Defne Bayık
Hande Gazey
CS-112 WEEK 2 (Feb. 18-22, 2008)

Topics Covered

1. Call by Value
2. Call by Reference
3. Passing to methods
4. Returning array by methods
5. Selection Sorting
6. Linear Search
7. Multidimensional array
8. Introduction to Classes and Objects

Call by Value:

Example:

```
        public static void swap(int n1, int n2)
{
                int temp=n1;
                n1=n2;
                n2=temp;
}
```

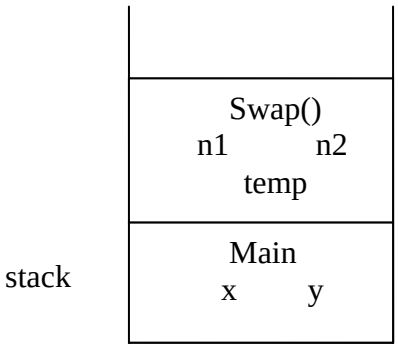
Method invocation

Main

```
int x=10, y=5;
swap(x,y);
display(x,y) ;
```

expected results : x=5 and y=10
actual results: x=10 and y=5

Memory



Call by value: The reason for that is call by value variables. Stay in method and are not sent to main n1, n2, temp are local variables for swap method

Call by Reference:

Example:

```
public static void m( int number, int
[] numbers){

        int x = 1;
        int [] y = new int [10];
```

```
        m(x,y);
        number = 1000;
        numbers[0] = 555;
```

Main

```
int x = 1;
int [] y = new int [10];
m(x,y);
System.out.println ( " x: " + x );
System.out.println ( " y[0]: " + y[0] );
```

expected result : x=1, y[0]=555
actual result : x=1, y[0]=555

Call by reference: pass the reference is only for arrays, we pass the reference and change array locations/parameters. When we are done with them, they will disappear from the main memory.

There are two types of memory:
Stack and Heap

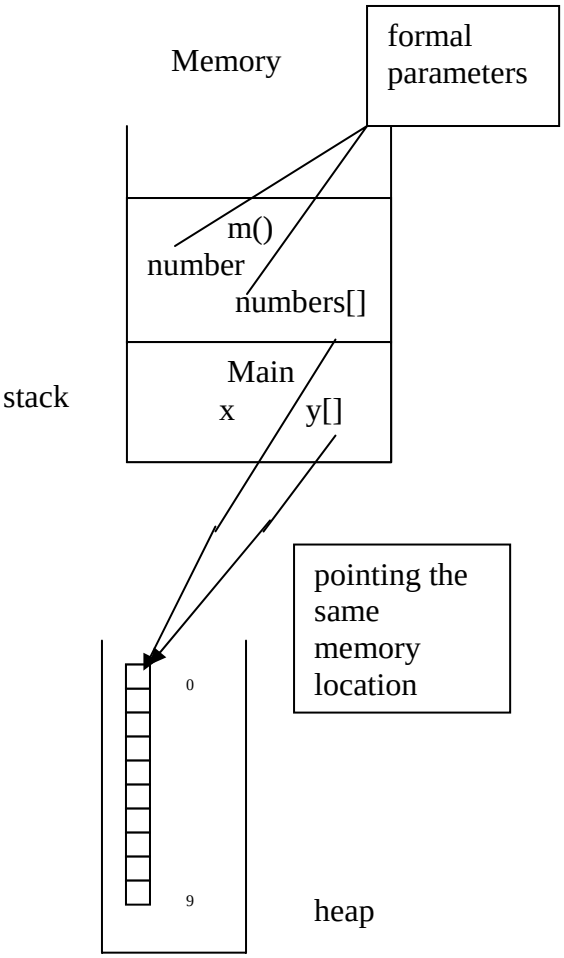
Stack concept: When we are done with the information and the methods, they are erased from main memory.

Heap concept: Special memory used for storage allocation for arrays.

Passing Arrays to Methods:

- Linear Search
- Algorithm Complexity

```
public static int linearSearch(int[] list, int key){
        // return the index location of "key" if
        // it does not exist in array return -1
        int r=-1;
        for (int i=0; i<list.length && r== -1; i++){
                if(key==list[i])
                        r=i;
        }
        return r;
}
```



passing an array to method syntax

Complexity of Algorithms:

number of
compression= 1/n + 2/n … + n/n

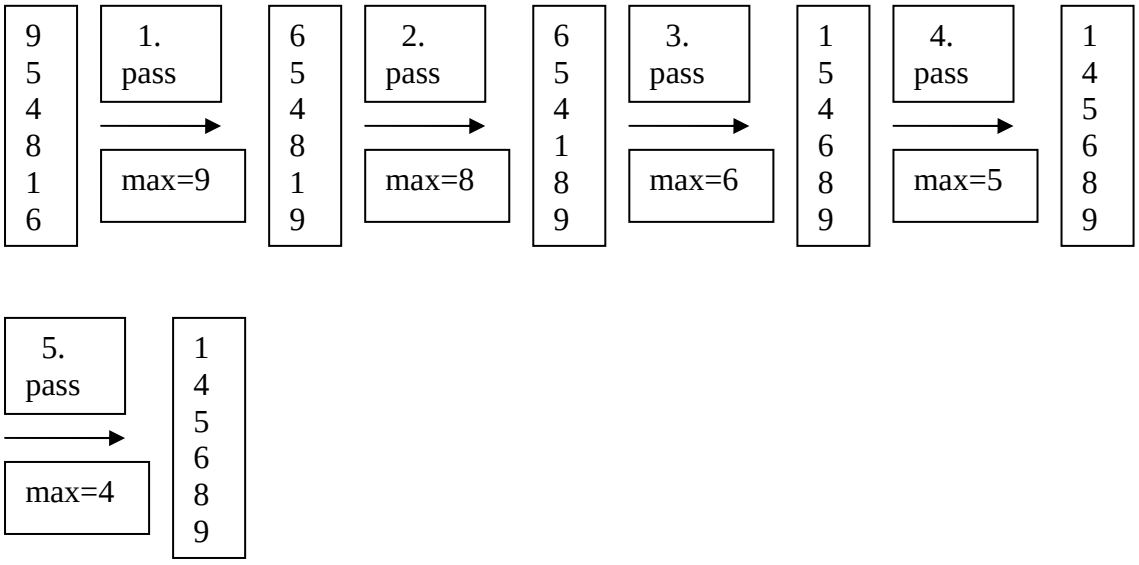
= (1 + 2 …. + n)/n = n(n+1)/2n = (n+1)/2

if n is a very large number ~= n

Selection Sort:

ascending order = increasing order
descending order = decreasing order

Example:



numer of compressions= 5

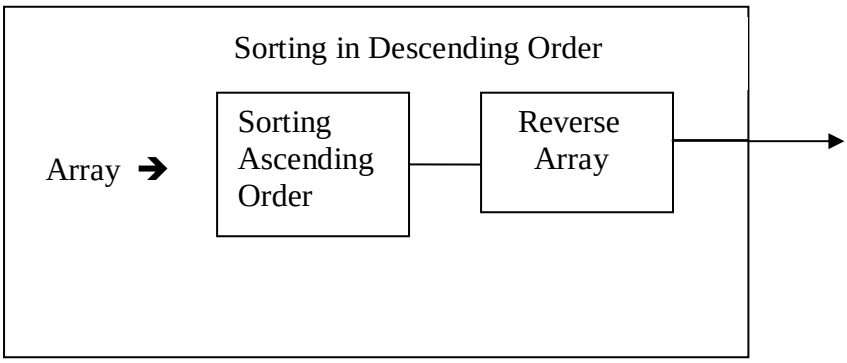
= 1 + 2 + 3 + 4 + 5 => 1 + 2 + …. + (n-1) = n.(n-1) / 2

assuming that n is very large $(n^2/2)-(n/2) \sim= n^2$ Big O notation.

```
public static void selectionSort(double[] list){

for (int I = list.length-1; i >= 1; i--){
// find the maximum in the list [0,…,i]
        double currentMax = list[0];
        int currentMaxIndex;
        for (int j=1; j <= i; j++){
                if (currentMax < list[i]){
                        currentMax = list[i];
```

```
                    currentMaxIndex = j;
                }
            }
            // swap list[i] with currentMax
            if (currentMaxIndex != i){
            list[currentMaxIndex] = list[i];
            }
        }
}
```

Return an Array From a Method:




```
public static int[] reverse(int[] list){
        int[] reverse = new int [list.length];
        for ( int i=0; i < list.length; i++){
                reverse[list.length-i-1]=list[i];
        }
        return reverse;
}
```

Two Dimensional Arrays:

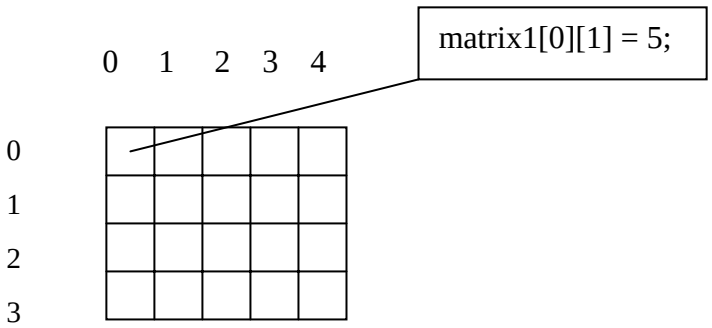
dataType [ ][ ] arrayRefVariable;

int [ ][ ] matrix;

int matrix [ ][ ]; ➔ not preferable

matrix = new int [5][5];

int [ ][ ] matrix1 = new int [5][6];



Initializing a matrix with random values

```
for ( int row = 0; row < matrix.length;
row++ ){
        for ( int column = 0; column <
matrix[row].length;    column++ ){
```

```
        matrix [row] [column] = ( int ) (
        Math.random() * 100 );
        }
}
```

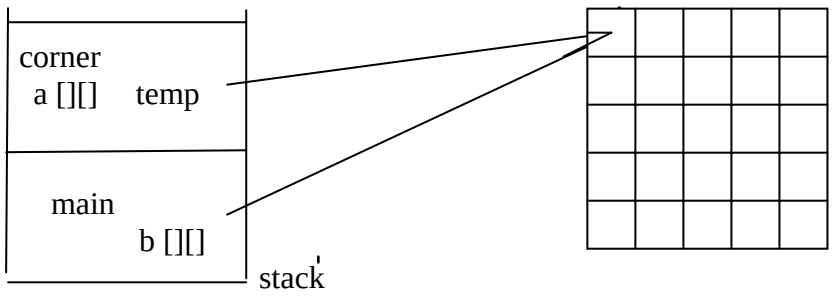
Initialization using an initialization list:

```
int [ ] [ ] = { {1, 2, 3},
                { 4, 5, 6},
                 { 7, 8, 9},
                   { 10, 11, 12}
              };
```

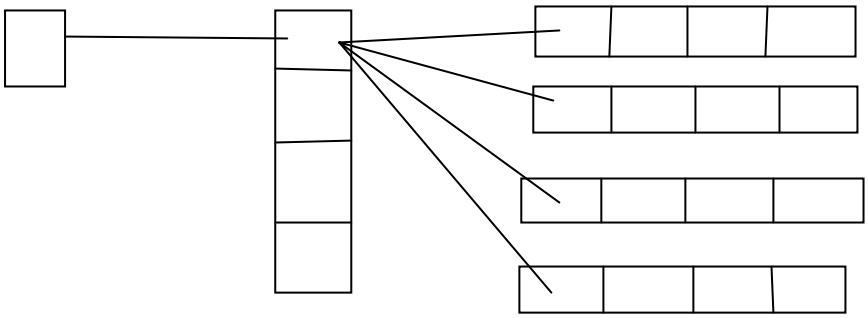
Returning a multidimensional array

```
public static double [ ] [ ] corner ( double [ ]
[ ] a, int size ) {

        double [ ] [ ] temp = new double
[size] [size];
        int row, column;

        for ( int row = 0; row < size; row++ )
        {
                for ( int column = 0; column
< size; column++ ){
                        temp [row] [column] =
a [row] [column];
                }
        }
        return temp;
}

main ()

double [ ] [ ] b = corner (  );
```

For arrays, we use call by reference. We pass the location to methods.
The storage location for arrays is the heap.

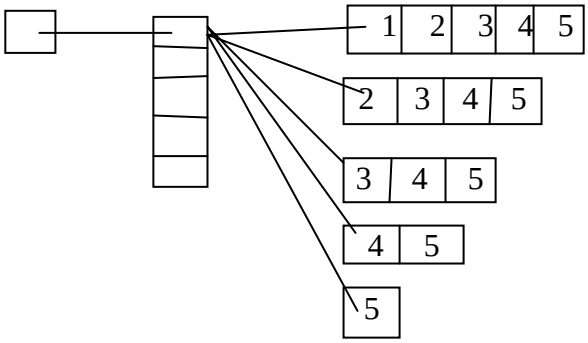


pointer ======> an array of pointers
=======> multidimentional array



Ragged arrays

```
int [ ] [ ]  triangleArray = { { 1,  2, 3, 4, 5 },
                               { 2, 3, 4, 5 },
                               { 3, 4, 5 },
                               { 4, 5 },
                               { 5 }
                             };
```




```
int [ ] [ ]  triangleArray = new int [5] [ ];

triangleArray [0] = new int [5];
triangleArray [1] = new int [4];
triangleArray [2] = new int [3];
triangleArray [3] = new int [2];
triangleArray [4] = new int [1];
```

Chapter 7

Classes and Objects:

Programming Paradigms (Approaches):

Procedural Programming
Object Oriented Programming
Functional Programming

What is a class?

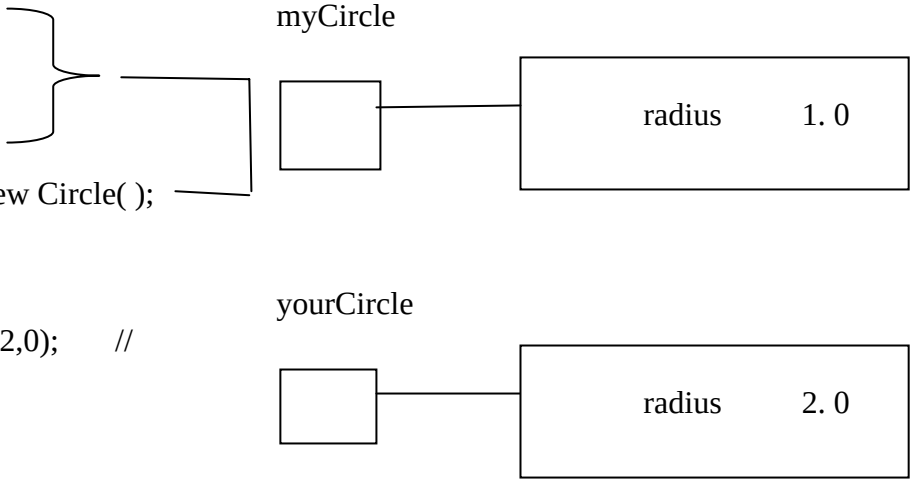
A class is an encapsulation of data
methods.
It is inaccessible and safe.

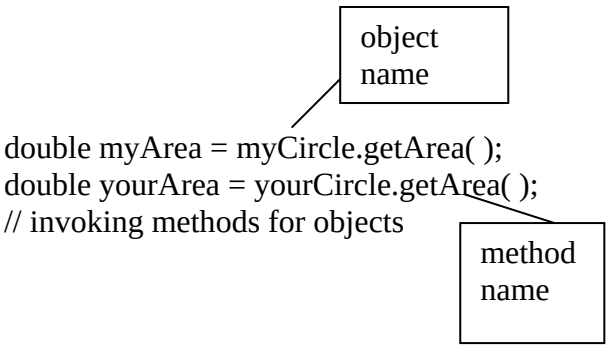
Example:

```
class Circle {
        double radius;

        Circle ( ) = { };//default constructor

        Circle (  double newRadius ){
        //constructor
                radius = newRadius;
        }

        double getArea( ) {
        //method
                return
(radius*radius*Math.PI);
        }
}
```

Another class

```
main( )

Circle myCircle;

myCircle = new Circle( );

Circle myCircle = myCircle = new Circle( );
// invokes the default method


Circle yourCircle = new Circle (2,0);       //
invokes the other constructor
```





Questions:

1. Why does call by value changes the values of the variables?

Because, by call by value local variables are created inside the method and erased from main memory after we finish using that method.

3. How multidimensional arrays are stored in the memory?

A pointer points an array of pointers that is present in the heap memory and each pointer in this array stores the location of one row of the multidimensional array.

5. What should be done to do column wise initiation?

```
for ( int column = 0; column <
matrix[0].length; column++ )
{
        for ( int row = 0; row <
matrix.length; row++ )
        {
                matrix[row][column]=5;
        }
}
```

2. What are the differences between stack and heap type of memory?

Heap memory is a special storage allocation for arrays. Stack memory contains variables and methods, whereas heap memory stores array elements.

4. What is the meaning of encapsulation?

The user of the class does not need to know how the class is implemented so the details of implementation are encapsulated and hidden from the user. This is known as class encapsulation, this also provides safety.