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

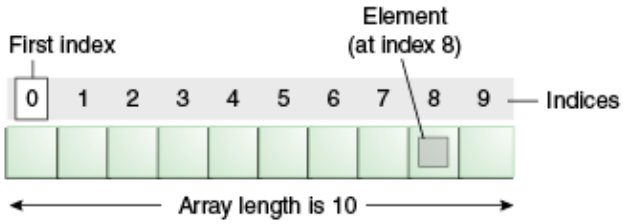
Course Code: CS-1004	Course: Object Oriented Programming Lab
Instructor(s) :	Abeer Gauher, Hajra Ahmed, Shafique Rehman

LAB - 5

Arrays, Array list and Static 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);
    }
}
```

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

```
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);}
}
```

LAB TASKS

Task 1:

You are working as a software engineer for a financial company. The company has a requirement to calculate the daily profit or loss of their stock portfolio. They have a list of the opening and closing prices of stocks for a given day.

- Write a Java program that takes two arrays of doubles as input, one containing the opening prices and the other containing the closing prices of the stocks, and returns an array of doubles representing the daily profit or loss for each stock in the portfolio.
- Your program should include a class named "PortfolioAnalyzer" with a function named "calculateDailyProfitLoss" that takes the input arrays of opening and closing prices as arguments and returns an array of doubles representing the daily profit or loss for each stock.
- For example, if the opening prices array is [10.0, 12.5, 8.0, 15.0] and the closing prices array is [11.0, 13.0, 7.0, 16.0], the program should return an array of doubles containing [1.0, 0.5, -1.0, 1.0], since the profit or loss for each stock can be calculated as the difference between the closing price and the opening price.
- Note: You may assume that the input arrays have the same length, and that the order of the stocks in the opening prices array corresponds to the order of the stocks in the closing prices array.

Task 2:

Write a Java program that takes an ArrayList of integers and removes all duplicates, leaving only the distinct elements in the list.

- Your program should include a class named "DuplicateRemover" with a static function named "removeDuplicates" that takes the input ArrayList as an argument and returns an ArrayList containing only the distinct elements.
- For example, if the input ArrayList is [1, 2, 3, 1, 4, 2, 5], the program should return an ArrayList containing [1, 2, 3, 4, 5].
- Note: You may not use any built-in Java functions or methods that directly remove duplicates from the input ArrayList. You should implement your own algorithm to solve this problem.

Task 3:

Create an Array list that:

- Ask the user to enter 5 numbers as input
- Check if the list contains a prime number, then display that the list has prime numbers and display the sum of the numbers present at even indices like elements present at 0th, 2nd and 4th indices will be added.

- If the list doesn’t contain prime numbers, then display the sum of the numbers present at odd indices like elements present at 1st and 3rd indices will be added.

Task 4:

Create a class called Student that has attributes name, roll number and a static attribute university name.

- Make a static variable counter and initialize it to 0.
- Make a static set method for roll number that increments the counter and returns the counter.
- Create a parameterized constructor that sets the name as the parameter. To set the roll number call the static set roll number method.
- Create a static method that takes a parameter and sets the university name.
- Create a display method that displays the student’s information name, roll number and university.
- In the main, call the set university name method and set it to “FAST University”.
- Create three objects of the student class and display their information along with their roll number that should be different for each student.

Task 5:

Assume you're building a simple car rental application in Java.

- You have a Car class that has the following instance variables: String make, String model, int year, double pricePerDay;
- You also have a Rental class that has the following static variables:
- static int numRentals = 0; static double totalRevenue = 0;
- The Rental class also has a static method called rentCar, which takes in a Car object and the number of days that the car will be rented.
- When the rentCar method is called, it should increase the numRentals static variable by 1 and add the rental price (which is calculated by multiplying the car's pricePerDay by the number of rental days) to the totalRevenue static variable. Write the code for the Rental class, including the rentCar method.

Hint: You can calculate the rental price by multiplying the pricePerDay instance variable of the Car object by the number of rental days, like this: double rentalPrice = car.pricePerDay * numDays;