National University of Computer & Emerging Sciences,
Karachi
Computer Science Department
Fall 2023, Lab Manual – 05

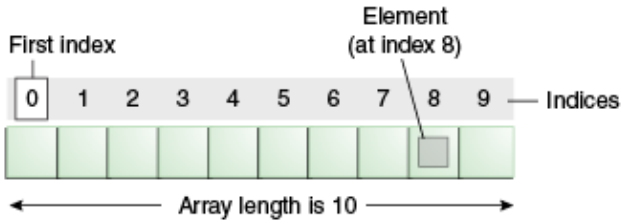
Course Code: CL-217	Course: Object Oriented Programming Lab
Instructor:	Shafique Rehman

LAB - 5

Arrays, Array list, Static Keyword and Final Keyword

Arrays:

Java array is a construct which contains elements of a similar data type. Additionally, the elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array. Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.



Types of Arrays

1. Single Dimensional Array
2. Multi-dimensional Array

Single Dimensional Array

Syntax for Declaration

Option 1: dataType[] arr;
Option 2: dataType []arr;
Option 3: dataType arr[];

Syntax for Instantiation

arr = **new** datatype[size];

Example

```
public class Test {
    public static void main(String args[])
    {
        int a[]=new int[5];//declaration and instantiation
        a[0]=10;//initialization
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;
        //traversing array
        for(int i=0;i<a.length;i++)
            System.out.println(a[i]);
    }
}
```

Multi-Dimensional Array

In such case, data is stored in row and column based index (also known as matrix form).

Syntax for Declaration

dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];

Syntax for Instantiation

int[][] arr=new int[3][3]; //3 row and 3 column

Example

```
public class Test{
    public static void main(String args[])
    {
    //declaring and initializing 2D array
        int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
    //printing 2D array
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();
        }
    }}
```

Passing Arrays to Functions:

We can pass the java array to method so that we can reuse the same logic on any array.

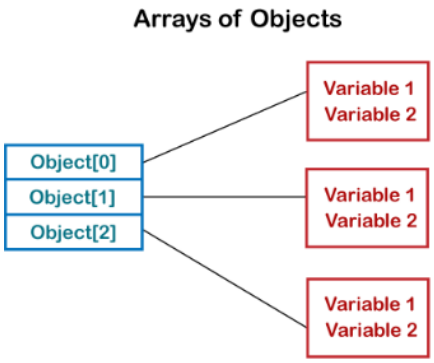
Example

```java
public class Test {
    public static void main(String args[])
    {
        int a[]={33,3,4,5}; //declaring and initializing an array
        min(a); //passing array to method
    }
    static void min(int arr[])
    {
        int min=arr[0];
        for(int i=1;i<arr.length;i++)
            if(min>arr[i])
                min=arr[i];

        System.out.println(min);
    }
}
```

Array of Objects:

Java is an object-oriented programming language. Most of the work done with the help of objects. We know that an array is a collection of the same data type that dynamically creates objects and can have elements of primitive types. Java allows us to store objects in an array. In Java, the class is also a user-defined data type. An array that conations class type elements are known as an array of objects. It stores the reference variable of the object.



Example:

```
public class ArrayOfObjects
{
public static void main(String args[])
{
//create an array of product object
Product[] obj = new Product[2] ;
//create & initialize actual product objects using constructor
obj[0] = new Product(23907,"Dell Laptop");
obj[1] = new Product(91240,"HP 630");

//display the product object data
System.out.println("Product Object 1:");
obj[0].display();
System.out.println("Product Object 2:");
obj[1].display();
}
}
//Product class with product Id and product name as attributes
class Product
{
int pro_Id;
String pro_name;
//Product class constructor
Product(int pid, String n)
{
pro_Id = pid;
pro_name = n;
}
public void display()
{
System.out.print("Product Id = "+pro_Id + "   " + " Product Name = "+pro_name);
System.out.println();
}
}
```

Java ArrayList

ArrayList class uses a *dynamic* array for storing the elements. It is like an array, but there is *no size limit*. We can add or remove elements anytime. So, it is much more flexible than the traditional array. It is found in the *java.util* package.

- ArrayList class can contain duplicate elements.
- ArrayList class maintains insertion order.
- ArrayList allows random access because array works at the index basis.

Commonly used Constructors

- ArrayList( ): It is used to build an empty array list.
- ArrayList (int capacity): It is used to build an array list that has the specified initial capacity.

Common Methods of ArrayList

`add(`**value**`)`	appends value at end of list
`add(`**index**`, `**value**`)`	inserts given value just before the given index, shifting subsequent values to the right
`E get(`**index**`)`	returns the value at given index
`E remove(`**index**`)`	removes/returns value at given index, shifting subsequent values to the left
`E set(`**index**`, `**value**`)`	replaces value at given index with given value, returns element that was previously at index.
`int size()`	returns the number of elements in list
`toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

Additional Methods of ArrayList

`addAll(`**list**`)` `addAll(`**index**`, `**list**`)`	adds all elements from the given list to this list (at the end of the list, or inserts them at the given index)
`contains(`**value**`)`	returns true if given value is found somewhere in this list
`containsAll(`**list**`)`	returns true if this list contains every element from given list
`equals(`**list**`)`	returns true if given other list contains the same elements
`iterator()` `listIterator()`	returns an object used to examine the contents of the list (seen later)
`lastIndexOf(`**value**`)`	returns last index value is found in list (-1 if not found)
`remove(`**value**`)`	finds and removes the given value from this list
`removeAll(`**list**`)`	removes any elements found in the given list from this list
`retainAll(`**list**`)`	removes any elements *not* found in given list from this list
`subList(`**from**`, `**to**`)`	returns the sub-portion of the list between indexes **from** (inclusive) and **to** (exclusive)
`toArray()`	returns the elements in this list as an array

- *You have to import java.util.* to use ArrayList*
- *E refers to type of elements in the above methods*

Example

```java
import java.util.*;
public class Test{
    public static void main(String args[]){
        ArrayList<String> list=new ArrayList<String>();//Creating arraylist
        list.add("Mango");//Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Printing the arraylist object
        System.out.println(list);
    }
}
```

Iterating ArrayList using Iterator

```java
import java.util.*;
public class Test{
    public static void main(String args[]){
        ArrayList<String> list=new ArrayList<String>();//Creating arraylist
        list.add("Mango");//Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Traversing list through Iterator
        Iterator itr=list.iterator();//getting the Iterator
        while(itr.hasNext()){//check if iterator has the elements
            System.out.println(itr.next());//printing the element and move to next
        }
    }
}
```

Example: Sorting ArrayList

```java
import java.util.*;
public class Test{
    public static void main(String args[]){
        //Creating a list of fruits
        List<String> list1=new ArrayList<String>();
        list1.add("Mango");
        list1.add("Apple");
        list1.add("Banana");
        //Sorting the list
        Collections.sort(list1);
        //Traversing list through the for-each loop
        for(String fruit:list1)
            System.out.println(fruit);
    }
}
```

Static Keyword

We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class rather than an instance of the class.

Static Variable

The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc. The static variable gets memory only once in the class area at the time of class loading.

Example

```
public class Test{
    public static void main(String args[]){
        Student s1 = new Student( r: 111, n: "Ali");
        Student s2 = new Student( r: 222, n: "Abid");
        s1.display();
        s2.display();
    }
}
class Student{
    int rollno;//instance variable
    String name;
    static String college ="IT College";//static variable
    //constructor
    Student(int r, String n){
        rollno = r;
        name = n;
    }
    //method to display the values
    void display (){System.out.println(rollno+" "+name+" "+college);}
}
```

Static Methods

A static method belongs to the class rather than the object of a class. A static method can be invoked without the need for creating an instance of a class. It can access static data member and can change the value of it.

Example

```java
public class Test{
    public static void main(String args[]){
        Student.change();
        //creating objects
        Student s1 = new Student( r: 111, n: "Ali");
        Student s2 = new Student( r: 222, n: "Abid");
        Student s3 = new Student( r: 333, n: "Bilal");
        //calling display method
        s1.display();
        s2.display();
        s3.display();
    }
}
class Student{
    int rollno;
    String name;
    static String college = "New Age College";
    static void change(){
        college = "IT College";
    }
    Student(int r, String n){
        rollno = r;
        name = n;
    }
    void display(){System.out.println(rollno+" "+name+" "+college);}
}
```

Final Keyword

The final keyword in java is used to restrict the user. The java final keyword can be used in many contexts. There can be blank final variables which are not initialized at the time of declaration can be initialized in constructor only. Final can be:

1. variable
2. method
3. class

1) Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant).

```
class Bike9{
    1 usage
    final int speedlimit=90;//final variable
    1 usage
    void run(){
        speedlimit=400;
    }
    no usages
    public static void main(String args[]){
        Bike9 obj=new  Bike9();
        obj.run();
    }
}
```

```
C:\Users\Faculty\IdeaProjects\untitled7\src\Main.java:38:9
java: cannot assign a value to final variable speedlimit
```

2) Java final method

If you make any method as final, you cannot override it.

```
35  class Bike{
        no usages
36      final void run(){System.out.println("running");}
37  }
38
    2 usages
39  class Honda extends Bike{
        no usages
40      void run(){System.out.println("running safely with 100kmph");}
41
        no usages
42      public static void main(String args[]){
43          Honda honda= new Honda();
44          honda.run();
45      }
46  }
```

```
C:\Users\Faculty\IdeaProjects\untitled7\src\Main.java:40:10
java: run() in Honda cannot override run() in Bike
  overridden method is final
```

3) Java final class

If you make any class as final, you cannot extend it.

```
35  final class Bike{}
36
    2 usages
37  class Honda1 extends Bike{
        1 usage
38      void run(){System.out.println("running safely with 100kmph");}
39
        no usages
40      public static void main(String args[]){
41          Honda1 honda= new Honda1();
42          honda.run();
43      }
44  }
```

```
C:\Users\Faculty\IdeaProjects\untitled7\src\Main.java:37:22
java: cannot inherit from final Bike
```