Java-Lab

Week-1

Command Line Arguments in Java

If any input value is passed through the command prompt at the time of running of the program is known as **command line argument** by default every command line argument will be treated as string value and those are stored in a string array of main() method

Syntax for Compile and Run CMD programs

Compile By -> javac Classname.java
Run By -> java Class value1 value2 value3 .....

1) Implement the following programs using **command line arguments** and **Scanner class**

Program:1 Accept two strings from the user and print it on console with concatenation of "and" in the middle of the strings.

Using CommandLine Arguments

```
class ConcatenateStrings {
    public static void main(String[] args) {
        // Check if the correct number of arguments are provided
        if (args.length != 2) {
            System.out.println("Please provide exactly two strings as arguments.");
            return;
        }

        // Retrieve the command line arguments
        String firstString = args[0];
        String secondString = args[1];

        // Concatenate the strings with "and" in between
        String result = firstString + " and " + secondString;

        // Print the result
        System.out.println("Result: " + result);
    }
}
```

javac ConcatenateStrings.java

output:1
java ConcatenateStrings
Please provide exactly two strings as arguments.

Output:2
java ConcatenateStrings vinod manikanta
Result: vinod and manikanta

Scanner Class:
The Scanner class is used to get user input, and it is found in the java.util package.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user
next().charAt(0)	Reads a character from the user

Using Scanner Class

ConcatenateStringsUsingScanner.java

```
 class ConcatenateStringsUsingScanner {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the console
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the first string
        System.out.print("Enter the first string: ");
        String firstString = scanner.nextLine();

        // Prompt the user to enter the second string
        System.out.print("Enter the second string: ");
        String secondString = scanner.nextLine();

        // Concatenate the strings with "and" in the middle
        String result = firstString + " and " + secondString;

        // Print the result
        System.out.println(result);
    }
}
```

javac ConcatenateStringsUsingScanner.java

java ConcatenateStringsUsingScanner

Enter the first string: vinod
Enter the second string: manikanta
vinod and manikanta

Write a program to find area and of circle using command line arguments and Scanner Class in java

Using Command Line Arguments

AreaAndPermeterOfCircleFromCommandLine.java

```
class AreaAndPermeterOfCircleFromCommandLine {

   public static void main(String[] args) {

      if (args.length > 0) {


            // Parse the first command line argument as a double (radius)

            double radius = Double.parseDouble(args[0]);

            // Calculate perimeter and area

            double perimeter = 2 * 3.14 * radius;

            double area = 3.14 * radius * radius;

            // Print the results

            System.out.println("Radius: " + radius);

            System.out.println("Perimeter of the circle: " + perimeter);

            System.out.println("Area of the circle: " + area);

      } else {

         System.out.println("Please provide the radius as a command line argument.");

      }

   }

}
```

javac AreaAndPermeterOfCircleFromCommandLine.java

java AreaAndPermeterOfCircleFromCommandLine 4

Radius: 4.0

Perimeter of the circle: 25.12

Area of the circle: 50.24

Write a program to find area and of circle using Scanner class in java

Using ScannerClass

AreaAndPerimeterOfCircleUsingScanner.java

```
import java.util.Scanner;
public class AreaAndPerimeterOfCircleUsingScanner {
   public static void main(String[] args) {
      // Create a Scanner object to read input from the console
      Scanner scanner = new Scanner(System.in);
      // Prompt the user to enter the radius
      System.out.print("Enter the radius of the circle: ");
      double radius = scanner.nextDouble();
      // Calculate the perimeter and area of the circle
      double perimeter = 2 * 3.14 * radius;
      double area = 3.14 * radius * radius;
      // Print the results
      System.out.println("Perimeter of the circle: " + perimeter);
      System.out.println("Area of the circle: " + area);
   }
}
```

```
javac AreaAndPerimeterOfCircleUsingScanner.java
```

```
java AreaAndPerimeterOfCircleUsingScanner
Enter the radius of the circle: 3.2
Perimeter of the circle: 20.096000000000004
Area of the circle: 32.153600000000004
```

Week-2

Constructor

Constructors are meant for initializing the object. Constructor is a special type of method that is used to initialize object values.

Constructor is invoked at the time of object creation. It constructs the values i.e. data for the object that is why it is known as constructor.

Constructor is just like the instance method but it does not have any explicit return type.

Constructor name must be same as its class name.
Constructor should not return any value even void also.

Constructors are called automatically whenever an object is cereating.

Types of Constructors:
There are two types of constructors:-
1. Default constructor (no-argument constructor)
2. Parameterized constructor

1. Default constructor (no-argument constructor):-

A constructor is one which will not take any parameter.

A constructor that have no parameter is known as default constructor.

Syntax:-

```
class < class name >
{
        classname() //default constructor
        {
                Block of statements;
                .....;
                .....;
        }
        .....;
        .....;
};
```

2. Parameterized Constructor:- A constructor is one which takes some parameters.

Syntax:-

```
class < class name >
{
        classname(list of parameters) //paramiterized constructor
        {
                Block of statements;
                .....;
                .....;
        }
        .....;
        .....;
};
```

Constructor overloading is a technique in Java in which a class can have any number of constructors such that each constructor differ with their parameters

Constructors with diff arguments is known as Constructor Overloading

Constructor over loading example

Write a program to call the default constructor first and then any other constructor in the class.

Student.java

```
class Student
{
int rollNumber;
String branchName;
        Student()
        {
        rollNumber=100;
        branchName="CSE";
        System.out.println(rollNumber);
        System.out.println(branchName);
        }
        Student(int rollNumber)
        {
        this.rollNumber=rollNumber;
        branchName="CSE";
        System.out.println(rollNumber);
```

```
        System.out.println(branchName);

        }
        Student(int rollNumber,String branchName)
        {
        this.rollNumber=rollNumber;
        this.branchName=branchName;
        System.out.println(rollNumber);
        System.out.println(branchName);

        }
public static void main(String args[])
{
        Student ravi=new Student();
        System.out.println("-----------");
        Student seetha=new Student(101);
        System.out.println("-----------");
        Student balu=new Student(102,"CSE");

}




}
```

Output

```
javac Student.java
java Student
100
CSE
-----------
101
CSE
-----------
102
CSE
```

Array in java

Array is a collection of similar type of data. It is fixed in size means that you can't increase the size of array at run time. It is a collection of homogeneous data elements. It stores the value on the basis of the index value.

Advantage of Array

One variable can store multiple value: The main advantage of the array is we can represent multiple value under the same name.

Random access: We can retrieve any data from array with the help of the index value.

Disadvantage of Array

The main limitation of the array is **Size Limit** when once we declare array there is no chance to increase and decrease the size of an array according to our requirement, Hence memory point of view array concept is not recommended to use. To overcome this limitation in Java introduce the collection concept.

Note: At the time of array declaration we cannot specify the size of the array. For Example int[5] a; this is wrong.

Syntax Array in Java

```
1.   int[][]  a;
2.   int  a[][];
3.   int  [][]a;
4.   int[]  a[];
5.   int[]  []a;
6.   int   []a[];
```

Array creation

Every array in a Java is an object, Hence we can create array by using **new** keyword.

Single dimensional Arrays

Syntax

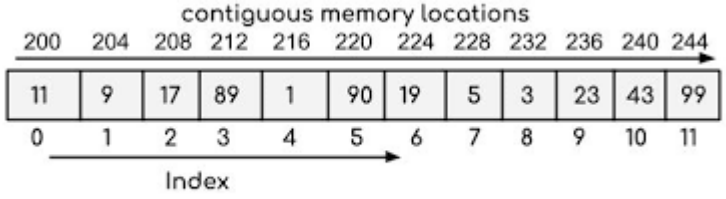
```
int[] arr = new int[10];    //  declare, instantiate
or
int[] arr = {10,20,30,40,50};//declare instantiate, initialize

Multi dimensional arrays

int[][] arr=new int[2][3];// declare instantiate
or
int [][] arr={{1,2},{2,3,4,},{4,5,6}};// declare instantiate ,initialize
```

How the memory allocation of arrays in java. Is array object? If yes explain.

```
Yes Array is an object in java
Memory is stored  as contagious memory locations in heap .
```



Write a program that accepts an array of integers and print those which are both odd and prime. If no such element in that array print “Not found”.

```
import java.util.Scanner;
 class OddAndPrime {
   // Function to check if a number is prime
   boolean isPrime(int n)
   {
      // Corner case
      if (n <= 1)
         return false;

      // Check from 2 to n-1
      for (int i = 2; i < n; i++)
         if (n % i == 0)
            return false;
      return true;
   }
   // Function to check if a number is odd
    boolean isOdd(int num) {
      return num % 2 != 0;
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      OddAndPrime op=new OddAndPrime();
      System.out.print("Enter the number of elements in the array: ");
      int n = scanner.nextInt();
      int[] array = new int[n];
      System.out.println("Enter the elements of the array:");
```

```
        for (int i = 0; i < n; i++) {
            array[i] = scanner.nextInt();
        }
        boolean found = false;
        System.out.println("Odd and prime numbers in the array:");
        for (int i=0;i<array.length;i++) {
            if (op.isOdd(array[i]) && op.isPrime(array[i])) {
                System.out.print(array[i] + " ");
                found = true;
            }
        }

        if (!found) {
            System.out.println("Not found");
        }
    }
}
```

```
javac OddAndPrime.java
java OddAndPrime
Enter the number of elements in the array: 8
Enter the elements of the array:
4
5
7
3
2
7
```

9

5

Odd and prime numbers in the array:

5 7 3 7 5

Write a program that accepts an ‘m x n’ double dimension array, where ‘m’ represents financial years and ‘n’ represents Ids of the items sold. Each element in the array represents the number of items sold in a particular year. Identify the year and id of the item which has more demand.

HighestDemandFinder.java

```
import java.util.Scanner;

 class HighestDemandFinder {


   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);


      // Input the dimensions of the array

      System.out.println("Enter the number of financial years (m): ");

      int m = scanner.nextInt();

      System.out.println("Enter the number of item IDs (n): ");

      int n = scanner.nextInt();


      int[][] salesData = new int[m][n];


      // Input the sales data

      System.out.println("Enter the sales data (number of items sold) for each year and item ID:");

      for (int i = 0; i < m; i++) {

         for (int j = 0; j < n; j++) {

            System.out.printf("Year %d, Item ID %d: ", i + 1, j + 1);

            salesData[i][j] = scanner.nextInt();
```

```
            }
        }

        // Initialize variables to track the maximum sales
        int maxSales = 0;
        int maxYear = 0;
        int maxItemId = 0;

        // Find the maximum sales and corresponding year and item ID
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (salesData[i][j] > maxSales) {
                    maxSales = salesData[i][j];
                    maxYear = i + 1;
                    maxItemId = j + 1;
                }
            }
        }

        // Output the result
        System.out.println("Year with highest demand: " + maxYear);
        System.out.println("Item ID with highest demand: " + maxItemId);
        System.out.println("Number of items sold: " + maxSales);

        scanner.close();
    }
}
```

javac HighestDemandFinder.java

java HighestDemandFinder

Enter the number of financial years (m):

4

Enter the number of item IDs (n):

2

Enter the sales data (number of items sold) for each year and item ID:

Year 1, Item ID 1: 3

Year 1, Item ID 2: 4

Year 2, Item ID 1: 5

Year 2, Item ID 2: 6

Year 3, Item ID 1: 3

Year 3, Item ID 2: 3

Year 4, Item ID 1: 4

Year 4, Item ID 2: 5

Year with highest demand: 2

Item ID with highest demand: 2

Number of items sold: 6

Matrix multiplication in java

```
import java.util.Scanner;
public class MatrixMultiplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read dimensions of the first matrix
        System.out.print("Enter the number of rows in the first matrix: ");
```

```
        int rows1 = scanner.nextInt();
        System.out.print("Enter the number of columns in the first matrix: ");
        int cols1 = scanner.nextInt();

        // Read dimensions of the second matrix
        System.out.print("Enter the number of rows in the second matrix: ");
        int rows2 = scanner.nextInt();
        System.out.print("Enter the number of columns in the second matrix: ");
        int cols2 = scanner.nextInt();

        // Check if the multiplication is possible
        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible. Number of columns in the first matrix must equal the number of rows in the second matrix.");
            return;
        }

        // Initialize the matrices
        int[][] matrix1 = new int[rows1][cols1];
        int[][] matrix2 = new int[rows2][cols2];
        int[][] result = new int[rows1][cols2];

        // Read elements of the first matrix
        System.out.println("Enter the elements of the first matrix:");
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols1; j++) {
                matrix1[i][j] = scanner.nextInt();
            }
```

```
}

// Read elements of the second matrix
System.out.println("Enter the elements of the second matrix:");
for (int i = 0; i < rows2; i++) {
    for (int j = 0; j < cols2; j++) {
        matrix2[i][j] = scanner.nextInt();
    }
}

// Perform matrix multiplication
for (int i = 0; i < rows1; i++) {
    for (int j = 0; j < cols2; j++) {
        result[i][j] = 0; // Initialize the element at (i, j) in the result matrix
        for (int k = 0; k < cols1; k++) {
            result[i][j] += matrix1[i][k] * matrix2[k][j];
        }
    }
}

// Display the resulting matrix
System.out.println("Resulting matrix after multiplication:");
for (int i = 0; i < rows1; i++) {
    for (int j = 0; j < cols2; j++) {
        System.out.print(result[i][j] + " ");
    }
    System.out.println();
```

```
        }
    }
}
```

Output

```
javac MatrixMultiplication.java
java MatrixMultiplication
Enter the number of rows in the first matrix: 2
Enter the number of columns in the first matrix: 2
Enter the number of rows in the second matrix: 2
Enter the number of columns in the second matrix: 2
Enter the elements of the first matrix:
1 2 3 4
Enter the elements of the second matrix:
1 2 3 4
Resulting matrix after multiplication:
7 10
15 22
```