Abstract Data Types (ADTs)

Data abstraction defines the domain and structure of the data, along with a collection of operations that access the data.

ADT is an abstract model that describes an interface between a client (user) and the data.

An ADT describes

- The data elements which make up the type,
- Data handling operations which can be performed on those elements.

1

ADT Format: A special format to describe an ADT.

```
ADT ADT_Name is
   Data
        Describe the structure of the data.
   Operations
        Constructor
              Initial values:Data used to initialize an object.
              Process:       Initialize the object.
        Operation1
              Input:         Data from the client.
              Preconditions: Necessary state of the system before
                             executing the operation.
              Process:       Actions performed on the data.
              Output:        Data returned to the client.
              Postconditions:State of the system after executing
                             the operation.
        Operation2
              . . .
        OperationN
              . . .
end ADT ADT_Name
```

2

Application: Dice Game

- A gaming program involves tossing a set of dice.
- In the design, the dice are described as an ADT.

- Dice ADT's data includes :
  - the number of dice that are tossed, and
  - a list that identifies the value of each die in the last toss.
- Dice ADT's operations include :
  - tossing the dice,
  - returning the sum of the dice on a toss, and
  - printing the value of each individual dice in the list.

Data: `diceTotal, diceList`
Operations: `Toss, Total, DisplayToss`

3

Ex: Dice ADT

```
ADT Dice is
   Data
        The number of dice (>=1) in each toss.
        An integer value containing the total of the dice on last toss.
   Operations
        Constructor
              Initial values:The number of dice to be tossed.
              Process:       Initialize the data values
        Toss
              Input:         None.
              Preconditions: None.
              Process:       Toss the dice and compute the dice total.
              Output:        None.
              Postconditions:The total contains the sum of the dice
                             on the toss, and the list identifies
                             the value of each die in the toss.
        DieTotal
              . . .
        DisplayToss
              . . .
end ADT Dice
```

4

Application: Pool Construction

- Building code requires that a concrete walkway must surround a swimming pool and that the entire area must be enclosed by a fence.
- The current fencing costs are $10 per meter and concrete costs are $5 per square meter.
- The application assumes that the width of the walkway is 1 meter, and the client specifies the radius of the circular pool.

5

Ex: Circle ADT

```
   ADT Circle is
      Data
           A non-negative real number specifying the radius of
           the circle.
      Operations
           Constructor
                 Initial values:The radius of the circle.
                 Process:       Assign an initial radius value.
           Area
                 Process:       Compute the area of the circle.
                 Output:        Return the area.
           Circumference
                 Process:       Compute the circumference of the
                                circle.
                 Output:        Return the circumference.
   end ADT Circle
```

6

C++ Classes (cont'd)

Message Passing: giving of an order to a receiving object by a client (sender) for it to perform some task.

State Change: When the receiving object performs some operation, it may update some of its internal data values and its state changes (i.e., new postconditions occur)

Class declaration: A C++ class is normally given by first declaring the class without defining the member functions. This is a concrete representation of an ADT.

Class definition: The actual definition of the class methods.

C++ Classes and ADTs

C++ provides the class data type to represent ADTs.

A class type consists of:
- Data members
- Methods: operations for handling data members.

Object: a variable of the class type.

Circle Class
private: radius
public: Constructor, Area, Circumference

C++ Classes (cont'd)

A class contains two separate parts:
- The public part: describes an interface for the client.
- The private part: contains the data and internal operations that assist in the implementation of the data abstraction.

Encapsulation: The class encapsulates information by bundling the data items and methods , and treating them as a single entity.

Information hiding: The class structure hides implementation details and carefully restricts outside access to both the data and operations.


```
#include <iostream.h>  // Application: Pool Construction

const float PI = 3.14152;
cost  float FencePrice = 10;
const float ConcretePrice = 5;

. . .

void main()
{
   float radius;
   float FenceCost, ConcreteCost;

   cout.setf(ios::fixed);
   cout.setf(ios::showpoint);
   cout.precision(2);

   cout << "Enter the radius of the pool:";
   cin >> radius;

   . . .
}
```

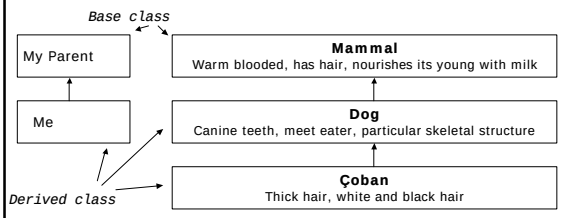

Designing New Objects by Code Reuse

Composition: more complex classes may contain that are themselves data objects

Point Class	Line Class	Rectangle Class

Designing New Objects by Code Reuse

Class Inheritance: a new class may be built as refinement of a previously defined class.



A derived class can use data and operations of the base class and can add new operations or overwrite some of its base class operations.

The SeqList ADT

```
ADT SeqList is
  Data
    A nonnegative integer
    specifying the size of the
    list.
    A list of data items.
  Operations
    Constructor
      In:None.
      Process: Set the size to 0.
    ListSize
      In:
      Pre:
      Process:
      Output:
      Post:
    ListEmpty
      In:
      Pre:
      Process:
      Output:
      Post:
```

```
ClearList
  In:
  Pre:
  Process:
  Output:
  Post:
Find
  In:
  Pre:
  Process:
  Output:
  Post:
Delete
  In:
  Pre:
  Process:
  Output:
  Post:
```


The SeqList ADT (cont'd)

```
  DeleteFront
    In:
    Pre:
    Process:
    Output:
    Post:
  GetData
    In:
    Pre:
    Process:
    Output:
    Post:
  end ADT SeqList
```

```
ADT OrderedList is
  Data <same as  SeqList ADT>
  Operations
   Constructor <executes the base
                class constructor>
   ListSize    <same as  SeqList ADT>
   ListEmpty   <same as  SeqList ADT>
   ClearList   <same as  SeqList ADT>
   Find        <same as  SeqList ADT>
   Delete      <same as  SeqList ADT>
   DeleteFront <same as  SeqList ADT>
   GetData     <same as  SeqList ADT>
   Insert
     In: Item to insert in the list
     Pre: -
     Process: Add new item in order
     Output: -
     Post: List has new item and its
           size increases by 1.
end ADT OrderedList
```


Base class

SeqList Class
private: <impl.details>
public: Constructor ListEmpty ClearList ListSize Find Insert Delete DeleteFront GetData

Derived class

OrderedList Class
private:
public: Constructor Insert

SeqList and OrderedList Class Specifications

```
class SeqList {
  private:
    DataType listitem[SIZE];
    int size;
  public:
    SeqList(void);

    int ListSize(void) const;
    int ListEmpty(void) const;
    int Find(DataType& item)const;
    DataType GetData(int pos)const;

  void Insert(const DataType&item);
  void Delete(const DataType&item);
  DataType DeleteFront(void);
  void ClearList(void);
};
```

```
class OrderedList:public SeqList
{
  public:
   OrderedList(void);
   void Insert(const DataType&
                              item);
};
```