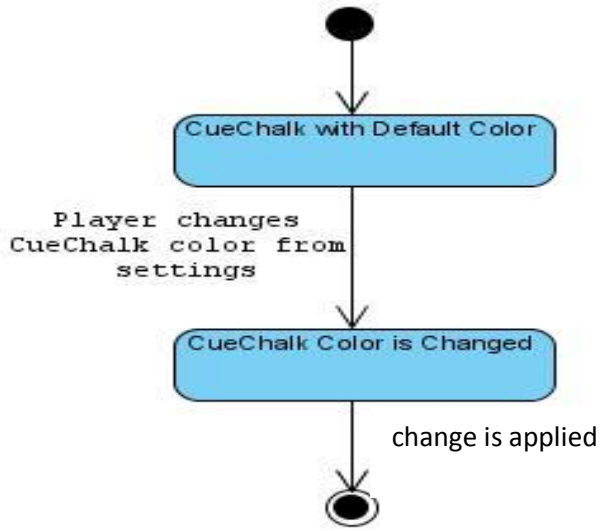


Figure 12: Cue Chalk State Diagram

5. SYSTEM DESIGN

5.1. Design Goals

Our primary design goal is to create a robust system for our billiard game. Robustness of a system is related with performance of a program under not only ordinary conditions but also unordinary conditions. So, billiard game will be able to handle exceptional conditions and it will do what is expected from it. We consider using design patterns which helps us to achieve low coupling/high cohesion, which will particularly be crucial during implementation and testing phases. Low coupling will provide less dependency among the objects which will decrease the complexity of the system. Hence, any change that will be made won't affect the others. High cohesion aims to decrease propagation between components. So, changes that will be made can be easily placed and will not have an effect on other components whereby high cohesion. In other words, every component will carry out one task. Another goal that we highlight is having a system that can be modified and understood easily. Model-View-Controller design pattern would be an ideal choice for us to achieve that goal. In Model-View-Controller design pattern, view, logic and control of a system are held separately. So, modification will be much easier by MVC design pattern. Reliability is another design goal in our project that is for preventing user from doing wrong things. With reliability, program has the ability to correct user mistakes. After achievement of these design goals, a playable, error-free billiard game with high performance will come out. Another goal we want to accomplish is to create an efficient system. Efficiency and high performance is important in our project since there are a lot of dynamic objects. Besides, we want to integrate sounds and music to our game. This would cause slowness in game but with a good design, we can prevent it. Fault tolerance is the last design goal we consider. We aim to design a system that will be easily fixed if some errors occur. Thus, it will not affect other components in system, it will be able to resolve its error in its own.

5.2. Subsystem Decomposition

In our Billiard Game there are 3 main subsystems: BilliardView, BilliardController and BilliardModel. Also each main subsystem has its own subsystems.

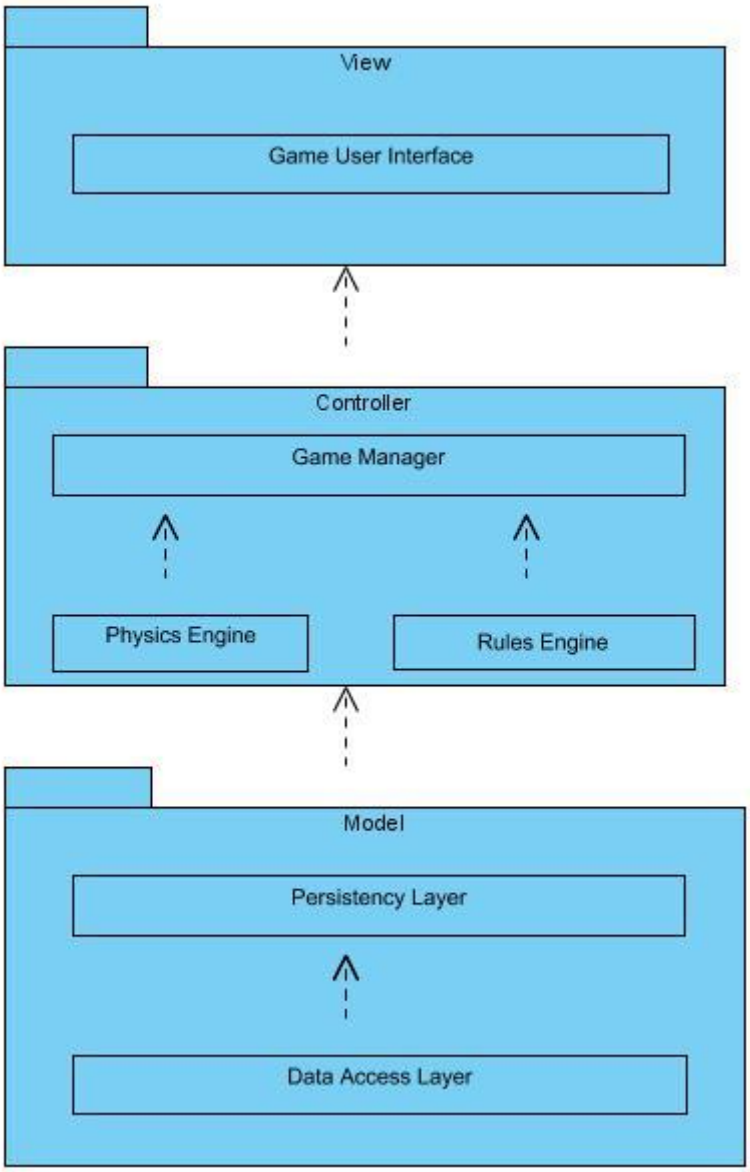


Figure 1. Package Diagram and Subsystem Decomposition of the System

Main hierarchy can be shown as:

- BilliardView
    - GameUserInterface
- BilliardController
    - GameManager

- PhysicalEngine
- RulesEngine

- BilliardModel
  - PersistencyLayer
  - DataAccessLayer

BilliardView

This subsystem is responsible for game UI for end user. It consists of frames, panels, buttons etc. (everything to provide functionality to the user). Also BilliardView manages transitions between different panels.

BilliardController

This subsystem is the place where all game logic is located. When user interacts with the game by entering menus, clicking buttons, hitting balls etc. functionality is provided to the user by channeling control from view to this subsystem. Also Billiard logic (collision of balls, their direction after collision- whether the balls hit the edge of table etc.) are all managed by BillardController's subsystem PhysicsEngine. PhysicsEngine manages the game always consulting to the RulesEngine (checks whether the game flow conforms to the Billiard rules). BilliardController coordinates these two subsystems (PhysicsEngine and RulesEngine) with GameManager.

BilliardModel

Here is where blueprints of states of the BilliardGame and all saved data are kept. All domain objects' classes are here and they are initialized and used by BilliardController. Also saved data (saved games and high scores) are read and written via DataAccessLayer. After first access, they are placed in PersistencyLayer for future use.

5.3. Architectural Patterns Used

In Billiard Game we used two main design patterns out of 5 listed in our lectures. They are Layered Pattern and MVC (Model – View – Controller) Pattern (Figure -1).

We used MVC Pattern to identify main subsystems that are BilliardView, BilliardConroller and BilliardModel. As their name suggest they are correspondent parts of View, Controller and Model. As the controller, BilliardController gets user actions and responds to them. To form the response, it identifies and determines necessary actions by getting state information from model and offers the result to the user by determining how UI should be rendered.

To assist BilliardController, BilliardModel saves the state of the game. It consists of two layers (Layered Pattern). The lowest layer in hierarchy, Data Access Layer, saves raw data to xml files by executing IO operations. Its services are used by Persistency Layer through CRUD operations on xml files. Persistency Layer also gets (loads) data from Data Access Layer to populate its domain objects. This domain objects are what BilliardController gets from BilliardModel. Our decision to put a Persistency Layer here is to provide a central mechanism that gets raw data from xml (via Data Access Layer) saves it in RAM and use it a place that we can get domain objects easily (This layer can be thought as a very simple version of Java Persistence API or ADO.NET in .NET world besides a container for domain objects).

BilliardController has two layers which are Game Engine (core logic) and the lower one consists of Physics Engine and Rules Engine. Physic Engine offers services to GameEngine. These services provides logic and result of operations of ball collision, their angles after that collisions, balls locations etc. Rules Engine has services that determine the rules of the game. Game Manager continuously checks current state of the game (whether it is in the constraints of the Billiard game) by using these services.

Finally, apart from these two patterns we have designed the architecture for our system as a client-server system. In this architecture we have used client-server pattern. Additional information about the client-server system can be obtained from Hardware-Software mapping part of the report (5.4).

5.4. Hardware Software Mapping

Our Billiard game runs on one PC at a time, i.e. a stand-alone system. Below, there is the deployment diagram of the stand-alone system.

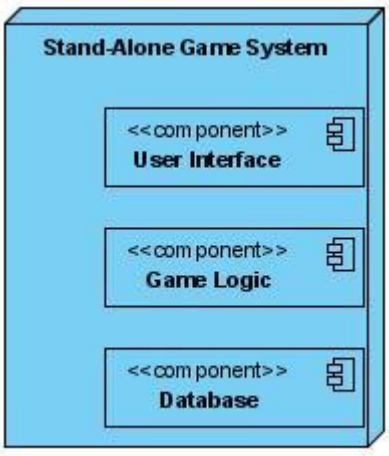


Figure 2. Stand-Alone System Deployment Diagram

However, our system can be extendible by including client-server architecture through network. Since we do not have time for implementing such a wide range system, our game will work

as a stand-alone system. As seen from the below diagram, there will be used client-server architectural pattern. (That will be another architectural pattern for our system. We have talked with the instructor and he suggests us that architecture to design even though we will not be able to implement it.)

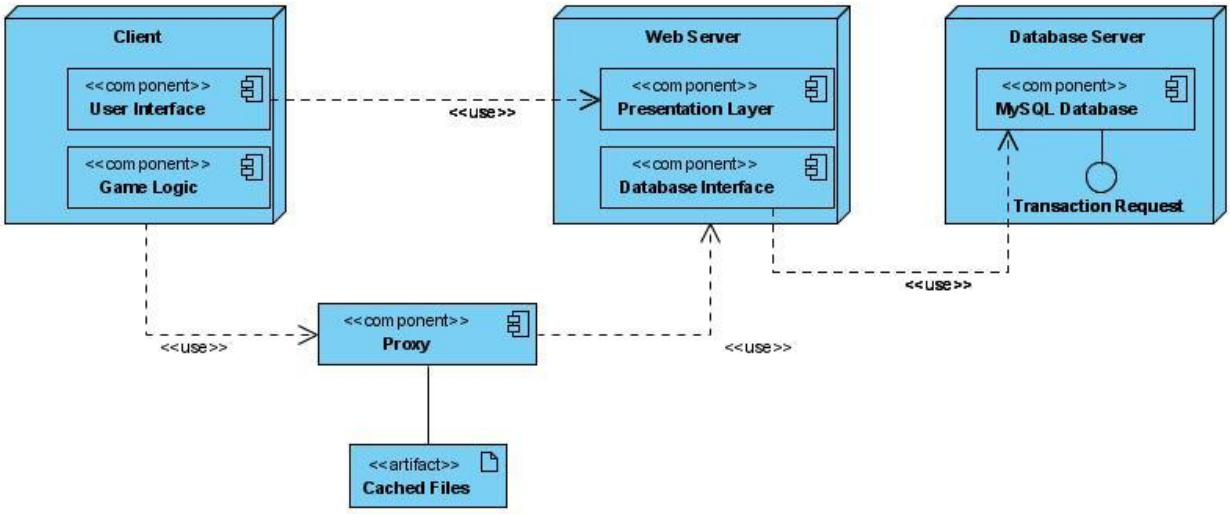


Figure 3. Client-Server System Deployment Diagram

Finally, the programming language that we will use for implementing the core design of our game project is JAVA. We will use JDK and J2SE Platforms to develop the program and corresponding JRE package will be required to run our Java application on Windows or Linux systems. We are used to implement Java programming language and we have adequate knowledge about it. Furthermore, Java has large library that helps us for practical usage. Also, Java answers our design goals that are reliable functionality and high performance since its compilers, memory management and multithreading make programs more reliable and faster. We have decided to use *NetBeans* v6.5.1 since it provides very useful tools to implement GUI components in designer's view.

5.5. Persistence Data Management

Some files of our system will be saved and maintained in order to provide the game some nonvolatile operations. We will need persistence data storage for basically three types of operations.

First one is for keeping profiles of the users and high scores. Second one is for saving the game for later use. And the last one is for keeping the game settings and songs.

First of all, we will have two types of players; one is registered player and the other one is anonymous. Registered players will have their own passwords and can login to the game with using their names and passwords. We will save personal information, i.e. player names and passwords, etc. in a relational database structure because it is easy to search and update through database instead of a file. Also while keeping high scores we will remember names of the registered users and anonymous users will be seen as anonymous. The game will keep high score list in a plain text file in order to display to the users in the Hall of Fame section. The reason for choosing file system instead of database storage is that we do not have complex system, just basically a name and a score. Moreover, for high scores we will not need any search, delete or update operations which can be implemented with a database system easier. We just need to read the scores and sort them. That's why we choose writing scores into files instead of a database system.

Next, our system will have save and load mechanism. However, only registered player can save or load their games, anonymous players do not have such luxury. The way we implement save and load operation is to use serializable interface of Java IO library. Through using this interface, we gain the ability of writing and reading of the object of a class into the hard drive, in this case this object will be the 'balls' object which will have the locations of the balls on the table. By this way we do not need to use a database for saving the game.

Finally, some images and music files are also used at different parts of the game in order to provide better experience on gaming. We will keep the paths of these files, i.e. URL, in a database and we will fetch these data from the database whenever we need to use the files. By this way we will gain flexibility for adding new songs and images to the system. The only thing that's need to be done is to add tuples in the database table for the new files.

Our database will be a relational database and we will use MySQL for database implementation.

5.6. Access Control and Security

Since our game will be worked as a stand-alone application and do not have any hierarchy like admin or moderator (we just have player), we have an access control mechanism just between the players. Besides, billiard game will be available for both authenticated users and annonymous users. Therefore, we need to control the usernames and passwords of users with saved ones.

In addition, a user will not be able to change any settings of another user. A user is not allowed to delete or rename a saved game of another user, change the ball type, table type, cuestick type and musics of another profile. An authenticated player has an access to change the table, ball and cuestick type and cuechalk color. Besides, s/he can save the game any time and load and play later. On the other hand, annonymous player doesn't have an access for all these, s/he can just play the game or create an authenticated player.

\Objects **** **Actors **	*Table*	*Ball*	*Scoreboard*	*Player*	*Cuestick*	*Cuechalk*	*Save-Load Game Handler*	*In-Game*
Authenticated Player	`changeTableType`	`changeBallType`		`logOut`	`changeType`	`changeColor`	`loadGame`	`strikeBall`
Annonymous Player				`createPlayer`				`strikeBall`

Table 1. Access Matrix of the Billiard Game System

5.7. Global Software Control

In our project, we consider to use event-driven control flow mechanism. In event-driven mechanism, flow of system is determined by events. There is an event detection part for detecting the changes made in program and a part for event handling which will decide what will be done when an event occurs. Since we will use Model-View-Controller, event-driven mechanism will be the best choice because it has separated control, view and logic parts. Thus, when an event occurs, detection will be made by a part and handling will be made by another part. Besides, our project should have a decentralized design because we plan to have more than one control object. Decentralized design stands for the distribution of dynamic behavior to objects. Our system will have physics engine and rules engine as control objects, so system should be decentralized to be able to spread responsibility. In addition to this, conformity of decentralized design to object-oriented development is another reason for choice of decentralized design.

5.8. Boundary Conditions

Initialization

- User starts the game by opening game file.
- A login screen shows up, there will be three options: one is to play the game as anonymous, other is to login as a previously created profile and the last is to create a profile.
- User chooses to login, create a profile or enter the game as anonymous.
- If user chooses to login, username and password is controlled. System must access username and passwords to check with what user entered. If it is true, user directed to main menu of the game.
- If create a profile is chosen, system must access usernames to check for the duplication in usernames. If what user entered as username is not the same with the ones in database, user directed to main menu of the game.
- If user chooses to go as anonymous, s/he is directed to main menu of the game.

Termination

- User has a chance to exit the program any time s/he wants. If s/he wants to terminate during game-play, s/he is asked to save the game.
- User has a chance to quit the game without exiting while playing (return to main menu).
- The information about the system that are kept in memory is cleaned up so that memory should be emptied and so the system could start from the beginning next time.
- If there is any high score or incomplete game, they can be saved before termination. In this case, username, name of the saved game and final position in the game is sent to database and they are saved to database.
- Any unsaved game or score is lost if termination takes place.

Failure

- If database connection is lost, network connection problem occurs in client-server architecture.

- Checkpoint system can be applied to prevent database connection problem. System is connected to database and required data is written regularly. If a problem occurs, system will be returned to its previous errorless state.

6. OBJECT DESIGN

6.1. DESIGN PATTERNS

1. **Observer Pattern**

   Observer pattern is used for observing the changes in an object's state. It doesn't know anything except the changes in the state of objects it's tracking. In our project Billiard, we use Observer pattern to track the coordinates of the balls and cuestick. If there is a change in one of these objects, Observer notifies the view to update its state.

2. **Façade Pattern**

   Façade pattern encapsulates a complex subsystem within a single interface object. Thus, it reduces complexity by grouping common properties of classes in one interface. In DatabaseHandler class of our project Billiard, we use this pattern to simplify database connection, access issues. We group all database classes under DatabaseHandler since they use many common operations . Besides, GameEngine is another component that we apply Façade pattern.

3. **Abstract Factory**

   Abstract factory let coder to take care of the bigger picture more than the unnecessary future details. By the help of abstract factory design pattern, similar user interface frameworks can be inherit from one common interface. In Billiard Game, since we have

similar user interfaces, we decided to apply abstract factory by ApplicationContainer class. For example, if we want to develop another window for Billiard game, we won't have much difficulty in creating that because this new interface will have many common features with Application Container.

6.2. CLASS INTERFACES

Controllers:

1) DatabaseHandler.java

DatabaseHandler class holds the information about user accounts. User names and passwords are kept in this class.

DatabaseHandler
-con: Connection -stmt: Statement -theDataBase: DatabaseHandler
+getInstance(): DatabaseHandler +getResultSet(query:String): ResultSet +getResultSetUpdate(query:String): int +addUser(username:String,password:String,name:String,surname:String,score:String) +close()

2) ImageHandler.java

ImageHandler class design the background photograph on the main panel.

ImageHandler
+ getBackgroundImage(): Image + getBackgroundLabel(): JLabel + getTableImage(): Image + getTableLabel(): JLabel +paint(g: Graphics)

3) LoginController.java

LoginController checks the passwords of the user whether it is right or wrong password. If it is true user can continue otherwise user gets a warning message about wring password.

LoginController
+checkUserPassword(username: String, password: String): boolean

4) PhysicsEngine.java

PhysicsEngine manages the following concepts; movement of balls, detects the collisions of balls, handles the collisions and decreases the velocity of the balls in time.

PhysicsEngine
- thePhysicsEngine: PhysicsEngine
+ getInstance():PhysicsEngine + moveBall(b: Ball): Boolean + detectCollision(b: Ball): Boolean + handleCollision(b1: Ball, b2: Ball): Boolean + decreaseVelocity(balls: ArrayList<Ball>): Boolean

5) SettingsManager.java

SettingsManager class changes the music, changes the music volume and changes the color of the cue chalk, cue stick and table.

SettingsManager
+musicLevel: int +sounVolum: int -tableColor: int -cueStickColor: int -cueChalkColor: int
+ getCueChalkColor(): int + setCueChalkColor(cueChalkColor: int) + getCueStickColor(): int + setCueStickColor(cueStickColor: int) + getMusicLevel(): int + setMusicLevel(musicLevel: int) + getSoundVolume(): int + setSoundVolume(soundVolume: int) + getTableColor(): int + setTableColor(tableColor: int)

6) SoundManager.java

SoundManager class plays the selected music and also stops it when it is required. Also handles with the unsupported audio file problem.

SoundManager
- musicClip: Clip - musicTimer: Timer - MusicAin: AudioInputStream -mf: File -sf: File - theSoundManager: SoundManager +filename: String
+getInstance():SoundManager + playMusic() + stopMusic(stop: boolean) + playCollision()

Models:

1) Ball.java

Ball class holds the ball informations. Sets the type of balls, draws balls, gets and sets the ball positions and velocities.

Ball
-x: int -y: int - WIDTH: int - HEIGHT: int -vx: double -vy: double -cx: double -cy: double -type: ImageIcon
+ setType(typeCode: int) + paint(g: Graphics) + setPoint(x: int, y:int) +getType():ImageIcon + setType(_type: ImageIcon) + getVx(): double + setVx(vx: double) + getVy():double + setVy(vy: double) + getX(): int + setX(x: int) + getY(): int

+ setY(y: int) + getCx(): double + setCx(cx: double) + getCy(): double + setCy(cy: double)

2) Configuration.java

Configuration class sets or gets the game mode. Game mode might be Player vs. Computer or Player vs. Player.

Configuration
-gameType: String -gameMode: String
+ getGameMode(): String + getGameType(): String + setGameMode(_gameMode: String) + setGameType(_gameType: String)

3) CueChalk.java

CueChalk keeps the color of the cue chalk.

CueChalk
- color: String
+ getColor(): String + setColor(color: String)

4) CueStick.java

CueStick class draws the cue stick and place its location. Also calculates the hitting angle to balls.

-color: String - directionAngle: String - power: String -x: int -y: int -WIDTH: int - HEIGHT:int -a: int -b: int + radian: double + g: Graphics2D - stickIcon: ImageIcon

+ paint(g: Graphics) +findRadian(mousex: double, mousey: double)

5) EightBallTable.java

EightBallTable defines the table of the 8th ball billiard game. Hole coordinates are sets in this class.

EightBallTable
-holeCoordinates: Point[]
+setHoleCoordinates(holeCoordinates: Point[]) + getHoleCoordinates():Point[]

6) Player.java

Player class holds the player's ball, hold player's turn and player's score.

Player
- isHisTurn: Boolean -score: int - selectedBallType: String - username: String
+ getSelectedBallType(): String + setSelectedBallType(selectedBallType: String) +getScore(): int + setScore(score: int) + getIsHisTurn(): Boolean + setIsHisTurn(isHisTurn: boolean) + getUsername(): String + setUsername(username: String)

7) SavaGame.java

SaveGame class saves the game when player wishes to save. Ball positions and the score of the user saved the user save the game.

SavaGame
- balls: ArrayList<Ball> - highScore : int
+getBalls():ArrayList<Ball> +setBalls(balls: ArrayList<Ball>) + getHighScore(): int + setHighScore(highScore: int)

8) ScoreBoard.java

ScoreBoard shows the scores of the player while game is continuing.

ScoreBoard
- p1Score: int -p2Score: int
+ getP2Score(): int + setP2Score(p2Score: int) + getP1Score():int + setP1Score(p1Score: int)

9) Table.java

Table class holds the color of the table.

Table
- color: Color - cornerPoints: Point[]
+ getCornerPoints(): Point[] + setCornerPoints(cornerPoints: Point[]) + getColor(): Color + setColor(_color: Color):

Views:

1. Application Container: This class is responsible for transitions between panels.

Application Container
-currentPanel : JPanel -ApplicationContainer: ApplicationContainer
+getInstance() : ApplicationContainer +changePanel(newPanel : JPanel) +main(args : String[])

2. 2.CreditsView: This panel where our game's credit information is given

CreditsView
-sM : SoundManager
-jButton1ActionPerformed (evt: ActionEvent) +main(args : String[])

3. 3.GameModeView: This is for selecting Player vs. Computer or Player vs. Player

GameModeView
-jButton1ActionPerformed (evt: ActionEvent) -jButton2ActionPerformed (evt: ActionEvent) -jButton3ActionPerformed (evt: ActionEvent) +main(args : String[])

4. 4.GameTypeView: This panel is for displaying available game modes and enabling user to select one of them.

GameTypeView
-jButton1ActionPerformed (evt: ActionEvent) -jButton2ActionPerformed (evt: ActionEvent) -jButton3ActionPerformed (evt: ActionEvent) +main(args : String[])

5. 5.GameView: This panel is to show playing screen.

GameView
-stick: CueStick -b1 : Ball -b2 : Ball -b3 : Ball -b4 : Ball -b5 : Ball -b6 : Ball -b7 : Ball -b8 : Ball -b9 : Ball -b10 : Ball -b11 : Ball -b12 : Ball -b13 : Ball -b14: Ball -b15 : Ball -balls: ArrayList<Ball> -player: Player -timer: Timer -pEngine: PhysicsEngine -sM: SoundManager -speedBar: JprogressBar -isHitting: boolean -count: int -power: int
+paint (g : Graphics)

6. TimerListener:

TimerListener
+ actionPerformed(e:ActionEvent)

7. MouseActionsListener:

MouseActionsListener
+ mouseDragged (e: MouseEvent) +mouseMoved(e: MouseEvent) +mouseClicked (e: MouseEvent) +mouseEntered (e: MouseEvent) +mouseExited (e: MouseEvent) +mouseReleased (e: MouseEvent) +mousePressed (e: MouseEvent)

8. KListener:

KListener
+ keyTyped (e: KeyEvent) + keyPressed (e: KeyEvent) + keyReleased (e: KeyEvent)

9. HelpView:

HelpView
- jButton1ActionPerformed (evt: ActionEvent) +main(args : String[])

10. HighScoresView:

HighScoresView
- jButton1ActionPerformed (evt: ActionEvent) +main(args : String[])

11. LoadGameView:

LoadGameView
- jButton1ActionPerformed (evt: ActionEvent) +main(args : String[])

12. MainMenuPanel:

MainMenuPanel
- jButton1ActionPerformed (evt: ActionEvent) - jButton2ActionPerformed (evt: ActionEvent) - jButton3ActionPerformed (evt: ActionEvent) - jButton4ActionPerformed (evt: ActionEvent) - jButton5ActionPerformed (evt: ActionEvent) - jButton6ActionPerformed (evt: ActionEvent) - jButton7ActionPerformed (evt: ActionEvent)

13. OptionsView:

OptionsView
- jButton1ActionPerformed (evt: ActionEvent) - jButton2ActionPerformed (evt: ActionEvent)

6.3. SPECIFYING CONTRACTS USING OCL

1.context GameView inv: Balls -> Size() <= 16 && Balls -> Size() >= 0

2.context Ball inv: x >= 0 && y >= 0

3.context GameView inv: beginningX = 60 && beginningY = 133

4.context ScoreBoard inv: score => 0

5.context Ball inv: vx >= 0 && vy >= 0

6.context Ball inv: cx >= 0 && cy >= 0

7.context Ball inv: WIDTH = 20

8.context Ball inv: HEIGHT= 20

9.context Ball::setPoint(x:int, y:int)

post : self.x = x && self.y = y

10. context Ball::getType() : ImageIcon

post : result = type

11. context Ball::setType(type : ImageIcon)

post : self.type type

12. context Ball::getVx() : int

post : result = vx

13. context Ball::setVx(vx : int)

post : self.vx = vx

14. context Ball::getVy() : int

post : result = vy

15. context Ball::setVy(vy : int)

post : self.vy = vy

16. context Ball inv: x >= 0 && y >= 0

17. context Ball::getX() : int

post : result = x

18. context Ball::setX(x : int)

post : self.x = x

19. context Ball::getY() : int

post : result = y

20. context Ball::setY(y : int)

post : self.y = y

21. context Ball::getCx() : int

post : result = cx

22. context Ball::setCx(cx : int)

post : self.cx = cx

23. context Ball::getCy() : int

post : result = cy

24. context Ball::setCy(cy : int)

post : self.cy = cy

25. context SettingsManager:: getCueChalkColor () : int

post : result = cueChalkColor

26. context SettingsManager:: setCueChalkColor (cueChalkColor: int)

post : self. cueChalkColor= cueChalkColor

27. context SettingsManager:: getMusicLevel () : int

post : result = cueChalkColor

28. context SettingsManager:: setMusicLevel (cueChalkColor: int)

post : self. musicLevel= musicLevel

29. context SavedGame:: setBalls (balls : ArrayList<Ball>) :

post : self.balls = balls

30. context SavedGame:: getHighScore () : int

post : result = highScore

7. CONCLUSION

In this report, we tried to explain what is our project, what it does, which functionalities it has, what we aim to do, how we will do it (design) and further details about the implementation of the project. Game features and given requirements were our main helper before we started to consider use cases and their scenarios. The analysis report was a useful report such that it helped us to think about the design process. The scenarios and use cases in this report were led us to pass to the design process easier and helped us about the contents of the design report. In design report, we used what we discussed in analysis and revised some parts accordingly. In design report, we saw more concrete image of the project. By considering further about project in design report, we saw some of our mistakes and corrected them. Generally, the main aim of us was to make people understand our project and see what and how we did and make a better image of our game in people's minds. To achieve that aim, we used case tools such as Visual Paradigm and using a modeling language, Unified Modeling Language (UML). Since UML can be considered as a unique programming language, it really helped us to communicate with other people easily. We used Visual Paradigm to create our UML diagrams such as Use Case diagrams, Sequence diagrams, Domain Analysis Diagram, State Diagrams etc. Finally, we converted our design into codes and implemented our project.

8. IMPLEMENTATION

Controller Package

DatabaseHandler.java

```
package controller;


import java.sql.*;
public class DatabaseHandler{
            private static Connection con;
            private static Statement stmt;
            private static DatabaseHandler theDatabase;
   private DatabaseHandler(){
      try {
         //to initialize sun.jdbc.odbc.JdbcOdbcDriver class
         //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         String filename = "";
         filename = "jdbc:mysql://localhost/billiard";


         con = DriverManager.getConnection( filename ,"root","admin");


                  stmt = con.createStatement();
      }catch (Exception e) {
         System.out.println("Error: " + e);
      }
   }
   public static DatabaseHandler getInstance(){
            if(theDatabase == null){
```

```
                theDatabase = new DatabaseHandler();
        }
        return theDatabase;
    }
    //executes query
    public ResultSet getResultSet(String query)throws SQLException{
        return stmt.executeQuery(query);
    }
    public int getResultSetUpdate(String query)throws SQLException{
        return stmt.executeUpdate(query);
    }
    //method for inserting a new user's username and password
    public void addUser(String username, String password, String name, String surname,
        String score)
//    throws SQLException
    {
        try{
                    String sql = "INSERT INTO User (name, surname,score,username,
password)"+
                            " values(?,?,?,?,?)";
                    PreparedStatement pstmt = con.prepareStatement(sql);
                    pstmt.setString(1,name);
                    pstmt.setString(2,surname);
        pstmt.setString(3,score);
        pstmt.setString(4,username);
        pstmt.setString(5,password);
                    pstmt.executeUpdate();
        }catch(Exception ex){System.out.println(ex); }
    }
```

```
   public static void close()throws SQLException{
            con.close();
   }
```

ImageHandler.java

```
public class ImageHandler {
   public static Image getBackgroundImage(){
      return new ImageIcon("images/background.jpg").getImage();
   }
   public static JLabel getBackgroundLabel(){
       JLabel label = new JLabel(new ImageIcon("images/background.jpg")){
          @Override
          public void paint(Graphics g){
              super.paint(g);
             //((Graphics2D)g).setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
0.4f));
             g.drawImage(ImageHandler.getBackgroundImage(),-150,-150,670,500,null);
             //((Graphics2D)g).setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
1.0f));
          }
      };
      label.setSize(670,510);
      return label;
   }
   public static Image getTableImage(){
      return new ImageIcon("images/table" +Table.type+ ".jpg").getImage();
   }
```

```
   public static JLabel getTableLabel(){
       JLabel label = new JLabel(){
           @Override
           public void paint(Graphics g){
               super.paint(g);
               ((Graphics2D)g).setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
0.4f));
               g.drawImage(ImageHandler.getTableImage(),0,0,400,300,null);
               ((Graphics2D)g).setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
1.0f));
           }
       };
       label.setSize(400,300);
       return label;
   }
}
```

LoginController.java

```
public class LoginController {
   public static boolean checkUserPassword(String userName, String password){
       DatabaseHandler db = DatabaseHandler.getInstance();
       String query = "SELECT * FROM Player WHERE username = '" + userName + "'";
       try{
           ResultSet rs = db.getResultSet(query);
           rs.next();
           String pass = rs.getString("password");
           if(password.equals(pass)){
               return true;
           }
```

```
        else{
            return false;
        }
      }catch(Exception ex){ex.printStackTrace();}

      return false;
   }
}
```

PhysicsEngine.java

```
public class PhysicsEngine {

   private static PhysicsEngine thePhysicsEngine;
   public static int first = 0;
   public static int second = 0;
   public static boolean isScore = false;

   public PhysicsEngine(){
   }

   public static PhysicsEngine getInstance(){
                if(thePhysicsEngine == null){
                        thePhysicsEngine = new PhysicsEngine();
                }
                return thePhysicsEngine;
         }
```

```
public boolean moveBall(Ball b){
    detectCollision(b);

    //checking x coordinates
    if(!(b.getX()+ b.getVx() > 344)){
       b.setCx(b.getCx()+ b.getVx());
       b.setX((int)b.getCx());
    }
    if(b.getX()+ b.getVx() > 344){
       b.setVx(-b.getVx());
    }
    if(b.getX()+ b.getVx() < 26){
       b.setVx(-b.getVx());
       b.setCx(b.getCx()+ b.getVx());
       b.setX((int)b.getCx());
    }

    //checking y coordinates
    if(!(b.getY()+ b.getVy() > 241)){
       b.setCy(b.getCy()+ b.getVy());
       b.setY((int)b.getCy());
    }
    if(b.getY()+ b.getVy() > 241){
       b.setVy(-b.getVy());
    }
    if(b.getY()+ b.getVy() < 32){
       b.setVy(-b.getVy());
       b.setCy(b.getCy()+ b.getVy());
```

```
            b.setY((int)b.getCy());
        }

        return true;
    }

    public boolean detectCollision(Ball b){
        Point apoint, point;
        apoint = new Point(b.getX()+10,b.getY()+10);
        try{
            for(Ball b1:GameView.balls){
                if(!b1.equals(b)){
                    point = new Point(b1.getX()+10,b1.getY()+10);
                    if (apoint.distance(point) <= 20){
                        handleCollision(b, b1);
                        return true;
                    }
                }
            }
        }
        catch(Exception ex){
            determineScore(b);
        }
        return false;
    }

    public boolean handleCollision(Ball b1, Ball b2){
        double x,y,d2;
```

```
y = (b2.getY() - b1.getY());

x = (b2.getX() - b1.getX());


// distance squared

d2 = x * x + y * y;

if(d2 == 0)

   return false;

double kii, kji, kij, kjj;

// i is b  j is b1

kji = (x * b1.getVx() + y * b1.getVy()) / d2; // k of j due to i

kii = (x * b1.getVy() - y * b1.getVx()) / d2; // k of i due to i

kij = (x * b2.getVx() + y * b2.getVy()) / d2; // k of i due to j

kjj = (x * b2.getVy() - y * b2.getVx()) / d2; // k of j due to j


// set velocity of i

double x1,y1,x2,y2;


x1 = kij * x - kii * y;

y1 = kij * y + kii * x;

x2 = kji * x - kjj * y;

y2 = kji * y + kjj * x;


b1.setVy(y1);

b1.setVx(x1);


// set velocity of j

b2.setVy(y2);

b2.setVx(x2);
```

```
    //System.exit(0);
    return true;
}

public boolean decreaseVelocity(ArrayList<Ball> balls){
    for(Ball b: balls){
        if(!(b.getVx() == 0 && b.getVy() == 0)){
            if(b.getVy() == 0){
                b.setVx(b.getVx()*4/5);
            }
            else if(b.getVx() == 0){
                b.setVy(b.getVy()*4/5);
            }
            else{
                double tan = b.getVy()/b.getVx();
                b.setVx(b.getVx()*4/5);
                b.setVy(b.getVx()*tan);
            }
            double velocity = Math.sqrt(Math.pow(b.getVx(), 2)+Math.pow(b.getVy(), 2));
            if(velocity < 0.3){
                b.setVx(0);
                b.setVy(0);
            }
        }
        //GameView.timer.stop();

    }
```

```
    return true;
}

public void determineScore(Ball b){
    Point apoint, point;
    apoint = new Point(b.getX()+10,b.getY()+10);
    for(Ball b1:GameTView.balls){
        if(!b1.equals(b)){
            point = new Point(b1.getX()+10,b1.getY()+10);
            if (apoint.distance(point) <= 20 &&
                    (b1 == GameTView.balls.get(1) &&
                        b == GameTView.balls.get(0))){
                first = 1;
                first += second;
                if(first == 2)
                    isScore = true;
                handleCollision(b, b1);
            }
            if (apoint.distance(point) <= 20 &&
                    (b1 == GameTView.balls.get(2) &&
                        b == GameTView.balls.get(0))){
                second = 1;
                second += first;
                if(second == 2)
                    isScore = true;
                handleCollision(b, b1);
            }
```

```
        }
      }
   }
}
```

RulesEngine.java

```
public class RulesEngine {
   private static RulesEngine theRulesEngine;

   public RulesEngine(){
   }

   public static RulesEngine getInstance(){
                if(theRulesEngine == null){
                        theRulesEngine = new RulesEngine();
                }
                return theRulesEngine;
          }

   public boolean isPocketed(Ball b){
      if (b.getX() <= 29 && b.getY() <= 32)
         return true;
      else if (b.getX() >= 341 && b.getY() <= 32)
         return true;
      else if (b.getX() <= 210 && b.getX() >= 188 && b.getY() <= 32)
         return true;
      if (b.getX() <= 29 && b.getY()+5 >= 241)
         return true;
```

```
        else if (b.getX() >= 342 && b.getY() >= 241)
            return true;
        else if (b.getX() <= 210 && b.getX() >= 188 && b.getY() >= 243)
            return true;

        return false;
    }
}
```

ScoreController.java

```
public class ScoreController {
    public static ArrayList<Integer> scores;
    public static ArrayList<String> names;

    public ScoreController(){}

    public static void highScores(){
        DatabaseHandler db = DatabaseHandler.getInstance();
        String query = "SELECT name,score FROM Player";
        scores = new ArrayList();
        names = new ArrayList();
        try{
            ResultSet rs = db.getResultSet(query);
            while(rs.next()){
                String name = rs.getString("name");
                int score = Integer.parseInt(rs.getString("score"));
                scores.add(score);
                names.add(name);
```

```
        }
      }catch(Exception ex){ex.printStackTrace();}
   }

   public static ArrayList<String> getNames() {
      return names;
   }

   public static void setNames(ArrayList<String> names) {
      ScoreController.names = names;
   }

   public static ArrayList<Integer> getScores() {
      return scores;
   }

   public static void setScores(ArrayList<Integer> scores) {
      ScoreController.scores = scores;
   }
}
```

SettingsManager.java

```
public class SettingsManager{
        public static int musicLevel=60;
        public static int soundVolume=60;
   public static int tableColor = 0;
   public static int cueStickColor = 0;
   public static int cueChalkColor = 0;
```

```
public static int getCueChalkColor() {
   return cueChalkColor;
}

public static void setCueChalkColor(int cueChalkColor) {
   SettingsManager.cueChalkColor = cueChalkColor;
}

public static int getCueStickColor() {
   return cueStickColor;
}

public static void setCueStickColor(int cueStickColor) {
   SettingsManager.cueStickColor = cueStickColor;
}

public static int getMusicLevel() {
   return musicLevel;
}

public static void setMusicLevel(int musicLevel) {
   SettingsManager.musicLevel = musicLevel;
}

public static int getSoundVolume() {
   return soundVolume;
}
```

```
    public static void setSoundVolume(int soundVolume) {
        SettingsManager.soundVolume = soundVolume;
    }

    public static int getTableColor() {
        return tableColor;
    }

    public static void setTableColor(int tableColor) {
        SettingsManager.tableColor = tableColor;
    }
}
```

SoundManager.java

```
public class SoundManager{
            private static Clip musicClip;
            private static Timer musicTimer;
            private static AudioInputStream musicAin;
            private static File mf,sf;
    private static SoundManager theSoundManager;
    public static String fileName = "Pat Benatar-Hit Me With Your Best Shot";

    public SoundManager(){
    }

    public static SoundManager getInstance(){
                    if(theSoundManager == null){
```

```
                    theSoundManager = new SoundManager();
                }
                return theSoundManager;
    }

    /**
     *Plays the "in-game" music when player is in in game.
     */
    public void playMusic()throws IOException,
     UnsupportedAudioFileException,
     LineUnavailableException{
  if(musicClip == null ){
     mf = new File("sounds/" + fileName +".wav");
     musicAin = AudioSystem.getAudioInputStream(mf);
          try {
             DataLine.Info info = new DataLine.Info(Clip.class,musicAin.getFormat( ));
             musicClip = (Clip) AudioSystem.getLine(info);
             musicClip.open(musicAin);
          }catch(Exception ex){}
          finally {
             musicAin.close( );
          }
       }

       if(!musicClip.isActive()){
          FloatControl fc = (FloatControl)musicClip.getControl(FloatControl.Type.MASTER_GAIN);
          double max = fc.getMaximum(), min = fc.getMinimum();
          double width = (max-min)/100.0;
```

```
        fc.setValue( (float)(min+width*SettingsManager.getMusicLevel()));
        musicClip.start();

        musicTimer = new Timer(100,new ActionListener(){
    public void actionPerformed(ActionEvent e){
       try{
            musicClip.start();
       }catch(Exception ex){ex.printStackTrace();}
       if(!musicClip.isActive()){
          try{
             musicClip.close();
             musicAin.close();
             musicAin = AudioSystem.getAudioInputStream(mf);
             musicClip.open(musicAin);
             musicClip.start();
          }
          catch(Exception ex){}
       }
    }
        } );
   musicTimer.start();
     }
   }


   public void stopMusic(boolean stop){
  if(musicClip != null ){
```

```
    if(stop){
        musicTimer.stop();
        musicClip.stop();
    }else{
        musicTimer.start();
    }
}
  }

  public void playCollision()throws IOException,
  UnsupportedAudioFileException,
  LineUnavailableException{
File f = new File("sounds/collision.wav");
AudioInputStream ain = AudioSystem.getAudioInputStream(f);
Clip clip=null;
try {
    DataLine.Info info = new DataLine.Info(Clip.class,ain.getFormat( ));
    clip = (Clip) AudioSystem.getLine(info);
    clip.open(ain);
}
catch(Exception ex){ex.printStackTrace();}
finally {
    ain.close( );
}
FloatControl fc = (FloatControl)clip.getControl( FloatControl.Type.MASTER_GAIN);

double max = fc.getMaximum(), min = fc.getMinimum();
double width = (max-min)/100.0;
```

```
        fc.setValue( (float)(min+width*SettingsManager.getSoundVolume()));
        clip.start();
        }
}
```

Model Package

Ball.java

```
public class Ball implements Serializable{


   private int x,y;
         private final static int WIDTH = 20, HEIGHT = 20;
         private double vx, vy, cx, cy;
         private ImageIcon type=null;
   private boolean isPocketed;
   private boolean isHit;


   public Ball(int x, int y, int num){
      isPocketed = false;
      isHit = false;
                  setPoint(x,y);
      cx = x;
      cy = y;
      vx = 0;
      vy = 0;
      setType(num);
         }


   public void setType(int typeCode){
                        type = new ImageIcon("images/"+ typeCode +".png");
         }


   public void paint(Graphics g){
```

```
                g.drawImage(type.getImage(),x+3,y+3,WIDTH,HEIGHT,null);
        }

public void setPoint(int x, int y){
                this.x = x;
   this.y = y;
}

public ImageIcon getType() {
   return type;
}

public void setType(ImageIcon _type) {
   type = _type;
}

public double getVx() {
   return vx;
}

public void setVx(double vx) {
   this.vx = vx;
}

public double getVy() {
   return vy;
}
```

```
public void setVy(double vy) {
   this.vy = vy;
}

public int getX() {
   return x;
}

public void setX(int x) {
   this.x = x;
}

public int getY() {
   return y;
}

public void setY(int y) {
   this.y = y;
}

public double getCx() {
   return cx;
}

public void setCx(double cx) {
   this.cx = cx;
}
```

```
    public double getCy() {
        return cy;
    }

    public void setCy(double cy) {
        this.cy = cy;
    }

    public boolean getIsPocketed() {
        return isPocketed;
    }

    public void setIsPocketed(boolean isPocketed) {
        this.isPocketed = isPocketed;
    }

    public boolean isIsHit() {
        return isHit;
    }

    public void setIsHit(boolean isHit) {
        this.isHit = isHit;
    }
}
```

Configuration.java

```
public class Configuration {
   private static String gameType;
   private static String gameMode;

   public Configuration(){}

   public static String getGameMode() {
      return gameMode;
   }

   public static String getGameType() {
      return gameType;
   }

   public static void setGameMode(String _gameMode) {
      gameMode = _gameMode;
   }

   public static void setGameType(String _gameType) {
      gameType = _gameType;
   }
}
```

CueChalk.java

```
public class CueChalk {


   protected String color;


   public CueChalk() {}


   /**
    * Get the value of color
    *
    * @return the value of color
    */
   public String getColor() {
      return color;
   }


   /**
    * Set the value of color
    *
    * @param color new value of color
    */
   public void setColor(String color) {
      this.color = color;
   }
}
```

CueStick.java

```
public class CueStick {


   public String stickType;
   int X,Y,WIDTH, HEIGHT,a,b;
   public double radian;
   Graphics2D g;
   public ImageIcon stickIcon;


   public CueStick(int x, int y){
      if(SettingsManager.getCueStickColor() == 0)
         stickType = "black";
      else if(SettingsManager.getCueStickColor() == 1)
         stickType = "red";
      else if(SettingsManager.getCueStickColor() == 2)
         stickType = "white";
      stickType = "stick";
      stickIcon = new ImageIcon("images/" + stickType + ".jpg");
      X = x;
      Y = y;
      radian = 0;
                  WIDTH = stickIcon.getIconWidth();
                  HEIGHT = stickIcon.getIconHeight();
      try{
         a = GameView.balls.get(0).getX()-WIDTH+2;
         b = GameView.balls.get(0).getY()+HEIGHT/2+1;
      }
      catch(Exception ex){
```

```
        a = GameTView.balls.get(0).getX()-WIDTH+2;
        b = GameTView.balls.get(0).getY()+HEIGHT/2+1;
    }
}

public String getStickType() {
    return stickType;
}

public void setStickType(String stickType) {
    this.stickType = stickType;
}

public void paint(Graphics g){
                this.g = (Graphics2D)g;
    this.g.rotate(-radian,X, Y);
                this.g.drawImage(stickIcon.getImage(),a,b,null );
        }

public void findRadian(double mousex,double mousey){
    double tan, deltaHeight, angle;
    boolean isTurned = false;
    if((X+WIDTH-mousex)>0){
        deltaHeight = X+WIDTH-mousex;
        try{
            a = GameView.balls.get(0).getX()-WIDTH;
            b = GameView.balls.get(0).getY()+HEIGHT/2+1;
        }
```

```
    catch(Exception ex){
        a = GameTView.balls.get(0).getX()-WIDTH;
        b = GameTView.balls.get(0).getY()+HEIGHT/2+1;
    }
}
else{
    try{
        a = GameView.balls.get(0).getX()-WIDTH;
        b = GameView.balls.get(0).getY()+HEIGHT/2+4;
    }
    catch(Exception ex){
        a = GameTView.balls.get(0).getX()-WIDTH;
        b = GameTView.balls.get(0).getY()+HEIGHT/2+4;
    }
    isTurned = true;
    deltaHeight = X+WIDTH-mousex;
}
                if (mousey == Y)
                        tan = 0;
                else{
                        tan = (mousey-Y-HEIGHT)/deltaHeight;
}
if(!isTurned)
    angle = Math.atan(tan);
else
    angle = Math.atan(tan) + Math.PI;
try{
    X = GameView.balls.get(0).getX()+10;
```

```
            Y = GameView.balls.get(0).getY()+10;
        }
        catch(Exception ex){
            X = GameTView.balls.get(0).getX()+10;
            Y = GameTView.balls.get(0).getY()+10;
        }

        radian = angle;

            }
}
```

EightBallTable.java

```
public class EightBallTable extends Table{
    protected Point[] holeCoordinates;

    public EightBallTable() {}




    public void setHoleCoordinates(Point[] holeCoordinates) {
        this.holeCoordinates = holeCoordinates;
    }

    public Point[] getHoleCoordinates() {
        return holeCoordinates;
    }
}
```

Player.java

```
public class Player {

   protected boolean isHisTurn;
   protected int score;
   protected String selectedBallType;
   protected static String username;
   public boolean pocket;



   public Player() {
      isHisTurn = false;
      pocket = false;
   }

   /**
    * Get the value of selectedBallType
    *
    * @return the value of selectedBallType
    */
   public String getSelectedBallType() {
      return selectedBallType;
   }

   /**
    * Set the value of selectedBallType
    *
```

```
 * @param selectedBallType new value of selectedBallType
 */
public void setSelectedBallType(String selectedBallType) {
   this.selectedBallType = selectedBallType;
}



/**
 * Get the value of score
 *
 * @return the value of score
 */
public int getScore() {
   return score;
}

/**
 * Set the value of score
 *
 * @param score new value of score
 */
public void setScore(int score) {
   this.score = score;
}

/**
 * Get the value of isHisTurn
 *
```

```
     * @return the value of isHisTurn
     */
    public boolean getIsHisTurn() {
        return isHisTurn;
    }

    /**
     * Set the value of isHisTurn
     *
     * @param isHisTurn new value of isHisTurn
     */
    public void setIsHisTurn(boolean isHisTurn) {
        this.isHisTurn = isHisTurn;
    }


    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

SavedGame.java

```
public class SavedGame implements Serializable {
    private ArrayList<Ball> balls;
```

```
    private int highScore;

    public SavedGame(int highScore) {
        setBalls(view.GameView.balls);
        this.highScore = highScore;
    }

    public ArrayList<Ball> getBalls() {
        return balls;
    }

    public void setBalls(ArrayList<Ball> balls) {
        this.balls = balls;
    }

    public int getHighScore() {
        return highScore;
    }

    public void setHighScore(int highScore) {
        this.highScore = highScore;
    }
}
```

ScoreBoard.java

```
public class ScoreBoard {

   protected int p1Score;
   protected int p2Score;

   /**
    * Get the value of p2Score
    *
    * @return the value of p2Score
    */
   public int getP2Score() {
      return p2Score;
   }

   /**
    * Set the value of p2Score
    *
    * @param p2Score new value of p2Score
    */
   public void setP2Score(int p2Score) {
      this.p2Score = p2Score;
   }


   /**
    * Get the value of p1Score
    *
```

```
     * @return the value of p1Score
     */
    public int getP1Score() {
        return p1Score;
    }

    /**
     * Set the value of p1Score
     *
     * @param p1Score new value of p1Score
     */
    public void setP1Score(int p1Score) {
        this.p1Score = p1Score;
    }
    public ScoreBoard(){}
}
```

Table.java

```
public class Table {

    protected Point[] cornerPoints;
    public static String type = "g";

    public Table() {
    }

    public Table(Point[] cornerPoints) {
```

```
        this.cornerPoints = cornerPoints;
    }

    public Point[] getCornerPoints() {
        return cornerPoints;
    }

    public void setCornerPoints(Point[] cornerPoints) {
        this.cornerPoints = cornerPoints;
    }
}
```

View Package

ApplicationContainer.java

```
public class ApplicationContainer extends JFrame{
	private static JPanel currentPanel = new JPanel();
	private static ApplicationContainer theMainFrame;
	private ApplicationContainer(){
		theMainFrame = this;


		setResizable(false);


		currentPanel = new JPanel();
		add(currentPanel);
		pack();
		setVisible(true);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
	private ApplicationContainer(JPanel initial){
		this();
		changePanel(initial);
	}
	public static ApplicationContainer getInstance(){
		if( theMainFrame == null ){
			theMainFrame = new ApplicationContainer();
		}
		return theMainFrame;
	}
	public static void changePanel(JPanel newPanel){
```

```
                if( theMainFrame == null ){
                        theMainFrame = new ApplicationContainer();
                }
                theMainFrame.remove(currentPanel);
                currentPanel  = newPanel;
                theMainFrame.add(currentPanel);
                theMainFrame.pack();
        }
        public static void main(String[] args){
                new ApplicationContainer(new LoginView() );
        }
}
```

CreditsView.java

```
public class CreditsView extends javax.swing.JPanel {

   private SoundManager sM;
   /** Creates new form CreditsView */
   public CreditsView() {
      initComponents();
      sM = SoundManager.getInstance();
      try{
         sM.fileName = "Crazy Frog-We are the champions";
         sM.playMusic();
      }
      catch(Exception ex){ex.printStackTrace();}
   }
```

```
public static void main(String[] args){
   ApplicationContainer.changePanel(new CreditsView() );
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

   jButton1 = new javax.swing.JButton();
   jLabel1 = new javax.swing.JLabel();
   jLabel2 = new javax.swing.JLabel();
   jLabel3 = new javax.swing.JLabel();
   jLabel4 = new javax.swing.JLabel();

   jButton1.setText("Back");
   jButton1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         jButton1ActionPerformed(evt);
      }
   });

   jLabel1.setFont(new java.awt.Font("Snap ITC", 0, 12)); // NOI18N
```

```
        jLabel1.setText("Hüseyin GÜLER");

        jLabel2.setFont(new java.awt.Font("Snap ITC", 0, 14)); // NOI18N
        jLabel2.setText("Enes TAYLAN");

        jLabel3.setFont(new java.awt.Font("Snap ITC", 0, 14)); // NOI18N
        jLabel3.setText("Alperen ERASLAN");

        jLabel4.setFont(new java.awt.Font("Snap ITC", 0, 14)); // NOI18N
        jLabel4.setText("Ömer DURMUS");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(359, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addGap(35, 35, 35)
                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGap(304, 304, 304))
            .addGroup(layout.createSequentialGroup()
                .addGap(100, 100, 100)
                .addComponent(jLabel2)
                .addContainerGap(210, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
```

```
            .addContainerGap(169, Short.MAX_VALUE)
            .addComponent(jLabel3)
            .addGap(107, 107, 107))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(270, Short.MAX_VALUE)
            .addComponent(jLabel4)
            .addGap(34, 34, 34))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addGap(47, 47, 47)
            .addComponent(jLabel1)
            .addGap(32, 32, 32)
            .addComponent(jLabel2)
            .addGap(40, 40, 40)
            .addComponent(jLabel3)
            .addGap(41, 41, 41)
            .addComponent(jLabel4)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 50,
Short.MAX_VALUE)
            .addComponent(jButton1)
            .addContainerGap())
    );
  }// </editor-fold>

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    ApplicationContainer.changePanel(new MainMenuPanel());
    sM.stopMusic(true);
```

```
    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    // End of variables declaration

}
```

GameModeView.java

```
public class GameModeView extends javax.swing.JPanel {

    /** Creates new form GameModeView */
    public GameModeView() {
        initComponents();
        add(controller.ImageHandler.getBackgroundLabel());
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();

    jButton1.setText("8 - Ball Game");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jButton2.setText("3 - Ball Game");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    jButton3.setText("Back");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });
```

```
    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
    this.setLayout(layout);
    layout.setHorizontalGroup(
       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(layout.createSequentialGroup()
          .addGap(142, 142, 142)
          .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
             .addComponent(jButton2)
             .addComponent(jButton1))
          .addContainerGap(163, Short.MAX_VALUE))
       .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
          .addContainerGap(335, Short.MAX_VALUE)
          .addComponent(jButton3)
          .addContainerGap())
    );
    layout.setVerticalGroup(
       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(layout.createSequentialGroup()
          .addGap(71, 71, 71)
          .addComponent(jButton1)
          .addGap(66, 66, 66)
          .addComponent(jButton2)
          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 83,
Short.MAX_VALUE)
          .addComponent(jButton3)
          .addContainerGap())
    );
  }// </editor-fold>
```

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        model.Configuration.setGameType("8ball");
        ApplicationContainer.changePanel(new GameView() );
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        model.Configuration.setGameType("3ball");
        ApplicationContainer.changePanel(new GameTView() );
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        ApplicationContainer.changePanel(new MainMenuPanel());
    }



    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    // End of variables declaration

}
```

GameTView.java

```
public class GameTView extends javax.swing.JPanel {

    private static GameTView theGameTView;
    private CueStick stick;
```

```
private Ball b0,b1,b2;
private final int beginningX = 60, beginningY = 133;
public static ArrayList<Ball> balls;
public static javax.swing.Timer timer;
private PhysicsEngine pEngine;
private SoundManager sM;
public JProgressBar speedBar;
private boolean isHitting = false;
private int count;
private int power;
public static int lNo;
private static Player p1;
private static Player p2;
private boolean turn;
private int p1Score;
private int p2Score;



/** Creates new form GameTView */
public GameTView() {
   initComponents();
   setLayout(null);
                setPreferredSize(new Dimension(400,400));
   initComponents();
   setBackground(Color.LIGHT_GRAY);
   Table.type = "3";
   add(controller.ImageHandler.getTableLabel());
   turn = true;
```

```
lNo = 1;

p1 = new Player();

p2 = new Player();

p1Score = 0;

p2Score = 0;

p1.setIsHisTurn(turn);

pEngine = PhysicsEngine.getInstance();


sM = SoundManager.getInstance();

try{

   System.out.println(SoundManager.fileName);

   sM.playMusic();

}

catch(Exception ex){ex.printStackTrace();}


setFocusable(true);

                addKeyListener(new KListener() );


addMouseMotionListener(new MouseActionsListener() );

                addMouseListener(new MouseActionsListener() );



speedBar = new JProgressBar(1,10);

add(speedBar);

speedBar.setVisible(false);



b0 = new Ball(beginningX,beginningY,0);
```

```
    //b0 = new Ball(190,240,0);
    b0.setVx(0);
    b0.setVy(0);
    b1 = new Ball(beginningX + 240,beginningY-50,1);
    b2 = new Ball(beginningX + 240,beginningY+50,3);

    balls = new ArrayList();

    balls.add(b0);
    balls.add(b1);
    balls.add(b2);

    stick = new CueStick(beginningX,beginningY+10);

    timer = new javax.swing.Timer(20,new TimerListener() );
    timer.start();
}

public static GameTView getInstance(){
                if(theGameTView == null){
                        theGameTView = new GameTView();
                }
                theGameTView.setFocusable(true);
                return theGameTView;
        }

@Override
public void paint(Graphics g){
```

```
    super.paint(g);
    for(Ball b: GameTView.balls){
        b.paint(g);
    }
    super.paintComponents(g);
    stick.paint(g);
        }

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jProgressBar1 = new javax.swing.JProgressBar();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();

    jButton1.setText("Save Game");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Main Menu");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jLabel2.setFont(new java.awt.Font("Ravie", 3, 12)); // NOI18N
jLabel2.setText("Power:");

jLabel3.setFont(new java.awt.Font("Ravie", 3, 10)); // NOI18N
jLabel3.setText("Player-1 Score:");

jLabel4.setFont(new java.awt.Font("Ravie", 3, 10)); // NOI18N
jLabel4.setText("Player-2 Score:");

jLabel5.setFont(new java.awt.Font("Ravie", 3, 10)); // NOI18N
jLabel5.setText("0");

jLabel6.setFont(new java.awt.Font("Ravie", 3, 10)); // NOI18N
jLabel6.setText("0");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
```

```
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(11, 11, 11)
                        .addComponent(jLabel2)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                            .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                        .addGap(35, 35, 35)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel4)
                                .addGap(18, 18, 18)
                                .addComponent(jLabel6))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel3)
                                .addGap(18, 18, 18)
                                .addComponent(jLabel5)))))
```

```
            .addContainerGap(126, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(209, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2))
                .addGap(9, 9, 9)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jLabel3)
                    .addComponent(jLabel5))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jLabel4)
                    .addComponent(jLabel6))
                .addContainerGap())
        );
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        ApplicationContainer.changePanel(new MainMenuPanel());
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    try {
        SaveGame();
    } catch (Exception e) {
        e.printStackTrace();
    }
  }

  public void SaveGame() throws IOException
  {
    ObjectOutputStream objstream = new ObjectOutputStream(new
FileOutputStream("savedGames/billiard"+
        lNo+".ser"));
    lNo++;
    objstream.writeObject(new SavedGame(5));
    objstream.close();

  }

  private class TimerListener implements ActionListener{
                public void actionPerformed(ActionEvent e){
      count++;
      if(count < 30){
        if(isHitting){
          if(power != 100)
            power = power+2;
          else
            power = 0;
        }
        else
```

```
        power = 0;
    }
    if(count == 30){
       pEngine.decreaseVelocity(GameTView.balls);
       count = 0;
    }
    try{
       for(Ball b: GameTView.balls){
          pEngine.moveBall(b);
          if(turn){
             if(PhysicsEngine.isScore){
                p1Score++;
                jLabel5.setText(p1Score+"");
             }
             else if(b.getVx() == 0 && b.getVy()==0){
                PhysicsEngine.first = 0;
                PhysicsEngine.second = 0;
             }
          }
          else{
             if(PhysicsEngine.isScore){
                p2Score++;
                jLabel6.setText(p2Score+"");
             }
             else if(b.getVx() == 0 && b.getVy()==0){
                PhysicsEngine.first = 0;
                PhysicsEngine.second = 0;
             }
```

```
            }
        }
    }
    catch(Exception ex){}
    if(isHitting)
        jProgressBar1.setValue(power);
                    repaint();
                }
        }

/**
         *Takes the mouse actions and responses to them
         */
        private class MouseActionsListener
                    implements MouseMotionListener, MouseListener{
                /**
                 *sets the turrets angle
                 */
                public void mouseDragged(MouseEvent e){
    stick.stickIcon = new ImageIcon("images/" + stick.stickType + ".jpg");
    jProgressBar1.setVisible(true);
    stick.findRadian(e.getX(),e.getY());
    isHitting = true;
                    repaint();
                }
                /**
                 *sets the mouse angle and starts the game timer and take focus to the
                 * GamePanel at the beginning of the game.
```

```
                */
                public void mouseMoved(MouseEvent e){}
                public void mouseClicked(MouseEvent e){}
                public void mouseEntered(MouseEvent e){}
                public void mouseExited(MouseEvent e){}
                /**
                 *launches a ball from turret if no active ball exists
                 * changes the color of turret and next ball
                 */
                public void mouseReleased(MouseEvent e){
    if(isHitting){
       double sx, sy;
       sx = Math.sin(stick.radian + Math.PI/2) * jProgressBar1.getPercentComplete()*10;
       sy = Math.cos(stick.radian + Math.PI/2) * jProgressBar1.getPercentComplete()*10;
       b0.setVx(sx);
       b0.setVy(sy);
       stick.stickIcon = new ImageIcon("images/a.jpg");
       repaint();
    }
    isHitting = false;
    jProgressBar1.setVisible(false);
                }
                public void mousePressed(MouseEvent e){}
      }

/**
        *Takes the keyboard actions and responds them
        */
```

```
private class KListener implements KeyListener{
                public void keyTyped(KeyEvent e){}
                /**
                 *Does the action related to given keyboard input.
                 */
                public void keyPressed(KeyEvent e){
      System.out.println(e.getKeyText(e.getKeyCode()));
      if (e.getKeyText(e.getKeyCode()).equalsIgnoreCase("P")){
         timer.stop();

        JOptionPane.showMessageDialog(ApplicationContainer.getInstance(),
                                                "PAUSE!\nClick here to begin!",
                                                "PAUSE",1);
                                timer.start();
      }
                }
                public void keyReleased(KeyEvent e){}
        }

public static void main(String[] args){
   ApplicationContainer.changePanel(new GameTView());
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
```

```
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JProgressBar jProgressBar1;
    // End of variables declaration

}
```

GameTypeView.java

```
public class GameTypeView extends javax.swing.JPanel {

    /** Creates new form GameTypeView */
    public GameTypeView() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
```

```
jButton1.setText("Player vs. Computer");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Player vs. Player");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Back");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(127, 127, 127)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
```

```
                .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

                .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

            .addContainerGap(142, Short.MAX_VALUE))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

            .addContainerGap(335, Short.MAX_VALUE)

            .addComponent(jButton3)

            .addContainerGap())

    );

    layout.setVerticalGroup(

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup()

            .addGap(63, 63, 63)

            .addComponent(jButton1)

            .addGap(62, 62, 62)

            .addComponent(jButton2)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 95,
Short.MAX_VALUE)

            .addComponent(jButton3)

            .addContainerGap())

    );

  }// </editor-fold>//GEN-END:initComponents


  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed

    model.Configuration.setGameType("pp");

    ApplicationContainer.changePanel(new GameView());

  }//GEN-LAST:event_jButton1ActionPerformed
```

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton2ActionPerformed

        model.Configuration.setGameType("pc");

        ApplicationContainer.changePanel(new GameView());

    }//GEN-LAST:event_jButton2ActionPerformed


    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton3ActionPerformed

        ApplicationContainer.changePanel(new GameModeView());

    }//GEN-LAST:event_jButton3ActionPerformed




    // Variables declaration - do not modify//GEN-BEGIN:variables

    private javax.swing.JButton jButton1;

    private javax.swing.JButton jButton2;

    private javax.swing.JButton jButton3;

    // End of variables declaration//GEN-END:variables


}
```

GameView.java

```
public class GameView extends javax.swing.JPanel {


    /** Creates new form GameView */

            private static GameView theGameView;

    private CueStick stick;

    private Ball b0,b1,b2,b3,b4,b5,b6;

    private Ball b7,b8,b9,b10,b11,b12,b13,b14,b15;
```

```
private final int beginningX = 60, beginningY = 133;
private int INDX = 150,INPX = 150, INY = 340;
public static ArrayList<Ball> balls;
public static ArrayList<Ball> pocketedDBalls;
public static ArrayList<Ball> pocketedPBalls;
public static javax.swing.Timer timer;
private PhysicsEngine pEngine;
private RulesEngine rEngine;
private SoundManager sM;
public JProgressBar speedBar;
private boolean isHitting = false;
private int count;
private int power;
public static int lNo;
private static Player p1;
private static Player p2;
private boolean turn;

public GameView() {
   setLayout(null);
                setPreferredSize(new Dimension(400,400));
   initComponents();
   setBackground(Color.LIGHT_GRAY);
   add(controller.ImageHandler.getTableLabel());
   turn = true;
   lNo = 1;
   p1 = new Player();
   p2 = new Player();
```

```
p1.setIsHisTurn(turn);
pEngine = PhysicsEngine.getInstance();
rEngine = RulesEngine.getInstance();

sM = SoundManager.getInstance();
try{
   System.out.println(SoundManager.fileName);
   sM.playMusic();
}
catch(Exception ex){ex.printStackTrace();}

setFocusable(true);
                addKeyListener(new KListener() );

addMouseMotionListener(new MouseActionsListener() );
                addMouseListener(new MouseActionsListener() );


speedBar = new JProgressBar(1,10);
add(speedBar);
speedBar.setVisible(false);


b0 = new Ball(beginningX,beginningY,0);
//b0 = new Ball(190,240,0);
b0.setVx(0);
b0.setVy(0);
b1 = new Ball(beginningX + 140,beginningY,1);
```

```
b2 = new Ball(beginningX + 160,beginningY-10,2);
b3 = new Ball(beginningX + 160,beginningY+10,3);
b4 = new Ball(beginningX + 180,beginningY-20,4);
b5 = new Ball(beginningX + 180,beginningY,5);
b6 = new Ball(beginningX + 180,beginningY+20,6);
b7 = new Ball(beginningX + 200,beginningY-30,7);
b8 = new Ball(beginningX + 200,beginningY-10,8);
b9 = new Ball(beginningX + 200,beginningY+10,9);
b10 = new Ball(beginningX + 200,beginningY+30,10);
b11 = new Ball(beginningX + 220,beginningY-40,11);
b12 = new Ball(beginningX + 220,beginningY-20,12);
b13 = new Ball(beginningX + 220,beginningY,13);
b14 = new Ball(beginningX + 220,beginningY+20,14);
b15 = new Ball(beginningX + 220,beginningY+40,15);
balls = new ArrayList();
pocketedDBalls = new ArrayList();
pocketedPBalls = new ArrayList();
balls.add(b0);
balls.add(b1);
balls.add(b2);
balls.add(b3);
balls.add(b4);
balls.add(b5);
balls.add(b6);
balls.add(b7);
balls.add(b8);
balls.add(b9);
balls.add(b10);
```

```
    balls.add(b11);
    balls.add(b12);
    balls.add(b13);
    balls.add(b14);
    balls.add(b15);

    stick = new CueStick(beginningX,beginningY+10);

    timer = new javax.swing.Timer(20,new TimerListener() );
    timer.start();
}

public void setBalls(ArrayList<Ball> balls) {
    GameView.balls = balls;
}

public static GameView getInstance(){
            if(theGameView == null){
                theGameView = new GameView();
            }
            theGameView.setFocusable(true);
            return theGameView;
        }

@Override
public void paint(Graphics g){
    super.paint(g);
    for(Ball b: balls){
```

```
        if(!b.getIsPocketed())
            b.paint(g);
    }
    for(Ball b: pocketedDBalls)
        b.paint(g);
    for(Ball b: pocketedPBalls)
        b.paint(g);
    super.paintComponents(g);
    stick.paint(g);
          }

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jProgressBar1 = new javax.swing.JProgressBar();
    jButton1 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jButton2 = new javax.swing.JButton();

    setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
```

```
jProgressBar1.setFocusable(false);

jButton1.setText("Save Game");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Ravie", 3, 12)); // NOI18N
jLabel1.setText("Power:");

jLabel2.setFont(new java.awt.Font("Ravie", 2, 12)); // NOI18N
jLabel2.setText("Turn of Player-1");

jButton2.setText("Main Menu");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 137,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                    .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(212, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jLabel2)
                .addComponent(jLabel1)
                .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton1)
            .addGap(11, 11, 11)
```

```
            .addComponent(jButton2))
        );
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            SaveGame();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        ApplicationContainer.changePanel(new MainMenuPanel());
    }
    public void SaveGame() throws IOException
    {
        ObjectOutputStream objstream = new ObjectOutputStream(new
FileOutputStream("savedGames/billiard"+
            lNo+".ser"));
        lNo++;
        objstream.writeObject(new SavedGame(5));
        objstream.close();

    }

    private class TimerListener implements ActionListener{
                    public void actionPerformed(ActionEvent e){
            count++;
```

```
boolean change = false, moving = false;
if(count < 30){
   if(isHitting){
      if(power != 100)
         power = power+2;
      else
         power = 0;
   }
   else
      power = 0;
}
if(count == 30){
   pEngine.decreaseVelocity(GameView.balls);
   count = 0;

}
try{
   for(Ball b: GameView.balls){
      if(!b.getIsPocketed()){
         pEngine.moveBall(b);
         if(rEngine.isPocketed(b)){
            b.setIsPocketed(true);
            change = true;
            if(balls.indexOf(b) == 0){
               System.out.println("White ball is pocketed");
               turn = !turn;
               b.setX(beginningX);
               b.setY(beginningY);
```

```
    p1.setIsHisTurn(turn);
    p2.setIsHisTurn(!turn);
}
else if(balls.indexOf(b) < 8 && balls.indexOf(b) != 0){
    if(turn)
        p1.pocket = true;
    else
        p2.pocket = true;
    System.out.println("DUZ");
    b.setX(INDX); b.setY(INY);
    INDX +=20;
    pocketedDBalls.add(b);
}
else if(balls.indexOf(b) > 8){
    System.out.println("PARCALI");
    if(turn)
        p1.pocket = true;
    else
        p2.pocket = true;
    b.setX(INPX); b.setY(INY +25);
    INPX +=20;
    pocketedPBalls.add(b);
}

else if(balls.indexOf(b) == 8){
    timer.stop();
    JOptionPane.showMessageDialog(ApplicationContainer.getInstance(),
        "Player 1 Lost!!!",
```

```
                    "Game Over!!!",1);
                ApplicationContainer.changePanel(new CreditsView());
              }
            }
          }
        }
      }
      catch(Exception ex){}
      if(isHitting)
        jProgressBar1.setValue(power);
                        repaint();
              }
       }
/**
        *Takes the mouse actions and responses to them
        */
       private class MouseActionsListener
                        implements MouseMotionListener, MouseListener{
              /**
               *sets the turrets angle
               */
              public void mouseDragged(MouseEvent e){
      stick.stickIcon = new ImageIcon("images/" + stick.stickType + ".jpg");
      jProgressBar1.setVisible(true);
      stick.findRadian(e.getX(),e.getY());
      isHitting = true;
                        repaint();
              }
```

```
            /**
             *sets the mouse angle and starts the game timer and take focus to the
             * GamePanel at the beginning of the game.
             */
            public void mouseMoved(MouseEvent e){}
            public void mouseClicked(MouseEvent e){}
            public void mouseEntered(MouseEvent e){}
            public void mouseExited(MouseEvent e){}
            /**
             *launches a ball from turret if no active ball exists
             * changes the color of turret and next ball
             */
            public void mouseReleased(MouseEvent e){
if(isHitting){
   double sx, sy;
   sx = Math.sin(stick.radian + Math.PI/2) * jProgressBar1.getPercentComplete()*10;
   sy = Math.cos(stick.radian + Math.PI/2) * jProgressBar1.getPercentComplete()*10;
   b0.setVx(sx);
   b0.setVy(sy);
   stick.stickIcon = new ImageIcon("images/a.jpg");
   repaint();
}
if(p1.pocket){
   jLabel2.setText("Turn of Player-1");
   p2.pocket = false;
}
else{
   jLabel2.setText("Turn of Player-2");
```

```
            p1.pocket = false;
        }
        isHitting = false;
        jProgressBar1.setVisible(false);
                    }
                    public void mousePressed(MouseEvent e){}
         }

/**
          *Takes the keyboard actions and responds them
          */
private class KListener implements KeyListener{
                    public void keyTyped(KeyEvent e){}
                    /**
                     *Does the action related to given keyboard input.
                     */
                    public void keyPressed(KeyEvent e){
        System.out.println(e.getKeyText(e.getKeyCode()));
        if (e.getKeyText(e.getKeyCode()).equalsIgnoreCase("P")){
            timer.stop();

          JOptionPane.showMessageDialog(ApplicationContainer.getInstance(),
                                                    "PAUSE!\nClick here to begin!",
                                                    "PAUSE",1);
                                    timer.start();
        }
                    }
                    public void keyReleased(KeyEvent e){}
         }
```

```
    public static void main(String[] args){
        ApplicationContainer.changePanel(new GameView());
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JProgressBar jProgressBar1;
    // End of variables declaration

}
```

HelpView.java

```
public class HelpView extends javax.swing.JPanel {

    /** Creates new form HelpView */
    public HelpView() {
        initComponents();
    }

    public static void main(String[] args){
        ApplicationContainer.changePanel(new HelpView() );
    }

    /** This method is called from within the constructor to
     * initialize the form.
```

```
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        textArea1 = new java.awt.TextArea();
        textArea2 = new java.awt.TextArea();
        textArea3 = new java.awt.TextArea();

        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        textArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        textArea1.setEditable(false);
        textArea1.setText("8 BALL GAME\n\nOBJECT OF THE GAME\n\n      In Eight ball, the object of
the game is to pocket\n the Eight ball after having legally pocketed either all \nthe Stripes (#9 to #15)
or the Solids (#1 to #7). Once all\nthe player's object balls are pocketed, he/she may attempt\n to
sink the 8 ball. To win, the player must first call \nwhich pocket they plan to sink the Eight ball into. If
the\n Eight ball is shot into the wrong pocket or a foul (see \nbelow) occurs, the player loses.
Otherwise, the player's \nturn is over. \n\n\nOPENING BREAK\n\n      One person is chosen (by a
predetermined method, \ne.g., coin flip, win or loss of previous game, lag) to shoot\n first and break
the object ball rack apart. If the shooter\n who breaks fails to make a legal break (usually defined
as\n at least four balls hitting cushions or an object ball being\n pocketed), then the opponent can
demand a re-rack and become\n the breaker, or elect to play from the current position of\n the
```

balls.\n\n If the breaker pockets a ball, it is still that \nplayer's turn and the table is considered \"open\" (meaning the\n breaker can still make any object ball to determine if \nhe/she will only shoot solids or stripes throughout the \ngame). If the breaker fails to make another ball after the \nbreak, the table is still considered \"open\" until someone \nlegally pockets a ball.\n\n According to World Standardized Rules, if the 8-ball\n is pocketed on the break, breaker may ask for a re-rack or \nhave the 8-ball spotted and continue shooting. If the breaker\n scratches while pocketing the 8-ball on the break, the \nincoming player has the option of a re-rack or having the \n8-ball spotted and begin shooting with ball in hand behind \nthe head string. \n\n\nWINNING AN EIGHT BALL GAME\n\n* The player has legally pocketed the 8 ball, after all \nhis/her object balls have been pocketed\n* The opposing player illegally pockets the 8 ball (e.g.\n before clearing all of his/her object balls, in the same\n shot as the last such object ball, or into a pocket other\n than the one that was called)\n* The opposing player scratches the cue ball into a pocket,\n or knocks it off of the table, when the eight ball is \npocketed. A scratch or foul is not loss of game if the \n8-ball is not pocketed or jumped from the table\n* The opposing player commits any foul on the shot that \npocketed the 8 ball (in non-tournament situations, non-\ncue-ball fouls may be excused from this requirement)\n* The opposing player knocks the 8 ball off of the table \n\n\nRULES FOR PLAYING EIGHT BALL\n\n American-style eight-ball rules are played around\n the world by professionals, and in many amateur leagues.\n The rules for eight-ball may be the most contested of any\n billiard game. There are several competing sets of \n\"official\" rules. The non-profit World Pool-Billiard \nAssociation (WPA), with national affiliates such as the \nBilliard Congress of America (BCA), promulgates the World\n Standardized Rules for amateur and professional play.\n\n The for-profit International Pool Tour has also \nestablished an international set of rules for professional\n and semi-professional play, used in major tournaments \nbroadcast on television. Meanwhile, many amateur leagues, \nsuch as the American Poolplayers Association (APA) / \nCanadian Poolplayers Association (CPA), and the Valley \nNational Eight-ball Association (VNEA) / VNEA Europe, use\n their own rulesets as their standard (most of them at \nleast loosely based on the WPA/BCA version), while millions\n of individuals play informally using colloquial rules \nwhich vary not only from area to area but even from venue \nto venue. \n\nFOULS\n\n* The shooter fails to strike one of his/her own object \nballs (or the 8 ball, if all of said object balls are \nalready pocketed) with the cue ball, before other balls \n(if any) are contacted by the cue ball. This includes \n\"split\" shots, where the cue ball strikes one of the \nshooter's and one of the opponent's object ball \nsimultaneously.\n* No ball comes into contact with a cushion or is pocketed,\n after legal cue ball contact with the (first) object ball\n (or 8 ball).\n* The cue ball is pocketed.\n* The shooter does not have at least one foot on the floor\n (this requirement may be waived if the shooter is \ndisabled in a relevant way, or the venue has not provided\n a mechanical bridge)\n* The cue ball is shot before all balls have come to a \ncomplete stop from the previous shot.\n* The cue ball is struck more than once during a shot.\n* The cue ball is jumped entirely or partially over an \nobstructing ball with an illegal jump shot that scoops \nunder the cue ball.\n* The cue ball is clearly pushed, with the cue tip \nremaining in contact with it more than momentarily.\n* The shooter touches the cue ball with something other\n than the tip of the cue.\n* The shooter touches any other ball (with body, clothing\n or equipment), other than as necessary to move the cue \nball when the player has ball-in-hand * The shooter knocks\n a ball off of the table.\n* The shooter has shot out-of-turn.\n* On the break shot, no balls are pocketed and fewer than \nfour balls reach the cushions (in which case the incoming \nplayer can demand a re-rack and take the break or force \nthe original breaker to re-break, or may take ball-in-hand\n and shoot the balls as they lie)\n");

jTabbedPane1.addTab("8 Ball Game Rules", textArea1);

```
textArea2.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

textArea2.setEditable(false);

textArea2.setText("OPENING BREAK\n\n        In three ball, players' turn order is decided at random at \nthe beginning of the game or match, as in other several-player pool\n games. The cue ball is placed anywhere behind the head string (\"in\n the kitchen\") and a typical hard break (as in nine-ball or eight-\nball) is performed. The break is the first stroke of a player's game,\n and thus counts toward his or her score. Any balls pocketed on the\n break are considered to be legally pocketed and the player now only\n has to sink the remaining balls.\n\n        Very good players can sink all three object balls on the \nbreak with surprising frequency, resulting in the perfect (but still\n tieable) score of one point, especially if the balls are triangle-\nracked; this feat is achieved using an adaptation of the instant-win\n break technique from eight-ball and nine-ball; the straight rack \nwas introduced to make this more difficult, as it does not provide \nthe contact point and angles that the well-known technique requires. \n\n\nFOULS IN THREE BALL\n\n        Every shot costs one point, and a foul of any kind costs\n the player an additional one-point penalty. Fouls consist of: \npocketing the cue ball; knocking the cue ball off the table; a \ndouble hit on the cue ball with the cue stick (including illegal \n\"scoop-under\" jump shots); push shots; and (possibly, depending on\n how serious the game is) accidentally (or otherwise) moving a ball\n with a hand, the butt of the cue, etc. A shot in which the player \npocketed one or more object balls but also fouled incur a one point\n penalty - a foul always results in a penalty of 1 point. Thus, a \nbreak shot that sank all three object balls plus the cue ball is a \nscore of two (one for the actual shot, plus one for the foul), \nunless the \"instant loss\" rule (see below) is in effect.\n        It is not a foul to make a weak shot that does not pocket\n a ball or contact a cushion, since, again, these mistakes are \neffectively self-punishing, by costing the player a stroke.\n");

jTabbedPane1.addTab("3 Ball Game Rules", textArea2);


textArea3.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

textArea3.setEditable(false);

textArea3.setText("PLAYING THE GAME BILLIARD\nSTRIKING THE BALL\nPlayer should adjust the direction of the cuestick to where s/he would like to strike");

jTabbedPane1.addTab("How To Play", textArea3);


javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(

   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

   .addGroup(layout.createSequentialGroup()
```

```
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(jButton1, javax.swing.GroupLayout.Alignment.TRAILING)
               .addComponent(jTabbedPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 380,
Short.MAX_VALUE))
            .addContainerGap())
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 237,
Short.MAX_VALUE)
            .addGap(18, 18, 18)
            .addComponent(jButton1)
            .addContainerGap())
      );
   }// </editor-fold>

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      ApplicationContainer.changePanel(new MainMenuPanel());
   }

   // Variables declaration - do not modify
   private javax.swing.JButton jButton1;
   private javax.swing.JTabbedPane jTabbedPane1;
   private java.awt.TextArea textArea1;
   private java.awt.TextArea textArea2;
```

```
    private java.awt.TextArea textArea3;
    // End of variables declaration
}
```

HighScoresView.java

```
public class HighScoresView extends javax.swing.JPanel {

    /** Creates new form HighScoresView */
    public HighScoresView() {
        initComponents();
        add(controller.ImageHandler.getBackgroundLabel());
        ScoreController.highScores();
        ArrayList<Integer> sc = ScoreController.getScores();
        ArrayList<String> na = ScoreController.getNames();
        String str = "";
        for(int i =0; i<sc.size() ; i++){
            str += na.get(i) + "\t\t" + sc.get(i) +"\n";
        }
        jTextArea1.setText(str);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel1 = new javax.swing.JLabel();

        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        jLabel1.setFont(new java.awt.Font("Ravie", 3, 24)); // NOI18N
        jLabel1.setText("High Scores");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(81, 81, 81)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
            .addContainerGap(153, Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()
            .addGap(90, 90, 90)
            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(112, 112, 112))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(285, Short.MAX_VALUE)
            .addComponent(jButton1)
            .addGap(60, 60, 60))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(24, 24, 24)
            .addComponent(jLabel1)
            .addGap(29, 29, 29)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 161,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 29,
Short.MAX_VALUE)
            .addComponent(jButton1)
            .addContainerGap())
    );
  }// </editor-fold>

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    ApplicationContainer.changePanel(new MainMenuPanel());
  }
```

```
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration

}
```

LoadGameView.java

```
public class LoadGameView extends javax.swing.JPanel {

    /** Creates new form LoadGameView */
    public LoadGameView() {
        initComponents();
        String s = "billiard1";
            jComboBox1.addItem(s);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jComboBox1 = new javax.swing.JComboBox();

jButton1.setText("Back");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Load");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(335, Short.MAX_VALUE)
        .addComponent(jButton1)
        .addContainerGap())
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
```

```
            .addGap(147, 147, 147)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 99, Short.MAX_VALUE)
                .addComponent(jComboBox1, javax.swing.GroupLayout.Alignment.LEADING, 0, 99,
Short.MAX_VALUE))
            .addGap(154, 154, 154))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(91, 91, 91)
                .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(72, 72, 72)
                .addComponent(jButton2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 60,
Short.MAX_VALUE)
                .addComponent(jButton1)
                .addContainerGap())
        );
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        ApplicationContainer.changePanel(new MainMenuPanel());
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        SavedGame loadedGame;
        try {
```

```
            loadedGame = (SavedGame)loadObject("savedGames/billiard1.ser");
            GameView view = new GameView();
            view.setBalls(loadedGame.getBalls());
            view.repaint();
            ApplicationContainer.changePanel(view);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static Object loadObject(String filename) throws ClassNotFoundException, IOException {
        ObjectInputStream objstream = new ObjectInputStream(new FileInputStream(filename));
        Object object = objstream.readObject();
        objstream.close();
        return object;
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JComboBox jComboBox1;
    // End of variables declaration

}
```

LoginView.java

```
public class LoginView extends javax.swing.JPanel {

    /** Creates new form LoginView */
    public LoginView() {
        initComponents();
        JLabel l = controller.ImageHandler.getBackgroundLabel();
        l.setBounds(-40, -40, l.getWidth(), l.getHeight());
        add(l);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        userField = new javax.swing.JTextField();
        passField = new javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();

        jLabel1.setText("Username:");
```

```
        jLabel2.setText("Password:");

        passField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                passFieldActionPerformed(evt);
            }
        });

        jButton1.setText("Login");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(109, 109, 109)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jButton1)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
```

```
                .addComponent(passField)
                .addComponent(userField, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(151, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(100, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jLabel1)
          .addComponent(userField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(32, 32, 32)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jLabel2)
          .addComponent(passField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(39, 39, 39)
        .addComponent(jButton1)
        .addGap(66, 66, 66))
    );
  }// </editor-fold>

  private void passFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try{
```

```
            boolean success = LoginController.checkUserPassword(
                    userField.getText(), passField.getText());
            if(success){
               ApplicationContainer.changePanel(new MainMenuPanel());
            }else{
               JOptionPane.showMessageDialog(null, "username or password is incorrect",
                     "Error", JOptionPane.ERROR_MESSAGE);
            }
         }catch(Exception ex){
            JOptionPane.showMessageDialog(null, "database error",
                     "Error", JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace();
         }



      }



      // Variables declaration - do not modify
      private javax.swing.JButton jButton1;
      private javax.swing.JLabel jLabel1;
      private javax.swing.JLabel jLabel2;
      private javax.swing.JPasswordField passField;
      private javax.swing.JTextField userField;
      // End of variables declaration

}
```

MainMenuPanel.java

```
public class MainMenuPanel extends javax.swing.JPanel {

   /** Creates new form MainMenuPanel */
   public MainMenuPanel() {
      initComponents();
      add(controller.ImageHandler.getBackgroundLabel());
   }

   /** This method is called from within the constructor to
    * initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is
    * always regenerated by the Form Editor.
    */
   public static void main(String[] args){
      ApplicationContainer.changePanel(new MainMenuPanel() );
   }

   @SuppressWarnings("unchecked")
   // <editor-fold defaultstate="collapsed" desc="Generated Code">
   private void initComponents() {

      jButton1 = new javax.swing.JButton();
      jButton2 = new javax.swing.JButton();
      jButton3 = new javax.swing.JButton();
      jButton4 = new javax.swing.JButton();
      jButton5 = new javax.swing.JButton();
      jButton6 = new javax.swing.JButton();
```

```
jLabel1 = new javax.swing.JLabel();
jButton7 = new javax.swing.JButton();

jButton1.setText("New Game");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Load Game");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Options");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton4.setText("Credits");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
```

```
    }
});

jButton5.setText("Help");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jButton6.setText("Exit");
jButton6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton6ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Lucida Calligraphy", 3, 36));
jLabel1.setForeground(new java.awt.Color(255, 204, 51));
jLabel1.setText("Billiard");

jButton7.setText("High Scores");
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});
```

```
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(112, Short.MAX_VALUE)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(103, 103, 103))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(165, 165, 165)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButton6, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
                    .addComponent(jButton5, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
                    .addComponent(jButton3, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
                    .addComponent(jButton4, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
                    .addComponent(jButton7, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
                    .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE))
                .addGap(146, 146, 146))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
            .addGap(6, 6, 6)
            .addComponent(jLabel1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton1)
            .addGap(18, 18, 18)
            .addComponent(jButton2)
            .addGap(18, 18, 18)
            .addComponent(jButton3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jButton4)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jButton7)
            .addGap(15, 15, 15)
            .addComponent(jButton5)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jButton6)
            .addContainerGap(24, Short.MAX_VALUE))
    );
}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    ApplicationContainer.changePanel(new GameModeView() );
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    ApplicationContainer.changePanel(new LoadGameView());
}
```

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
   ApplicationContainer.changePanel(new HelpView());
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
   ApplicationContainer.changePanel(new OptionsView());
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
   ApplicationContainer.changePanel(new CreditsView());
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
   System.exit(0);
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
   ApplicationContainer.changePanel(new HighScoresView());
}


// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
```

```
    private javax.swing.JButton jButton7;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration

}
```

OptionsView.java

```
public class OptionsView extends javax.swing.JPanel {

    /** Creates new form OptionsView */
    public OptionsView() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        buttonGroup2 = new javax.swing.ButtonGroup();
        buttonGroup3 = new javax.swing.ButtonGroup();
        buttonGroup4 = new javax.swing.ButtonGroup();
        buttonGroup5 = new javax.swing.ButtonGroup();
```

```
jSlider1 = new javax.swing.JSlider();
jButton1 = new javax.swing.JButton();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jComboBox1 = new javax.swing.JComboBox();
jLabel3 = new javax.swing.JLabel();
jRadioButton1 = new javax.swing.JRadioButton();
jRadioButton2 = new javax.swing.JRadioButton();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jComboBox2 = new javax.swing.JComboBox();
jLabel6 = new javax.swing.JLabel();
jComboBox3 = new javax.swing.JComboBox();
jButton2 = new javax.swing.JButton();
jSlider2 = new javax.swing.JSlider();
jLabel7 = new javax.swing.JLabel();
jComboBox4 = new javax.swing.JComboBox();

jButton1.setText("Back");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Lucida Calligraphy", 3, 24));
jLabel1.setForeground(new java.awt.Color(255, 204, 0));
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
        jLabel1.setText("Options");

        jLabel2.setText("Game Music: ");

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Pat Benatar-Hit Me With Your Best Shot", "R.H.C.P-Suck My Kiss", "No Doubt-Spiderwebs", "Paramore-Misery Business" }));

        jLabel3.setText("Sound Effects: ");

        jRadioButton1.setText("On");
        jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jRadioButton1ActionPerformed(evt);
            }
        });

        jRadioButton2.setText("Off");

        jLabel4.setText("Music Volume: ");

        jLabel5.setText("Table Color: ");

        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Green", "Blue" }));
        jComboBox2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox2ActionPerformed(evt);
            }
        });
```

```
        jLabel6.setText("Cue Stick Color: ");

        jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Black", "Red",
"White" }));

        jButton2.setText("Save");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jLabel7.setText("CueChalk Color:");

        jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Blue", "Red" }));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(148, 148, 148)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap(405, Short.MAX_VALUE)
                    .addComponent(jButton2)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))
                .addGroup(layout.createSequentialGroup()
                    .addGap(113, 113, 113)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(jLabel2)
                        .addComponent(jLabel4)
                        .addComponent(jLabel3)
                        .addComponent(jLabel5)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel7)
                            .addComponent(jLabel6)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(18, 18, 18)
                            .addComponent(jSlider1, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                .addComponent(jComboBox3,
javax.swing.GroupLayout.Alignment.LEADING, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jComboBox2,
javax.swing.GroupLayout.Alignment.LEADING, 0, 154, Short.MAX_VALUE)
                                .addComponent(jSlider2, javax.swing.GroupLayout.Alignment.LEADING, 0, 0,
Short.MAX_VALUE)
```

```
                        .addComponent(jComboBox1,
javax.swing.GroupLayout.Alignment.LEADING, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

                        .addComponent(jComboBox4,
javax.swing.GroupLayout.Alignment.LEADING, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))

                    .addGroup(layout.createSequentialGroup()

                        .addGap(35, 35, 35)

                        .addComponent(jRadioButton1)

                        .addGap(18, 18, 18)

                        .addComponent(jRadioButton2)))

                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 41,
Short.MAX_VALUE)))

                .addComponent(jButton1)))

            .addContainerGap())

        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(18, 18, 18)

                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                    .addGroup(layout.createSequentialGroup()

                        .addGap(38, 38, 38)

                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                            .addComponent(jSlider1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                            .addComponent(jSlider2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addGap(43, 43, 43)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel5))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel6)))
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(34, 34, 34)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel3)
                            .addComponent(jRadioButton1)
                            .addComponent(jRadioButton2)))
                    .addComponent(jLabel4))))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel7)
                .addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addGap(21, 21, 21)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jButton1)
               .addComponent(jButton2))
            .addGap(24, 24, 24))
      );
   }// </editor-fold>

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      ApplicationContainer.changePanel(new MainMenuPanel());
   }

   private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
   }

   private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
}

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      if(jComboBox2.getSelectedIndex() == 1)
         Table.type = "b";
      SettingsManager.setCueStickColor(jComboBox3.getSelectedIndex());
      SettingsManager.setCueChalkColor(jComboBox3.getSelectedIndex());

      if(jComboBox1.getSelectedIndex() == 1)
         SoundManager.fileName = "Pat Benatar-Hit Me With Your Best Shot";
```

```
    if(jComboBox1.getSelectedIndex() == 1)
        SoundManager.fileName = "R.H.C.P-Suck My Kiss";
    else if(jComboBox1.getSelectedIndex() == 2)
        SoundManager.fileName = "No Doubt-Spiderwebs";
    else if(jComboBox1.getSelectedIndex() == 3){
        SoundManager.fileName = "Paramore-Misery Business";
        System.out.println(SoundManager.fileName);
    }

    SettingsManager.musicLevel = jSlider2.getValue();

    ApplicationContainer.changePanel(new MainMenuPanel());

}

// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.ButtonGroup buttonGroup3;
private javax.swing.ButtonGroup buttonGroup4;
private javax.swing.ButtonGroup buttonGroup5;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
private javax.swing.JComboBox jComboBox3;
private javax.swing.JComboBox jComboBox4;
```

```
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JSlider jSlider1;
    private javax.swing.JSlider jSlider2;
    // End of variables declaration

}
```