

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 01		
Title Of Lab Practical: INTRODUCTION TO JAVA				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

Practical No 1

A) Program to find prime no between 1 to N

Aim: Write a program to find prime no between 1 to N in java.

Description:

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Software Development Domain. Some of the key advantages of learning Java Programming:

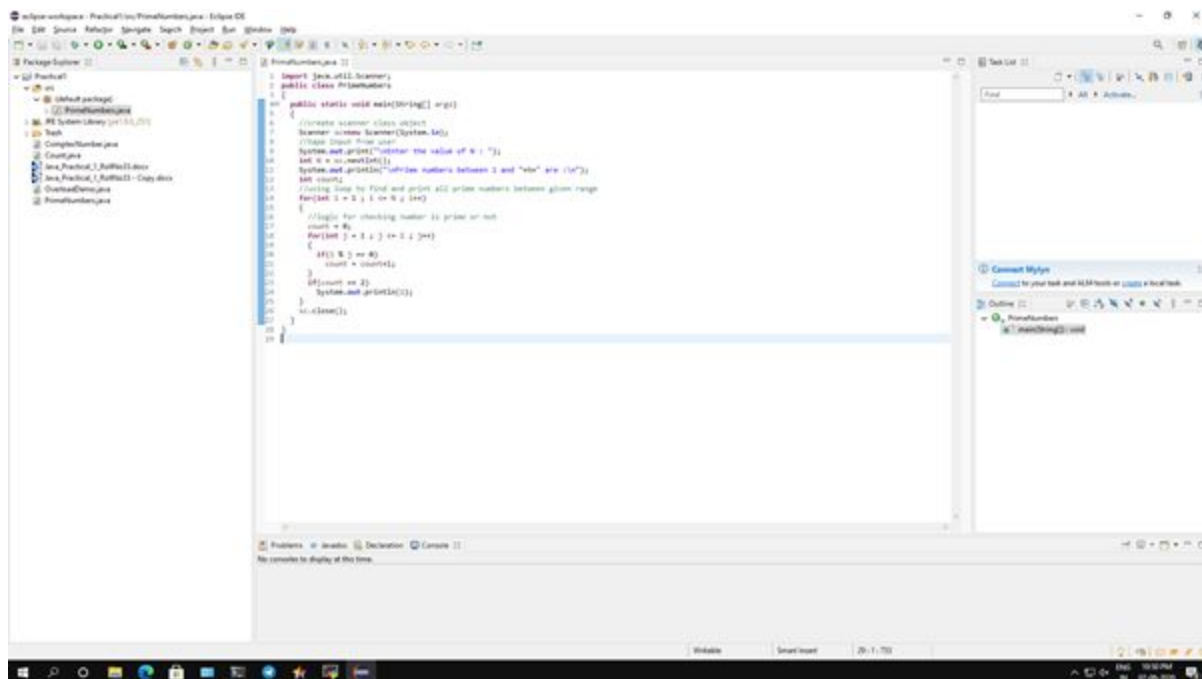
- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into a platform specific machine, rather into platform independent byte code.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral:** Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of a Java runtime system.
- **Portable:** Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java are written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

We have defined the packages that are imported in the program which is the Scanner util packages. A package in java is used to group related classes. We use packages to avoid name conflicts, and to write a better maintainable code. After the packages we have defined the class name which is class PrimeNumbers. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have created a Scanner class object and then we have defined the main method to accept an array of string arguments. Then it will take the input from the users when we enter the value of N. Once the value is entered it will go into loop till, we find the prime numbers until the value N. It will use

logic for checking the prime number till the N value and once we find the prime number the program will stop compiling and get the output.

Some of the Keywords used are:

- **Import:** import is a Java keyword. It declares a Java class to use in the code below the import statement.
- **Class:** A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type.
- **Public:** Public is a Java keyword which declares a member's access as public. Public members are visible to all other classes.
- **Static:** In Java, static keyword is mainly used for memory management. It can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or method of a given class.
- **Void:** The void keyword specifies that a method should not have a return value.
- **New:** It creates a Java object and allocates memory for it on the heap. new is also used for array creation, as arrays are also objects.
- **System.in:** System.in in java means to take input from the keyboard or user.
- **System.out.print:** System.out.print in java means to print the output to console.
- **System.out.println:** System.out.println in java means to print the output to console on the next line.



```
1 import java.util.Scanner;
2 public class PrimeNumber {
3
4     public static void main(String[] args) {
5
6         //Create scanner class object
7         Scanner scanner = new Scanner(System.in);
8         //Take input from user
9         System.out.print("Enter the value of N : ");
10        int n = scanner.nextInt();
11        System.out.println("Prime numbers between 1 and " + n + " are :");
12        int count = 0;
13        //Loop to find and print all prime numbers between given range
14        for(int i = 2; i <= n; i++) {
15            //Logic for checking number is prime or not
16            count = 0;
17            for(int j = 2; j <= i; j++) {
18                if(i % j == 0) {
19                    count = count + 1;
20                }
21            }
22            if(count == 1) {
23                System.out.println(i);
24            }
25        }
26        scanner.close();
27    }
28 }
```

Enter the value of N : 10

Prime numbers between 1 and 10 are : 2, 3, 5, 7

Conclusion: We have written a program to find prime number between 1 to N in java.

Code:

```
import java.util.Scanner;
public class PrimeNumbers
{
    public static void main(String[] args)
    {
        //create scanner class object
        Scanner sc=new Scanner(System.in);
        //take input from user
        System.out.print("\nEnter the value of N: ");
        int N = sc.nextInt();
        System.out.println("\nPrime numbers between 1 and "+N+" are:\n");
        int count;
        //using loop to find and print all prime numbers between given range
        for(int i = 1 ; i <= N ; i++)
        {
            //logic for checking number is prime or not
            count = 0;
            for(int j = 1 ; j <= i ; j++)
            {
                if(i % j == 0)
                    count = count+1;
            }
            if(count == 2)
                System.out.println(i);
        }
        sc.close();
    }
}
```

Output:

```
Enter the value of N : 20
Prime numbers between 1 and 20 are :
2
3
5
7
11
13
17
19
```

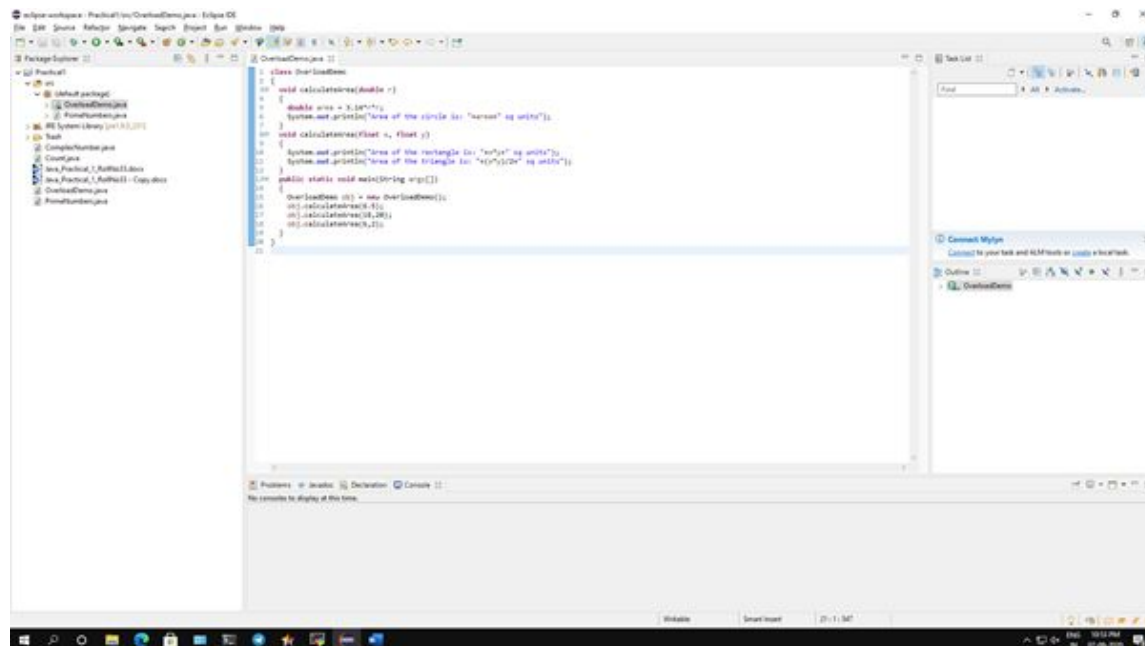
B) Program to find the function Overloading to find area of circle, rectangle and triangle.

Aim: Write a java program to find the function Overloading to find the area of circle, rectangle and triangle.

Description:

We can have two or more methods in java to have the same names with different parameters. These methods are called method overloading. It is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java.

In the given code we have taken three methods of the same name but with different numbers of arguments with different data types. All the three methods have the same name with different return types, which is much overloading. We also passed different parameters of single, integer and double to calculate the area of circle, rectangle and triangle.



```
1 class OverloadDemo {
2     void calculateArea(double r) {
3         double area = 3.14*r*r;
4         System.out.println("Area of the circle is: " + area + " sq units");
5     }
6     void calculateArea(float x, float y) {
7         System.out.println("Area of the rectangle is: " + x*y + " sq units");
8         System.out.println("Area of the triangle is: " + (x*y)/2 + " sq units");
9     }
10    public static void main(String srg[]) {
11        OverloadDemo o = new OverloadDemo();
12        o.calculateArea(5.5);
13        o.calculateArea(10, 20);
14        o.calculateArea(10, 15);
15    }
16 }
```

Conclusion: We have written a program on function Overloading to find the area of circle, rectangle and triangle.

Code:

```
class OverloadDemo
{
    void calculateArea(double r)
    {
        double area = 3.14*r*r;
        System.out.println("Area of the circle is: "+area+" sq units");
    }
    void calculateArea(float x, float y)
    {
        System.out.println("Area of the rectangle is: "+x*y+" sq units");
        System.out.println("Area of the triangle is: +(x*y)/2+" sq units");
    }
    public static void main(String args[])
    {
        OverloadDemo obj = new OverloadDemo();
        obj.calculateArea(6.5);
        obj.calculateArea(18,20);
        obj.calculateArea(6,2);
    }
}
```

Output:

```
Area of the circle is: 132.665 sq units
Area of the rectangle is: 360.0 sq units
Area of the triangle is: 180.0 sq units
Area of the rectangle is: 12.0 sq units
Area of the triangle is: 6.0 sq units
```

C) Program to implement a program to perform complex number addition, multiplication, subtraction using constructors overloading.

Aim: Write a java program to implement a program to perform complex number addition, multiplication, subtraction using constructors overloading.

Description:

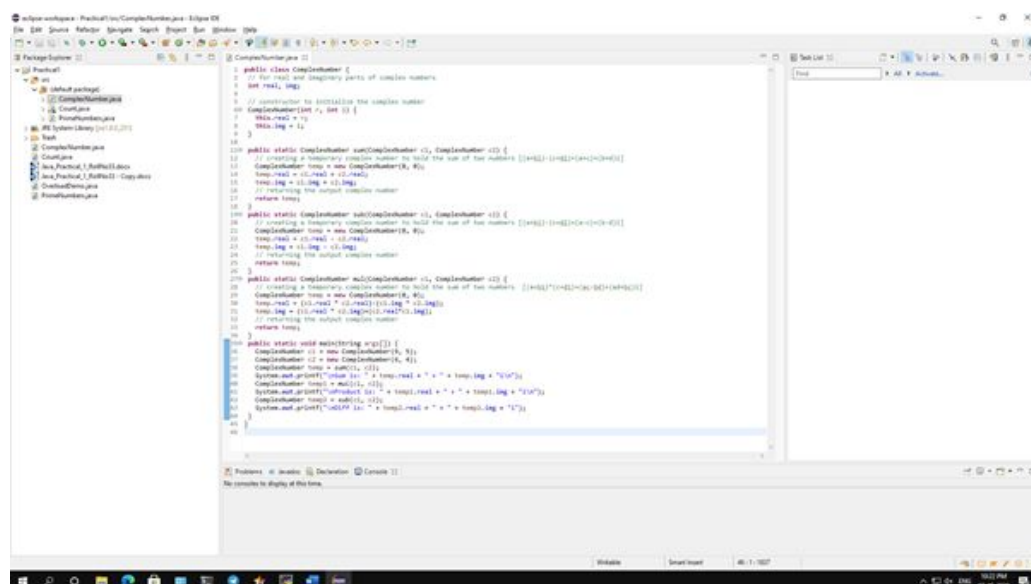
We can overload a constructor in java which can be done using constructor overloading.

Constructor overloading is a concept of having more than one constructor with different parameters list, in such a way that each constructor performs a different task. There are two types of constructors they are Default constructor and Parameterized constructor.

Following rules must be used for creating java constructor:

- Constructor name must be the same as its class name
- A Constructor must have no explicit return type

In the given code we have created a parameter constructor where 2 arguments can pass data type integers. We then created 3 methods for addition, subtraction and multiplication to find the complex numbers. Then the rest of the required operation will be performed in the main method, as these methods will return the values which are passed to another object of class.



```

1 public class ComplexNumber {
2     int real, img;
3
4     // constructor to initialize the complex number
5     ComplexNumber(int r, int i) {
6         this.real = r;
7         this.img = i;
8     }
9
10    // constructor to initialize the complex number
11    ComplexNumber() {
12        this.real = 0;
13        this.img = 0;
14    }
15
16    // creating a temporary complex number to hold the sum of two numbers [(a+bi)+(c+di)=(a+c)+(b+d)i]
17    ComplexNumber sum(ComplexNumber c1, ComplexNumber c2) {
18        ComplexNumber temp = new ComplexNumber(0, 0);
19        temp.real = c1.real + c2.real;
20        temp.img = c1.img + c2.img;
21        // returning the output complex number
22        return temp;
23    }
24
25    // creating a temporary complex number to hold the sum of two numbers [(a+bi)-(c+di)=(a-c)+(b-d)i]
26    ComplexNumber sub(ComplexNumber c1, ComplexNumber c2) {
27        ComplexNumber temp = new ComplexNumber(0, 0);
28        temp.real = c1.real - c2.real;
29        temp.img = c1.img - c2.img;
30        // returning the output complex number
31        return temp;
32    }
33
34    // creating a temporary complex number to hold the sum of two numbers [(a+bi)*(c+di)=(ac-bd)+(ad+bc)i]
35    ComplexNumber mul(ComplexNumber c1, ComplexNumber c2) {
36        ComplexNumber temp = new ComplexNumber(0, 0);
37        temp.real = c1.real * c2.real - c1.img * c2.img;
38        temp.img = c1.real * c2.img + c1.img * c2.real;
39        // returning the output complex number
40        return temp;
41    }
42
43    // displaying the output complex number
44    void displaying() {
45        ComplexNumber c1 = new ComplexNumber(1, 2);
46        ComplexNumber c2 = new ComplexNumber(3, 4);
47        ComplexNumber temp = sum(c1, c2);
48        System.out.println("Sum = " + temp.real + " + " + temp.img + "i");
49        temp = sub(c1, c2);
50        System.out.println("Sub = " + temp.real + " + " + temp.img + "i");
51        temp = mul(c1, c2);
52        System.out.println("Mul = " + temp.real + " + " + temp.img + "i");
53    }
54 }
55
56 // Driver class
57 public class Main {
58     public static void main(String[] args) {
59         ComplexNumber c1 = new ComplexNumber(1, 2);
60         ComplexNumber c2 = new ComplexNumber(3, 4);
61         c1.displaying();
62     }
63 }

```

Conclusion: We have implemented a program to perform complex number addition, multiplication, subtraction using constructors overloading.

Code:

```
public class ComplexNumber {
    // for real and imaginary parts of complex numbers
    int real, img;

    // constructor to initialize the complex number
    ComplexNumber(int r, int i) {
        this.real = r;
        this.img = i;
    }

    public static ComplexNumber sum(ComplexNumber c1, ComplexNumber c2) {
        // creating a temporary complex number to hold the sum of two numbers  $[(a+bi)-(c+di)=(a+c)+(b+d)i]$ 
        ComplexNumber temp = new ComplexNumber(0, 0);
        temp.real = c1.real + c2.real;
        temp.img = c1.img + c2.img;
        // returning the output complex number
        return temp;
    }

    public static ComplexNumber sub(ComplexNumber c1, ComplexNumber c2) {
        // creating a temporary complex number to hold the sum of two numbers  $[(a+bi)-(c+di)=(a-c)+(b-d)i]$ 
        ComplexNumber temp = new ComplexNumber(0, 0);
        temp.real = c1.real - c2.real;
        temp.img = c1.img - c2.img;
        // returning the output complex number
        return temp;
    }

    public static ComplexNumber mul(ComplexNumber c1, ComplexNumber c2) {
        // creating a temporary complex number to hold the sum of two numbers
         $[(a+bi) \times (c+di) = (ac-bd) + (ad+bc)i]$ 
        ComplexNumber temp = new ComplexNumber(0, 0);
```



```
temp.real = (c1.real * c2.real)-(c1.img * c2.img);
temp.img = (c1.real * c2.img)+(c2.real*c1.img);
// returning the output complex number
return temp;
}

public static void main(String args[]) {
    ComplexNumber c1 = new ComplexNumber(9, 5);
    ComplexNumber c2 = new ComplexNumber(6, 4);
    ComplexNumber temp = sum(c1, c2);
    System.out.printf("\nSum is: " + temp.real + " + " + temp.img + "i\n");
    ComplexNumber temp1 = mul(c1, c2);
    System.out.printf("\nProduct is: " + temp1.real + " + " + temp1.img + "i\n");
    ComplexNumber temp2 = sub(c1, c2);
    System.out.printf("\nDiff is: " + temp2.real + " + " + temp2.img + "i");
}
}
```

Output:

```
Sum is: 15 + 9i
Product is: 34 + 66i
Diff is: 3 + 1i
```

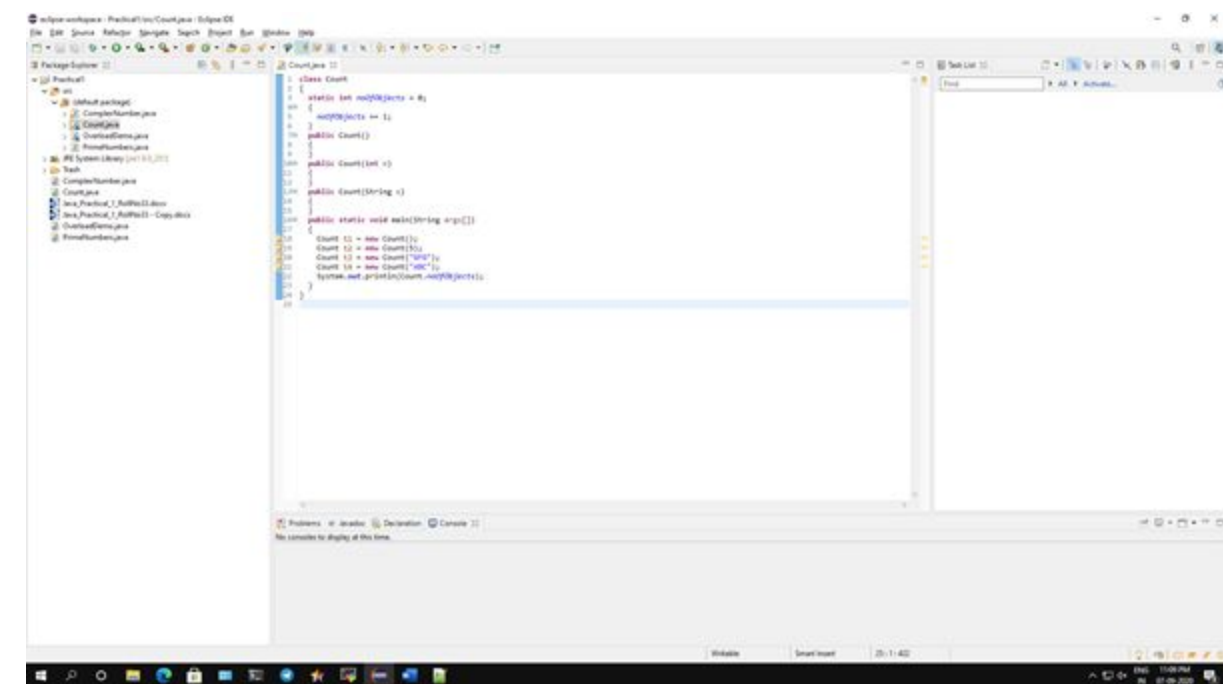
D) Program to count no of objects Created using Static method.

Aim: Write a java program to count no of object Created using Static method.

Description:

In Java, a static method is a method that belongs to a class rather than an instance of a class. The method is accessible to every instance of a class, but methods defined in an instance are only able to be accessed by that member of a class.

In the given program we have initialized the counter value noOfObjects with 0. This type of variable is static. Then we have defined the public class count. Now in the main static method we have all 4 constructors which are called as 4 objects of the same class. Each time when an object is created the counter value noOfObjects increments to 1. So, the objects were created 4 times and we got the counter value as 4.



The screenshot shows an IDE window titled 'workspace - Project\src\Count.java - Eclipse IDE'. The main editor displays the following Java code:

```
1 class Count
2 {
3     static int noOfObjects = 0;
4     public Count()
5     {
6         noOfObjects++;
7     }
8     public Count(int n)
9     {
10        noOfObjects++;
11    }
12    public Count(String s)
13    {
14        noOfObjects++;
15    }
16    public static void main(String args[])
17    {
18        Count c1 = new Count();
19        Count c2 = new Count(5);
20        Count c3 = new Count("NINAD");
21        Count c4 = new Count("ABC");
22        System.out.println("Count: "+noOfObjects);
23    }
24 }
```

The left sidebar shows a project structure with a package 'src' containing files like 'Count.java', 'ComplexNumber.java', 'OverloadDemo.java', and 'PrintDemo.java'. The bottom status bar indicates 'Runnable', 'Smart Insert', and '20:11:40'.

Conclusion: We have written a program to count no of object of object Created using the Static method.

Code:

```
class Count
{
    static int noOfObjects = 0;
    {
        noOfObjects += 1;
    }
    public Count()
    {
    }
    public Count(int n)
    {
    }
    public Count(String s)
    {
    }
    public static void main(String args[])
    {
        Count t1 = new Count();
        Count t2 = new Count(5);
        Count t3 = new Count("GFG");
        Count t4 = new Count("ABC");
        System.out.println("Number of object created: "+noOfObjects);
    }
}
```

Output:

```
Number of object created: 4
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 02		
Title Of Lab Practical: PROGRAM BASED ON ARRAY AND STRINGS				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

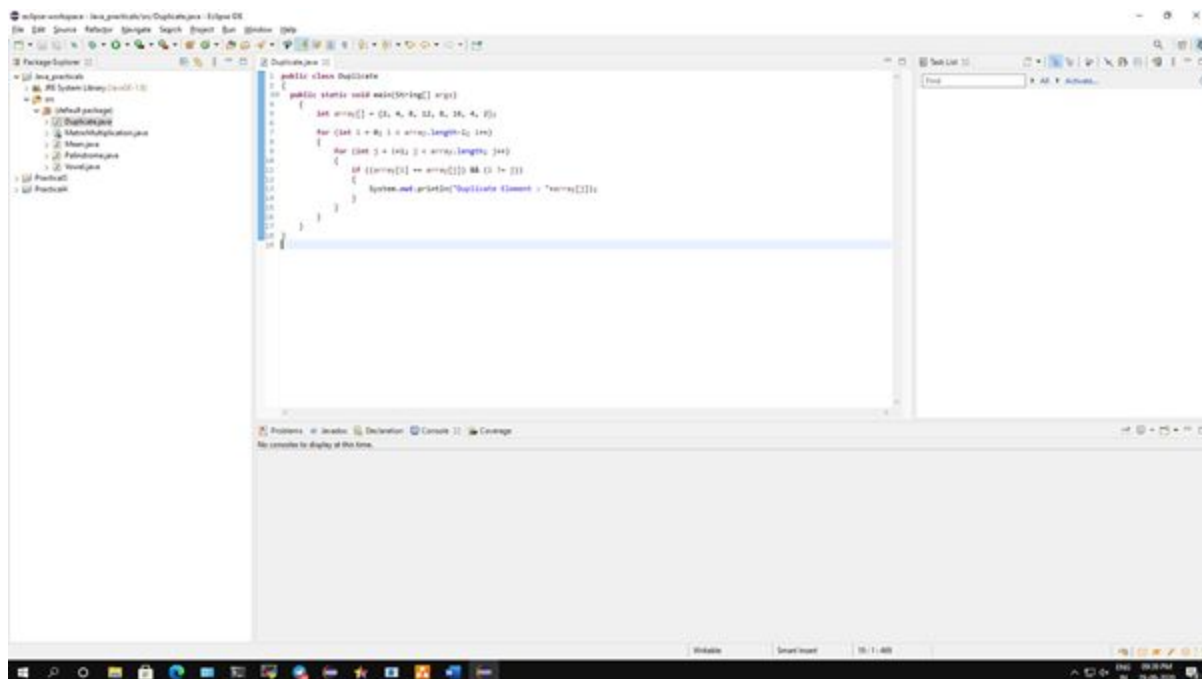
Practical No 2

A) Program to check if the array has any duplicate element

Aim: Write a program to check if the array has any duplicate element in java.

Description:

An array is a collection of similar types of elements which has contiguous memory location. Java array is an object 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. Arrays in Java are index-based, the first element of the array is stored at the 0th index, the 2nd element is stored on the 1st index and so on. Unlike C/C++, we can get the length of the array using the length member. In C/C++, we need to use the sizeof operator. In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimensional or multidimensional arrays in Java. We have defined the class name which is class Duplicate. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have defined the main method to accept an array of string arguments. We have 2 loops in one set of arrays, the first loop i.e. i will loop from start to end traversal. Then second loop i.e. j which is an inner loop which will take the value i to compare with another loop to find the duplicate array. Once we have found the duplicate array it will stop compiling and get the output.



```
1 public class Duplicate
2 {
3     public static void main(String[] args)
4     {
5         int array[] = {1, 4, 8, 32, 8, 16, 4, 2};
6         for (int i = 0; i < array.length-1; i++)
7         {
8             for (int j = i+1; j < array.length; j++)
9             {
10                 if (array[i] == array[j])
11                 {
12                     System.out.println("Duplicate Element = " + array[i]);
13                 }
14             }
15         }
16     }
17 }
```

Conclusion: We have written a program to check if the array has any duplicate element in java.

Code:

```
public class Duplicate
{
    public static void main(String[] args)
    {
        int array[] = {2, 4, 8, 12, 8, 16, 4, 2};

        for (int i = 0; i < array.length-1; i++)
        {
            for (int j = i+1; j < array.length; j++)
            {
                if ((array[i] == array[j]) && (i != j))
                {
                    System.out.println("Duplicate Element: "+array[j]);
                }
            }
        }
    }
}
```

Output:

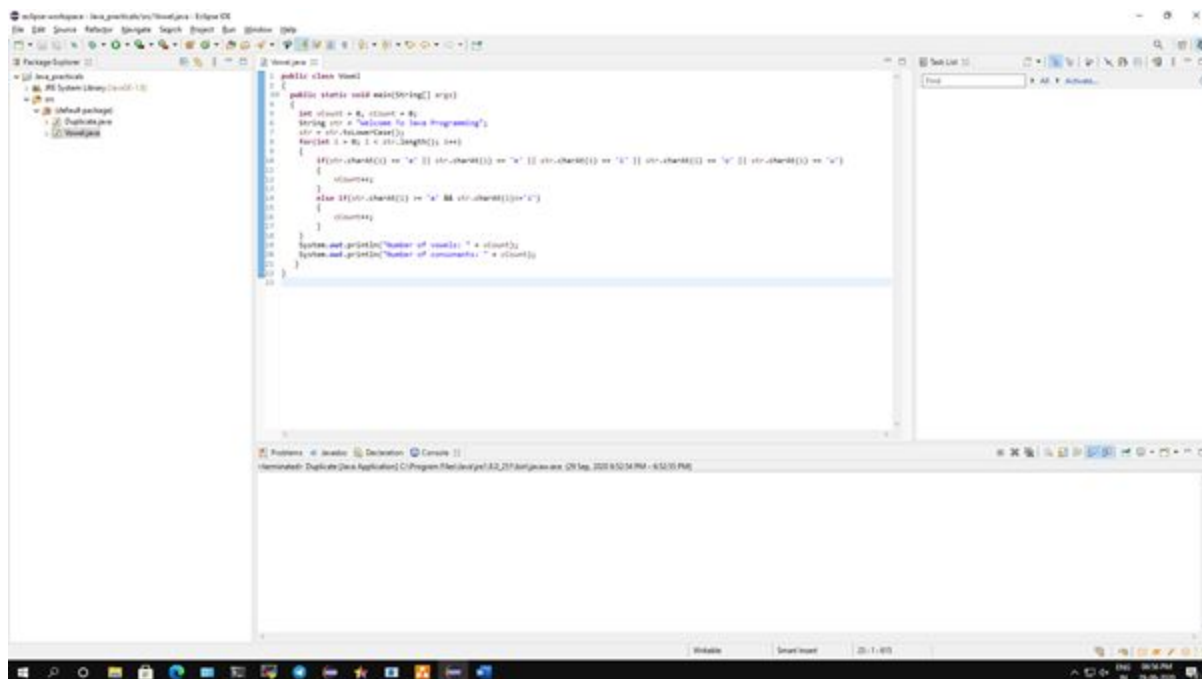
```
<terminated> Duplicate [Java Application]
Duplicate Element : 2
Duplicate Element : 4
Duplicate Element : 8
```

B) Program to count the no of Vowel and Consonant in the given string.

Aim: Write a java program to count the no of Vowel and Consonant in the given string.

Description:

We have defined the class name which is class Vowel. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Our task is to count the total number of vowels and consonants present in the given string. We know that the characters a, e, i, o, u are known as vowels in the English alphabet and any other character than vowels are known as consonants. Now we need to convert every upper-case character in the string to lower-case so that the comparisons can be done with the lower-case vowels only not upper-case vowels, i.e. (A, E, I, O, U). Then, we have to traverse the string using a for or while loop and match each character with all the vowels, i.e., a, e, i, o, u. If the match is found, increase the value of count by 1 otherwise continue with the normal flow of the program. Once the code is compiled, we will get the number of vowels and consonants in the output.



```
1 public class Vowel {
2     public static void main(String[] args) {
3         int vowel = 0, count = 0;
4         String str = "Welcome to Java Programming";
5         str = str.toLowerCase();
6         for(int i = 0; i < str.length(); i++)
7         {
8             if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' || str.charAt(i) == 'u')
9             {
10                 vowel++;
11             }
12             else if(str.charAt(i) != 'a' && str.charAt(i) != 'e')
13             {
14                 count++;
15             }
16         }
17         System.out.println("Number of vowel: " + vowel);
18         System.out.println("Number of consonant: " + count);
19     }
20 }
```

Conclusion: We have written a program to count the no of Vowel and Consonant in the given string.

Code:

```
public class Vowel
{
    public static void main(String[] args)
    {
        int vCount = 0, cCount = 0;
        String str = "Welcome To Java Programming";
        str = str.toLowerCase();
        for(int i = 0; i < str.length(); i++)
        {
            if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' || str.charAt(i) == 'u')
            {
                vCount++;
            }
            else if(str.charAt(i) >= 'a' && str.charAt(i) <= 'z')
            {
                cCount++;
            }
        }
        System.out.println("Number of vowels: " + vCount);
        System.out.println("Number of consonants: " + cCount);
    }
}
```

Output:

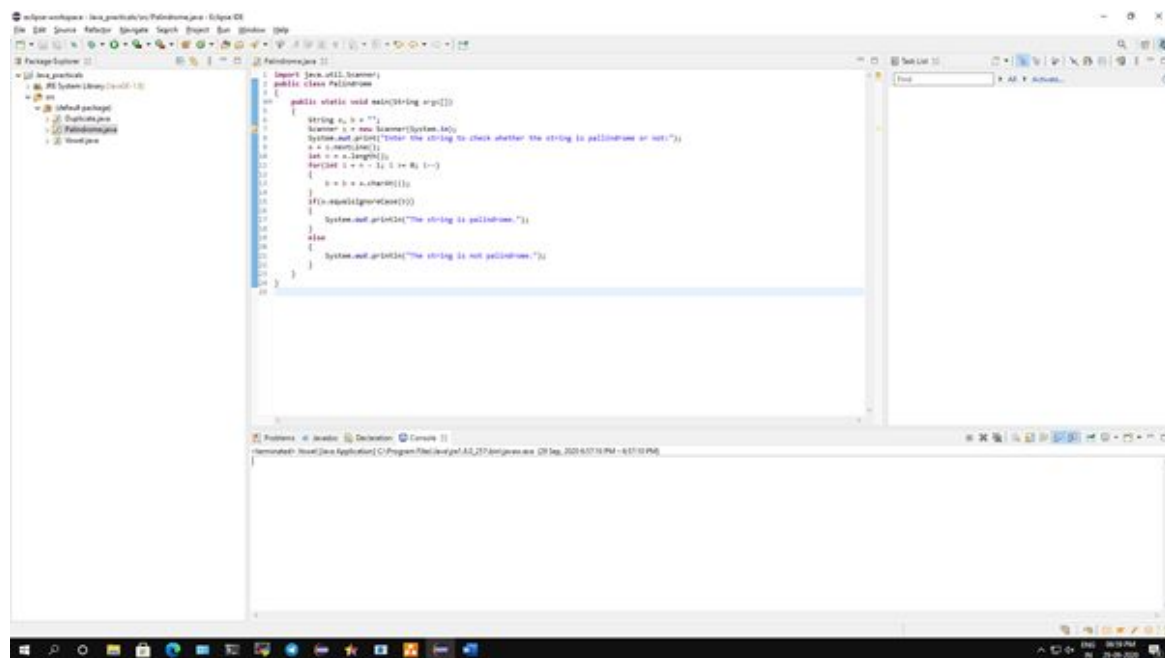
```
<terminated> Vowel [Java Application] !
Number of vowels: 9
Number of consonants: 15
```


C) Program to check whether the string is Palindrome or not.

Aim: Write a java program to check whether the string is Palindrome or not.

Description:

A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam, racecar. There are also numeric palindromes, including date/time stamps using short digits 11/11/11 11:11 and long digits 02/02/2020. Sentence-length palindromes ignore capitalization, punctuation, and word boundaries. We first imported the Scanner utils to load scanner class, then defined the class name which is class Palindrome. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have defined the main method to accept an array of string arguments, we have also defined the scanner class to check if the given number is palindrome or not if the given number is palindrome then the output will execute as number is palindrome otherwise it will show number is not palindrome. Palindrome works on the following algorithm Get the number/strings to check for palindrome. Hold the number/string in a temporary variable. Reverse the number/string. Compare the temporary number/string with reversed number/string. If both numbers/string are the same, print "palindrome". Else print "not a palindrome".



```
import java.util.Scanner;

public class Palindrome {

    public static void main(String args[]) {
        String s;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string to check whether the string is palindrome or not.");
        s = sc.next();
        int n = s.length();
        for(int i = n - 1; i >= 0; i--) {
            if(s.charAt(i) != s.charAt(0)) {
                System.out.println("The string is not palindrome.");
                return;
            }
        }
        System.out.println("The string is palindrome.");
    }
}
```

Conclusion: We have implemented a program to implement a program to check whether the string is Palindrome or not.

Code:

```
import java.util.Scanner;
public class Palindrome
{
    public static void main(String args[])
    {
        String a, b = "";
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the string to check whether the string is pallindrome or not:");
        a = s.nextLine();
        int n = a.length();
        for(int i = n - 1; i >= 0; i--)
        {
            b = b + a.charAt(i);
        }
        if(a.equalsIgnoreCase(b))
        {
            System.out.println("The string is palindrome.");
        }
        else
        {
            System.out.println("The string is not palindrome.");
        }
    }
}
```

Output:

```
Enter the string to check whether the string is pallindrome or not:Java
The string is not palindrome.
```

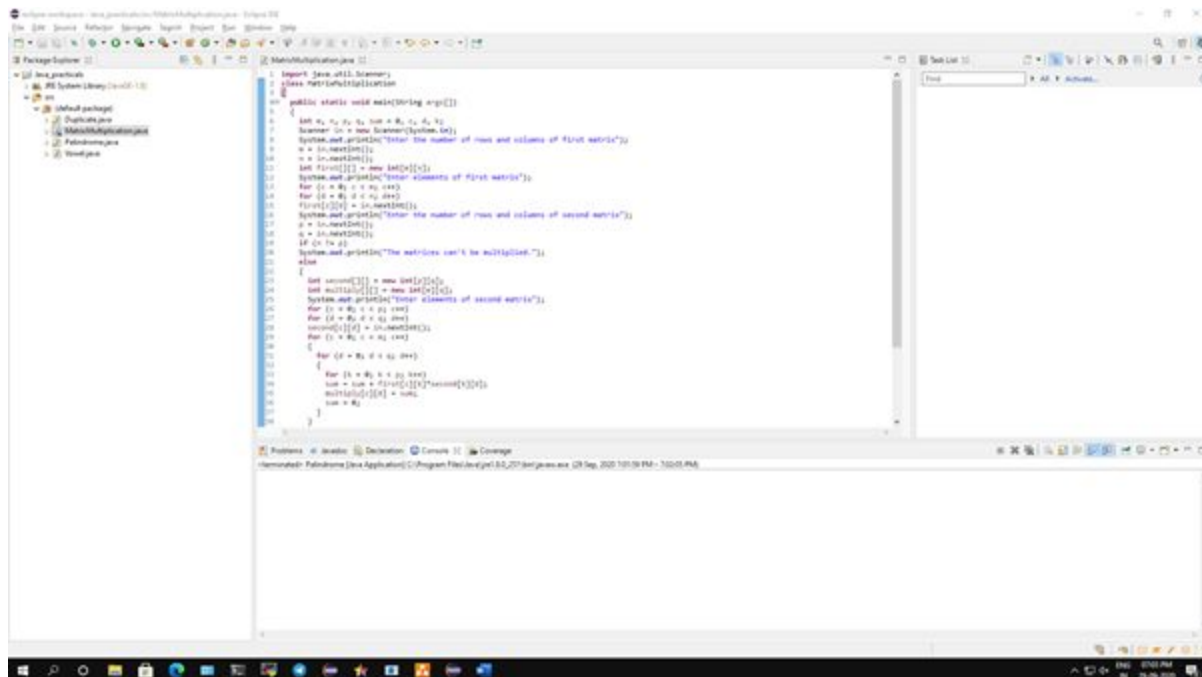
```
Enter the string to check whether the string is pallindrome or not:Javaj
The string is palindrome.
```

D) Program to perform Matrix multiplication.

Aim: Write a java program to perform Matrix multiplication.

Description:

Matrix multiplication is a binary operation that produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the matrix product, has the number of rows of the first and the number of columns of the second matrix. The product of matrices A and B is then denoted simply as AB. Matrix multiplication was first described by the French mathematician Jacques Philippe Marie Binet in 1812, to represent the composition of linear maps that are represented by matrices. Matrix multiplication is thus a basic tool of linear algebra, and as such has numerous applications in many areas of mathematics, as well as in applied mathematics, statistics, physics, economics, and engineering. Computing matrix products is a central operation in all computational applications of linear algebra. We have imported Scanner utils class to load scanner then we have defined the public class which is MatrixMultiplication, then we use int function for input operation of matrix. For matrix multiplication to happen the column of the first matrix should be equal to the row of the second matrix. Once the output is obtained we have successfully compiled the matrix multiplication in java.



```
import java.util.Scanner;
class MatrixMultiplication
{
    public static void main(String args[])
    {
        int n1, n2, r1, c1, r2, c2, i, j, k;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows and columns of first matrix");
        n1 = sc.nextInt();
        c1 = sc.nextInt();
        int first[][] = new int[n1][c1];
        System.out.println("Enter elements of first matrix");
        for (i = 0; i < n1; i++)
            for (j = 0; j < c1; j++)
                first[i][j] = sc.nextInt();
        System.out.println("Enter the number of rows and columns of second matrix");
        n2 = sc.nextInt();
        c2 = sc.nextInt();
        if (c1 != n2)
            System.out.println("The matrices can't be multiplied.");
        else
        {
            int second[][] = new int[n1][c2];
            int multi[][] = new int[n1][c2];
            System.out.println("Enter elements of second matrix");
            for (i = 0; i < n2; i++)
                for (j = 0; j < c2; j++)
                    second[i][j] = sc.nextInt();
            for (i = 0; i < n1; i++)
                for (j = 0; j < c2; j++)
                {
                    for (k = 0; k < c1; k++)
                        multi[i][j] = first[i][k] * second[k][j];
                    multi[i][j] = multi[i][j] + multi[i][j];
                }
            for (i = 0; i < n1; i++)
                for (j = 0; j < c2; j++)
                    multi[i][j] = multi[i][j] % 100;
            System.out.println("The product of two matrices is:");
            for (i = 0; i < n1; i++)
                for (j = 0; j < c2; j++)
                    System.out.print(multi[i][j] + " ");
            System.out.println();
        }
    }
}
```

Conclusion: We have performed a java program on Matrix multiplication.

Code:

```
import java.util.Scanner;
class MatrixMultiplication
{
    public static void main(String args[])
    {
        int m, n, p, q, sum = 0, c, d, k;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of rows and columns of first matrix");
        m = in.nextInt();
        n = in.nextInt();
        int first[][] = new int[m][n];
        System.out.println("Enter elements of first matrix");
        for (c = 0; c < m; c++)
            for (d = 0; d < n; d++)
                first[c][d] = in.nextInt();
        System.out.println("Enter the number of rows and columns of second matrix");
        p = in.nextInt();
        q = in.nextInt();
        if (n != p)
            System.out.println("The matrices can't be multiplied.");
        else
        {
            int second[][] = new int[p][q];
            int multiply[][] = new int[m][q];
            System.out.println("Enter elements of second matrix");
            for (c = 0; c < p; c++)
                for (d = 0; d < q; d++)
                    second[c][d] = in.nextInt();
            for (c = 0; c < m; c++)
            {
                for (d = 0; d < q; d++)
                {
```

```
        for (k = 0; k < p; k++)
            sum = sum + first[c][k]*second[k][d];
        multiply[c][d] = sum;
        sum = 0;
    }
}

System.out.println("Matrix Multiplication are:");
for (c = 0; c < m; c++)
{
    for (d = 0; d < q; d++)
        System.out.print(multiply[c][d]+" ");
    System.out.print("\n");
}
}
```

Output:

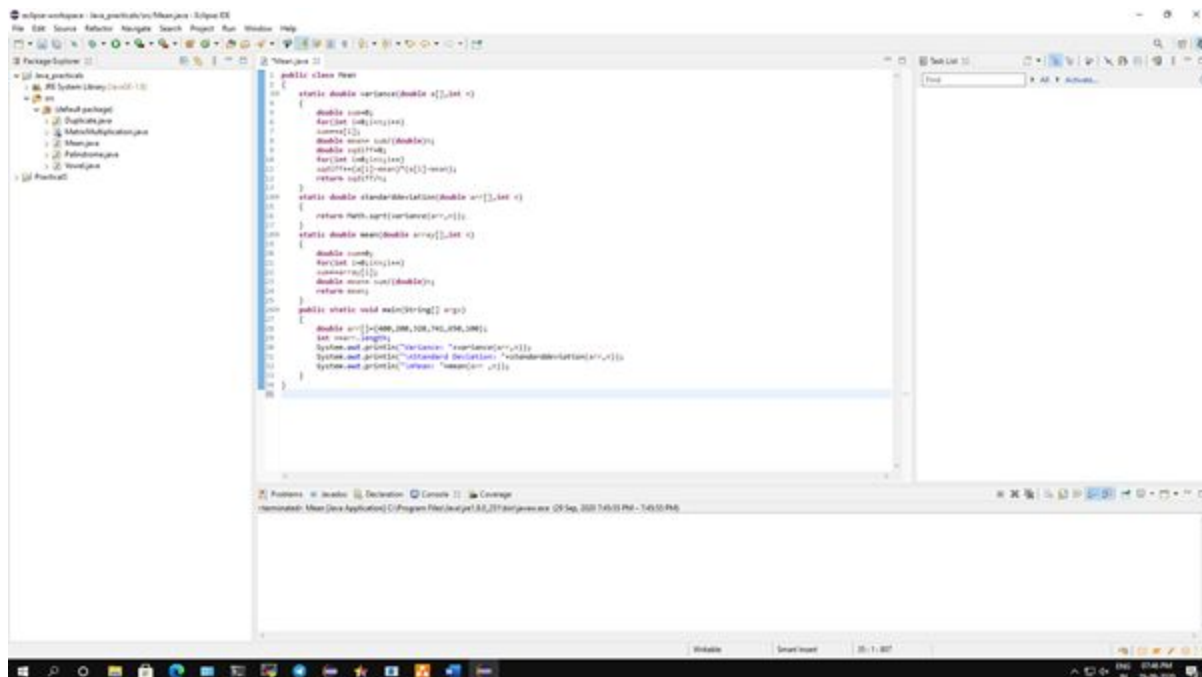
```
Enter the number of rows and columns of first matrix
2
2
Enter elements of first matrix
2
4
6
1
Enter the number of rows and columns of second matrix
2
2
Enter elements of second matrix
3
5
7
0
Matrix Multiplication are:
34      10
25      30
```

E) Program to perform Mean, variance and deviation of array.

Aim: Write a java program to perform Mean, variance and deviation of array.

Description:

The Arithmetic Mean is the average of the numbers: a calculated "central" value of a set of numbers. It is calculated by adding up all the numbers, then dividing by how many numbers there are. The Variance is defined as the average of the squared differences from the Mean. We can calculate the variance by working out the Mean, Then for each number: subtract the Mean and square the result, Then work out the average of those squared differences. The Standard Deviation is a measure of how spread out numbers are. It is defined as σ symbol. It can be calculated by using the square root of the Variance. In Java, a static method is a method that belongs to a class rather than an instance of a class. The method is accessible to every instance of a class, but methods defined in an instance are only able to be accessed by that member of a class. To calculate the mean, we have defined "+mean(arr ,n)" function. To calculate variance we have defined "+variance(arr,n)" and to find deviation we have defined "+standard deviation(arr,n)" functions this will helps us to find all the values of mean, variance and deviation and the output will be generated.



```
1 public class Mean {
2     static double variance(double a[],int n)
3     {
4         double sum=0;
5         for(int i=0;i<n;i++)
6             sum+=a[i]*a[i];
7         double mean=sum/(double)n;
8         double sum1=0;
9         for(int i=0;i<n;i++)
10             sum1+=a[i]*a[i]-mean*a[i];
11         return sum1/n;
12     }
13     static double standardDeviation(double a[],int n)
14     {
15         return Math.sqrt(variance(a,n));
16     }
17     static double mean(double a[],int n)
18     {
19         double sum=0;
20         for(int i=0;i<n;i++)
21             sum+=a[i];
22         double mean=sum/(double)n;
23         return mean;
24     }
25     public static void main(String[] args)
26     {
27         double a[]={400,500,100,140,650,500};
28         int n=a.length;
29         System.out.println("Variance: "+variance(a,n));
30         System.out.println("Standard Deviation: "+standardDeviation(a,n));
31         System.out.println("Mean: "+mean(a,n));
32     }
33 }
```

Conclusion: We have performed a java program on Mean, variance and deviation of array.

Code:

```
public class Mean
{
    static double variance(double a[],int n)
    {
        double sum=0;
        for(int i=0;i<n;i++)
            sum+=a[i];
        double mean= sum/(double)n;
        double sqdiff=0;
        for(int i=0;i<n;i++)
            sqdiff+=(a[i]-mean)*(a[i]-mean);
        return sqdiff/n;
    }
    static double standarddeviation(double arr[],int n)
    {
        return Math.sqrt(variance(arr,n));
    }
    static double mean(double array[],int n)
    {
        double sum=0;
        for(int i=0;i<n;i++)
            sum+=array[i];
    }
}
```

```
        double mean= sum/((double)n);

        return mean;

    }

    public static void main(String[] args)

    {

        double arr[]={400,200,320,741,650,100};

        int n=arr.length;

        System.out.println("Variance: "+variance(arr,n));

        System.out.println("\nStandard Deviation: "+standarddeviation(arr,n));

        System.out.println("\nMean: "+mean(arr ,n));

    }

}
```

Output:

```
Variance: 52526.80555555556
Standard Deviation: 229.1872718009348
Mean: 401.8333333333333
```


Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 03		
Title Of Lab Practical: PROGRAM ON INTERFACE, WRAPPER CLASS AND ABSTRACT DATA				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

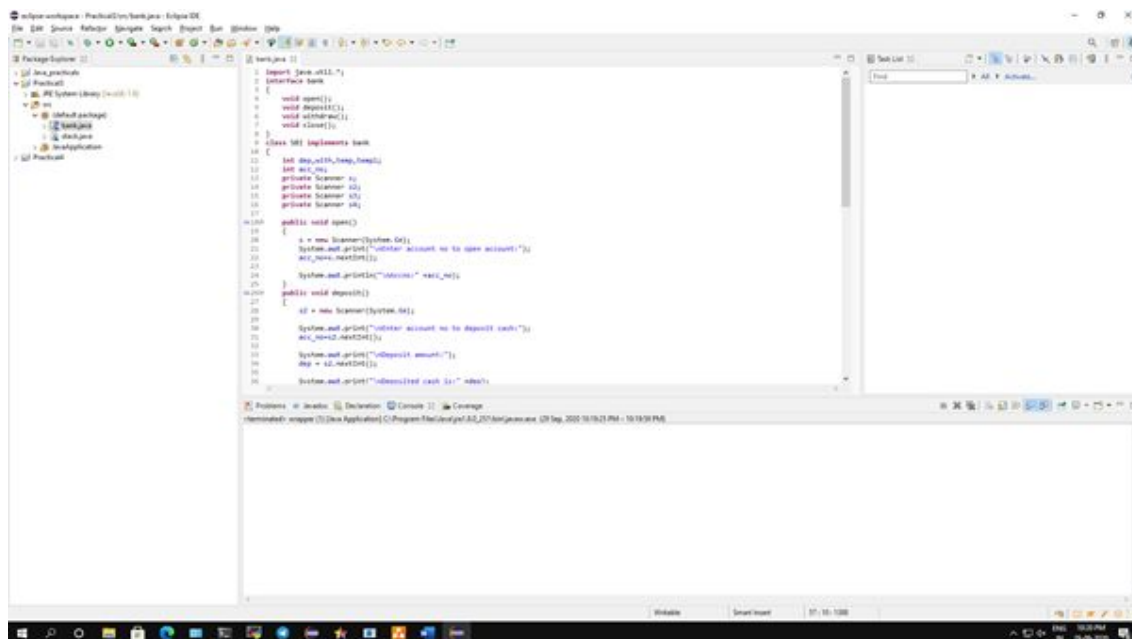
Practical No 3

A) Program to demonstrate interface: Create an interface for Bank with the following operation: -Deposit, Withdraw, Account open, Close account.

Aim: Write a program to demonstrate interface: Create an interface for Bank with following operation: -Deposit, Withdraw, Account open, Close account in java.

Description:

Interface in java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class. In this program we have defined the interface as bank, then we have defined the main class as SBI which implements bank interface then we have defined the private scanner class like S1, S2, S3 and S4. Scanner S1 will ask for the account number. Scanner S2 will ask for the account number in which you want to deposit cash. Scanner S3 will ask to enter the account number in which you want to withdraw cash and Scanner S4 will ask the account number in which you want to close the account. And then we have extended the class interface with bank1.



```
1 import java.util.*;
2 interface bank
3 {
4     void open();
5     void deposit();
6     void withdraw();
7     void close();
8 }
9
10 class SBI implements bank
11 {
12     int acc_no, temp_accno;
13     int acc_no;
14     private Scanner s1;
15     private Scanner s2;
16     private Scanner s3;
17     private Scanner s4;
18
19     public void open()
20     {
21         s1 = new Scanner(System.in);
22         System.out.println("enter account no to open account:");
23         acc_no = s1.nextInt();
24         System.out.println("account no: " + acc_no);
25     }
26
27     public void deposit()
28     {
29         s2 = new Scanner(System.in);
30         System.out.println("enter account no to deposit cash:");
31         acc_no = s2.nextInt();
32         System.out.println("deposit amount:");
33         temp_accno = s2.nextInt();
34         System.out.println("deposited cash is: " + temp_accno);
35     }
36 }
```

Problems | JavaDoc | Declaration | Console | Coverage
C:\Program Files\Eclipse Software\ eclipse.exe [C:\Program Files\Eclipse Software\ eclipse.exe, 2020-10-10-21 PM 10:10:10 PM]

Conclusion: We have written a program to demonstrate interface in java.

Code:

```
import java.util.*;
interface bank
{
    void open();
    void deposit();
    void withdraw();
    void close();
}
class SBI implements bank
{
    int dep,with,temp,temp1;
    int acc_no;
    private Scanner s;
    private Scanner s2;
    private Scanner s3;
    private Scanner s4;

    public void open()
    {
        s = new Scanner(System.in);
        System.out.print("\nEnter account no to open account:");
        acc_no=s.nextInt();

        System.out.println("\nAccount number:" +acc_no);
    }
    public void deposit()
    {
        s2 = new Scanner(System.in);
        System.out.print("\nEnter account no to deposit cash:");
        acc_no=s2.nextInt();
        System.out.print("\nDeposit amount:");
```

```
        dep = s2.nextInt();
        System.out.print("\nDeposited cash is:" +dep);
    }
    public void withdraw()
    {
        s3 = new Scanner(System.in);
        System.out.print("\n\nEnter account no to withdraw cash:");
        acc_no=s3.nextInt();
        System.out.print("\nWithdraw cash:");
        with=s3.nextInt();
        if(with<dep)
        {
            dep=dep-with;
            System.out.println("\nWithdrew cash is:" +with);
            System.out.println("\nBalance cash is:" +dep);
        }
        else
        {
            System.out.print("\nNo funds");
        }
    }
    public void close()
    {
        s4 = new Scanner(System.in);
        System.out.print("\n\nEnter account to close account:");
        acc_no=s4.nextInt();
        System.out.println("\nAccount Closed");
        System.exit(0);
    }
}

class bank1
{
    public static void main(String[] args)
    {
        SBI obj = new SBI();
        obj.open();
    }
}
```

```
        obj.deposit();
        obj.withdraw();
        obj.close();
    }
}
```

Output:

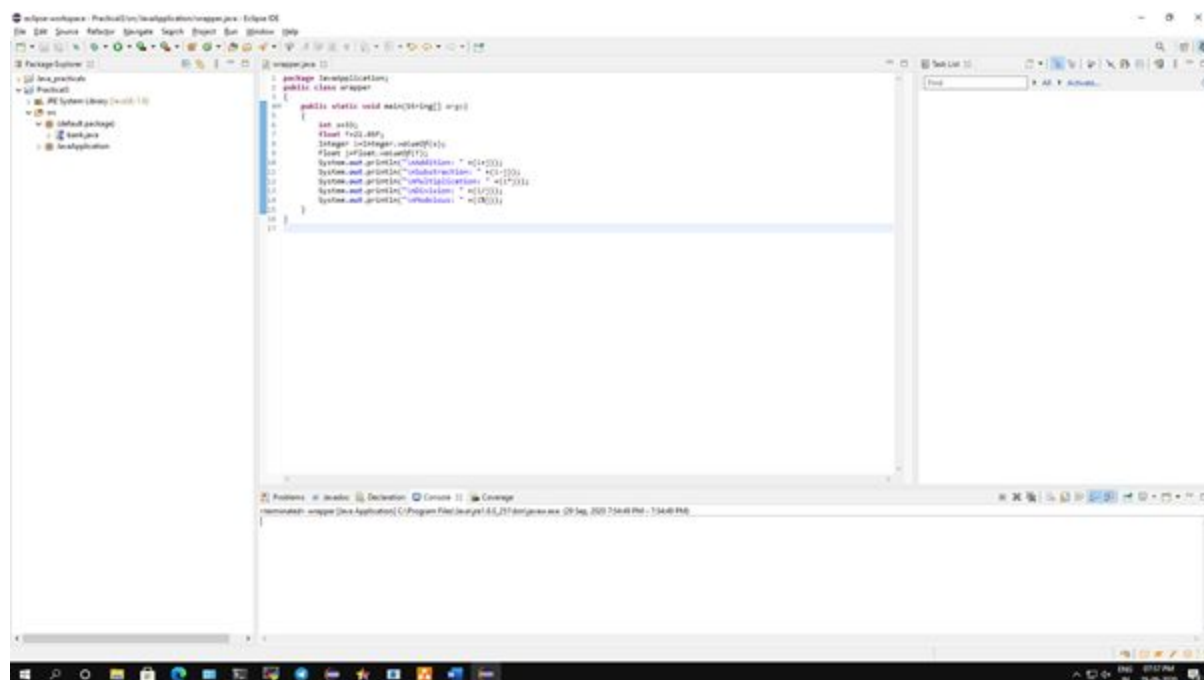
```
Enter account no to open account:33
Account number:33
Enter account no to deposit cash:33
Deposit amount:15000
Deposited cash is:15000
Enter account no to withdraw cash:33
Withdraw cash:12000
Withdrew cash is:12000
Balance cash is:3000
Enter account to close account:33
Account Closed
```

B) Program to implement basic arithmetic operation using Wrapper Class.

Aim: Write a java program to implement basic arithmetic operation using Wrapper Class.

Description:

Wrapper classes are those whose objects wraps a primitive data type within them. In the java.lang package java provides a separate class for each of the primitive data types namely Byte, Character, Double, Integer, Float, Long, Short. At the time of instantiation, these classes accept a primitive datatype directly, or in the form of String. Wrapper classes provide methods to, convert primitive datatypes within them to String objects and, to compare them with other objects etc. Using wrapper classes, you can also add primitive datatypes to various Collection objects such as ArrayList, HashMap etc. You can also pass primitive values over a network using wrapper classes. We have defined JavaApplication as packages and wrapper as the public class. Then we defined main function which will read integer and float. Once it read those functions it will solve the basic arithmetic operation and we will get the output.



Conclusion: We have implemented a program to perform basic arithmetic operation using Wrapper Class.

Code:

```
package JavaApplication;

public class wrapper
{
    public static void main(String[] args)
    {
        int a=33;
        float f=21.45F;
        Integer i=Integer.valueOf(a);
        Float j=Float.valueOf(f);
        System.out.println("\nAddition: " +(i+j));
        System.out.println("\nSubstraction: " +(i-j));
        System.out.println("\nMultiplication: " +(i*j));
        System.out.println("\nDivision: " +(i/j));
        System.out.println("\nModolous: " +(i%j));
    }
}
```

Output:

```
Addition: 54.45
Substraction: 11.549999
Multiplication: 707.85004
Division: 1.5384614
Modolous: 11.549999
```

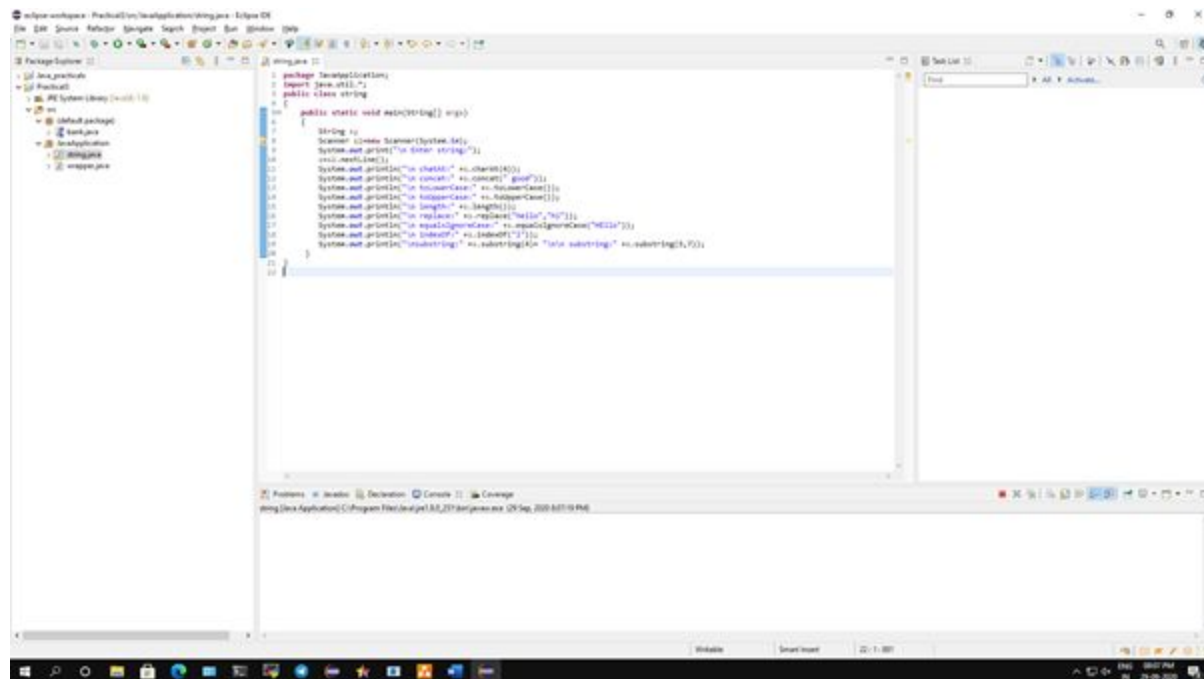
C) Implement a program to demonstrate basic in-built function used in String Class.

Aim: Implement a java program to demonstrate basic in-built function used in String Class.

Description:

Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings. In the given code we have created a package name JavaApplication and public class string then we have defined the main method. We use string s which write the string then we have used scanner function and then we defined different string types such as uppertolower, lowertoupper, concat, length etc. Then we will input the string for e.g. "Hello World" and it will execute the code and return the output of different string types.



```
1 package JavaApplication;
2 import java.util.*;
3 public class string
4 {
5     public static void main(String[] args)
6     {
7         String s;
8         Scanner sc=new Scanner(System.in);
9         System.out.println("Enter string:");
10        s=sc.nextLine();
11        System.out.println("Character at 0th index: "+s.charAt(0));
12        System.out.println("Concatenation of s and 'Hello': "+s.concat("Hello"));
13        System.out.println("String in uppercase: "+s.toUpperCase());
14        System.out.println("String in lowercase: "+s.toLowerCase());
15        System.out.println("Length of string: "+s.length());
16        System.out.println("Replace 'o' with 'X': "+s.replace("o","X"));
17        System.out.println("Replace 'Hello' with 'World': "+s.replace("Hello","World"));
18        System.out.println("Index of 'o': "+s.indexOf("o"));
19        System.out.println("Substring from index 1 to 5: "+s.substring(1,5));
20    }
21 }
```

Conclusion: We have implemented a java program to demonstrate basic in-built function used in String Class.

Code:

```
package JavaApplication;
import java.util.*;
public class string
{
    public static void main(String[] args)
    {
        String s;
        Scanner s1=new Scanner(System.in);
        System.out.print("\n Enter string:");
        s=s1.nextLine();
        System.out.println("\n chatAt:" +s.charAt(4));
        System.out.println("\n concat:" +s.concat(" good"));
        System.out.println("\n toLowerCase:" +s.toLowerCase());
        System.out.println("\n toUpperCase:" +s.toUpperCase());
        System.out.println("\n length:" +s.length());
        System.out.println("\n replace:" +s.replace("hello","hi"));
        System.out.println("\n equalsIgnoreCase:" +s.equalsIgnoreCase("HEllo"));
        System.out.println("\n indexOf:" +s.indexOf("l"));
        System.out.println("\nsubstring:" +s.substring(4)+ "\n\n substring:" +s.substring(3,7));
    }
}
```

Output:

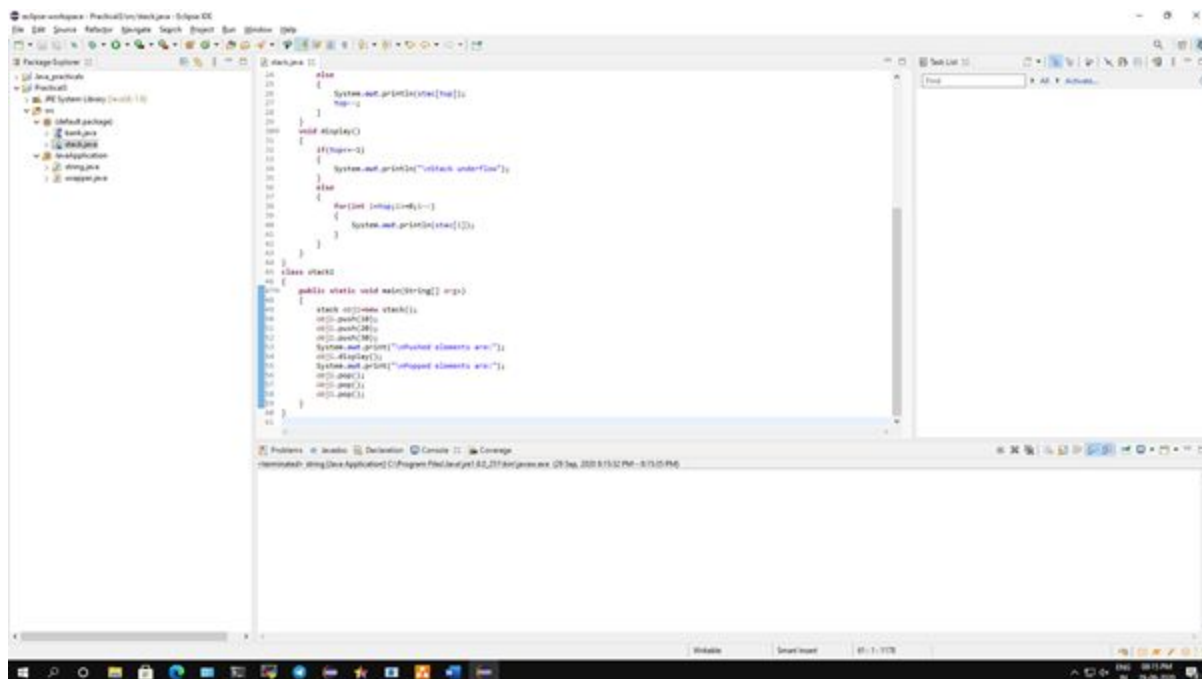
```
Enter string:Welcome to java programming
chatAt:o
concat:Welcome to java programming good
toLowerCase:welcome to java programming
toUpperCase:WELCOME TO JAVA PROGRAMMING
length:27
replace:Welcome to java programming
equalsIgnoreCase:false
indexOf:2
substring:ome to java programming
substring:come
```

D) To implement a program to implement Queue/Stack as ADT in JAVA.

Aim: Implement a java program to implement Queue/Stack as ADT.

Description:

A Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in First Out) or FILO (First in Last Out). There are two basic operations performed in the stack: Push and Pop. Push Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition. Whereas Pop Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition. In the given program we defined the class stack as the main class and then we have defined the pop and push functions. In the next iteration we void push which helps to push the element in stack and pop function helps to pop the element in stack and then we have extended class called as class stack1 which runs the main function to display the output.



```
14  class stack {
15      {
16          System.out.println(stack[top]);
17          top--;
18      }
19      void display()
20      {
21          if(top < 0)
22          {
23              System.out.println("Underflow");
24          }
25          else
26          {
27              for(int i=top; i >= 0; i--)
28              {
29                  System.out.println(stack[i]);
30              }
31          }
32      }
33  }
34  class stack2
35  {
36      public static void main(String[] args)
37      {
38          stack s1=new stack();
39          s1.push(10);
40          s1.push(20);
41          s1.push(30);
42          System.out.println("Pushed elements are:");
43          s1.display();
44          System.out.println("Popped elements are:");
45          s1.pop();
46          s1.pop();
47          s1.pop();
48      }
49  }
```

Conclusion: We have implemented a java program to implement Queue/Stack as ADT.

Code:

```
class stack
{
    int top,n;

    int stac[]=new int[10];

    void push(int n)
    {
        this.n=n;

        if(top>n-1)
        {
            System.out.println("\nStack overflow");
        }

        else
        {
            top++;

            stac[top]=n;
        }
    }

    void pop()
    {
        if(top<=-1)
        {
            System.out.println("\nStack underflow");
        }

        else
        {
            top--;
        }
    }
}
```

```
        System.out.println(stac[top]);

        top--;

    }

}

void display()

{

    if(top<=-1)

    {

        System.out.println("\nStack underflow");

    }

    else

    {

        for(int i=top;i>=0;i--)

        {

            System.out.println(stac[i]);

        }

    }

}

}

class stack1

{

    public static void main(String[] args)

    {

        stack obj1=new stack();

        obj1.push(20);

        obj1.push(30);

        obj1.push(40);
```

```
        System.out.print("\nPushed elements are: ");  
        obj1.display();  
        System.out.print("\nPopped elements are: ");  
        obj1.pop();  
        obj1.pop();  
        obj1.pop();  
    }  
}
```

Output:

```
Pushed elements are: 40  
30  
20  
0  
  
Popped elements are: 40  
30  
20
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 04		
Title Of Lab Practical: PROGRAM ON INHERITANCE AND PACKAGES				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

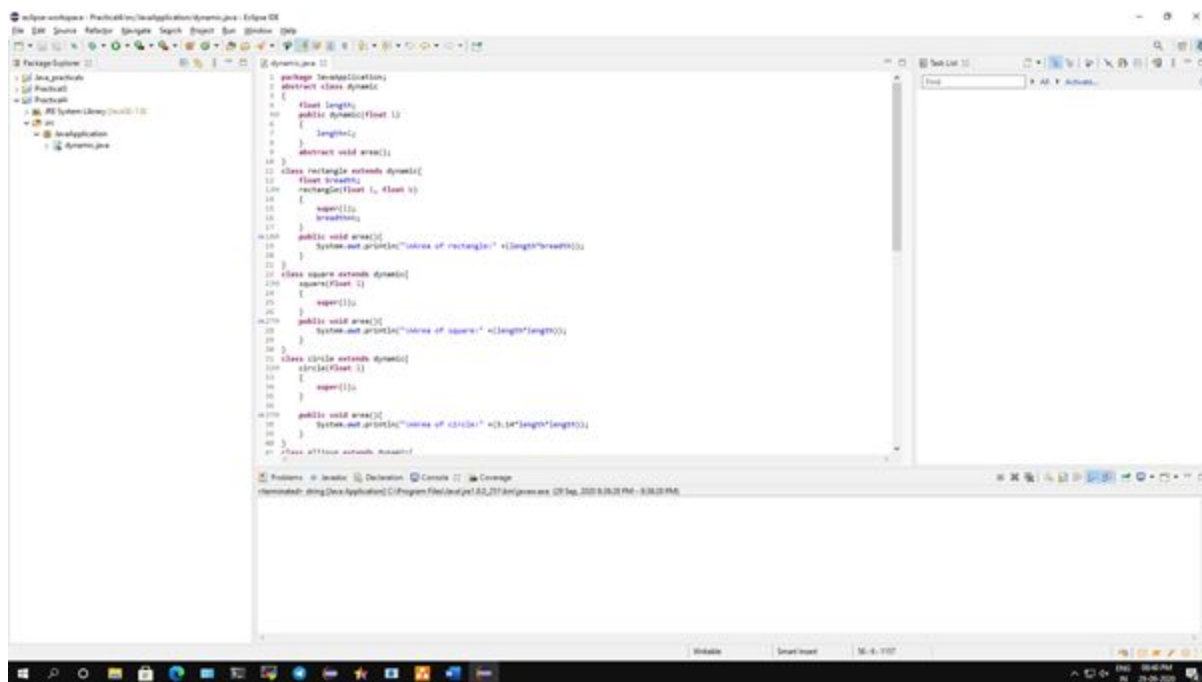
Practical No 4

A) Program to implement Dynamic Method Dispatch in Java using Abstract Class To find the area of various shapes: Rectangle, Circle, Ellipse, Square and Triangle.

Aim: Write a program to implement Dynamic Method Dispatch in Java using Abstract Class.

Description:

Method overriding is one of the ways in which Java supports Runtime Polymorphism. Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time. When an overridden method is called through a superclass reference, Java determines which version(superclass/subclasses) of that method is to be executed based upon the type of the object being referred to at the time the call occurs. Thus, this determination is made at run time. In this program we have defined the package as JavaApplication and abstract class as dynamic. Then we use float values to find the various shape values. We use length as l to define in class. Class rectangle is extended with dynamic class, then we used void area to get the value of the rectangle same goes for all other shape values. After all the values we have defined an abstract class in which the main function starts to get the desired output.



```
1 package JavaApplication;
2 abstract class dynamic {
3     float length;
4     public dynamic(float l) {
5         length=l;
6     }
7     abstract void area();
8 }
9
10 class rectangle extends dynamic {
11     float breadth;
12     rectangle(float l, float b) {
13         super(l);
14         breadth=b;
15     }
16     public void area() {
17         System.out.println("Area of rectangle: " + length*breadth);
18     }
19 }
20
21 class square extends dynamic {
22     square(float l) {
23         super(l);
24     }
25     public void area() {
26         System.out.println("Area of square: " + length*length);
27     }
28 }
29
30 class circle extends dynamic {
31     circle(float l) {
32         super(l);
33     }
34     public void area() {
35         System.out.println("Area of circle: " + 3.14*length*length);
36     }
37 }
38
39 class triangle extends dynamic {
40     triangle(float l, float b) {
41         super(l);
42         breadth=b;
43     }
44     public void area() {
45         System.out.println("Area of triangle: " + 0.5*length*breadth);
46     }
47 }
```

Conclusion: We have written a program to implement Dynamic Method Dispatch in Java using Abstract Class.

Code:

```
package JavaApplication;

abstract class dynamic
{
    float length;

    public dynamic(float l)
    {
        length=l;
    }

    abstract void area();
}

class rectangle extends dynamic{
    float breadth;

    rectangle(float l, float b)
    {
        super(l);
        breadth=b;
    }

    public void area(){
        System.out.println("\nArea of rectangle:" +(length*breadth));
    }
}
```



```
class square extends dynamic{

    square(float l)

    {

        super(l);

    }

    public void area(){

        System.out.println("\nArea of square:" +(length*length));

    }

}

class circle extends dynamic{

    circle(float l)

    {

        super(l);

    }

    public void area(){

        System.out.println("\nArea of circle:" +(3.14*length*length));

    }

}

class ellipse extends dynamic{

    float b;

    ellipse(float l, float b)

    {

        super(l);

        this.b=b;

    }

}
```

```
    }

    public void area(){

        System.out.println("\nArea of ellipse:" +(3.14*length*b));

    }

}
```

```
class triangle extends dynamic{

    float h;

    triangle(float l, float h)

    {

        super(l);

        this.h=h;

    }


    public void area()

    {

        System.out.println("\nArea of triangle:" +(length*h/2));

    }

}
```

```
class abstract_dispatch

{

    public static void main(String[] args)

    { rectangle obj=new rectangle(10.1F,20.2F);

        square objsquare=new square(40.5F);

        circle objcircle=new circle(15.5F);

        ellipse objellipse=new ellipse(55F,16.4F);

    }
```

```
triangle objtriangle=new triangle(80,44.4F);

dynamic ref;

ref=obj;

ref.area();

ref=objsquare;

ref.area();

ref=objcircle;

ref.area();

ref=objellipse;

ref.area();

ref=objtriangle;

ref.area();

}

}
```

Output:

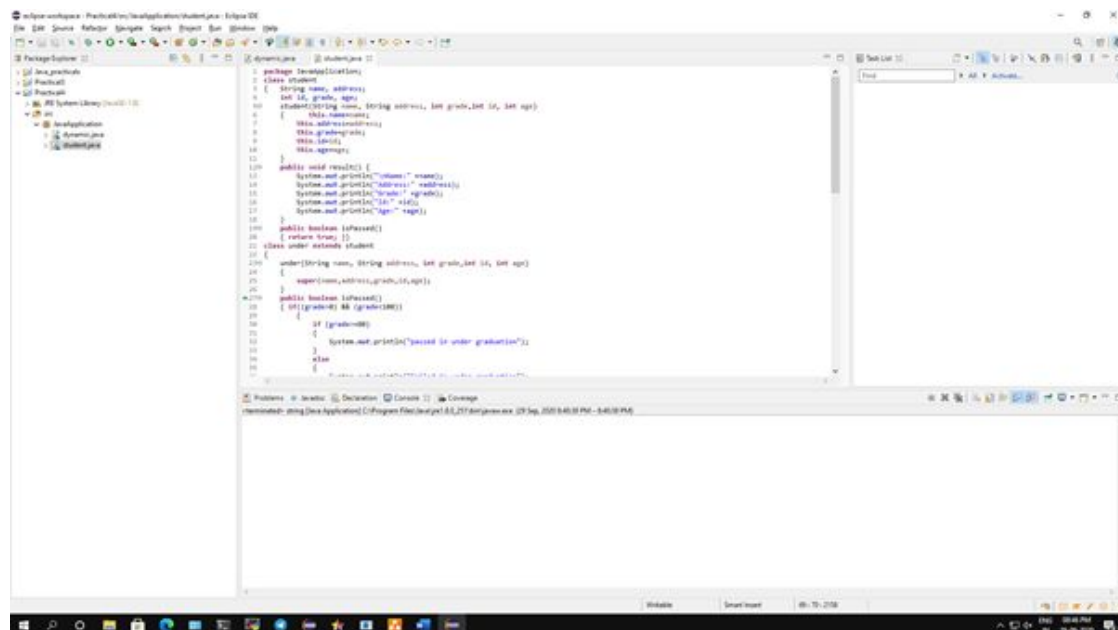
```
Area of rectangle:204.02002
Area of square:1640.25
Area of circle:754.385
Area of ellipse:2832.2799341201785
Area of triangle:1776.0
```

B) Create Super Class Student and two subclasses of it, Graduate and Under Graduate. The members of the Student are name, id, grade, age and address and at least one method: boolean method IsPassed which takes in the parameter integer grade (0-100). The two sub classes override the method, for UG its 70% for passing and for G its 80% as passing grade.

Aim: Write a java program to Create Super Class Students and two subclasses of it, Graduate and UnderGraduate.

Description:

The **super** keyword in Java is a reference variable which is used to refer to an immediate parent class object. Whenever you create an instance of a subclass, an instance of the parent class is created implicitly which is referred to by a super reference variable. Super can be used to refer to immediate parent class instance variables. Super can be used to invoke immediate parent class methods. super() can be used to invoke immediate parent class constructor. In this program we have to create the super class of students with two subclasses for graduate and undergraduate. We have defined the class student followed by string name and address. Then we used the integer int which defines id, grade and age. Then the void result will display the result output followed by extended class which is defined as passing grade and other extended class has failed grade. The output will display if a student scores more than 75% else it will fail if less than 75%.



```

package com.ninad;

import java.util.Scanner;

class Student {
    String name, address;
    int id, grade, age;

    Student(String name, String address, int grade, int id, int age) {
        this.name = name;
        this.address = address;
        this.grade = grade;
        this.id = id;
        this.age = age;
    }

    void result() {
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Grade: " + grade);
        System.out.println("Id: " + id);
        System.out.println("Age: " + age);
    }

    boolean isPassed() {
        return true;
    }
}

class Graduate extends Student {
    Graduate(String name, String address, int grade, int id, int age) {
        super(name, address, grade, id, age);
    }

    boolean isPassed() {
        if (grade >= 80) {
            System.out.println("Passed (Graduate)");
        } else {
            System.out.println("Failed (Graduate)");
        }
    }
}

class UnderGraduate extends Student {
    UnderGraduate(String name, String address, int grade, int id, int age) {
        super(name, address, grade, id, age);
    }

    boolean isPassed() {
        if (grade >= 70) {
            System.out.println("Passed (UnderGraduate)");
        } else {
            System.out.println("Failed (UnderGraduate)");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Name:");
        String name = sc.nextLine();
        System.out.println("Enter Address:");
        String address = sc.nextLine();
        System.out.println("Enter Grade:");
        int grade = sc.nextInt();
        System.out.println("Enter Id:");
        int id = sc.nextInt();
        System.out.println("Enter Age:");
        int age = sc.nextInt();

        Student student = new Student(name, address, grade, id, age);
        student.result();

        Graduate graduate = new Graduate(name, address, grade, id, age);
        graduate.isPassed();

        UnderGraduate underGraduate = new UnderGraduate(name, address, grade, id, age);
        underGraduate.isPassed();
    }
}

```

Conclusion: We have written a program to Create Super Class Students and two subclasses of it, Graduate and UnderGraduate.

Code:

```
package JavaApplication;
class student
{
    String name, address;
    int id, grade, age;
    student(String name, String address, int grade,int id, int age)
    {
        this.name=name;
        this.address=address;
        this.grade=grade;
        this.id=id;
        this.age=age;
    }
    public void result() {
        System.out.println("\nName:" +name);
        System.out.println("Address:" +address);
        System.out.println("Grade:" +grade);
        System.out.println("Id:" +id);
        System.out.println("Age:" +age);
    }
    public boolean isPassed()
    { return true; }}
class under extends student
{
    under(String name, String address, int grade,int id, int age)
    {
        super(name,address,grade,id,age);
    }
    public boolean isPassed()
    { if((grade>0) && (grade<100))
    {
        if (grade>=80)
        {
            System.out.println("passed in under graduation");
        }
        else
        {
            System.out.println("Failed in under graduation");
            System.exit(0);
        }
    }
    else
    {
        System.out.println("Grade should be between 0-100");
    }
}
```

```
        }return true; }
public void result()
{
    System.out.println("\n\nName:" +name);
    System.out.println("Address:" +address);
    System.out.println("Grade:" +grade);
    System.out.println("Id:" +id);
    System.out.println("Age:" +age); }}
class graduate extends under
{
    graduate(String name, String address, int grade,int id, int age)
    {
        super(name,address,grade,id,age);
    }
    public boolean isPassed() {
        if((grade>0) && (grade<100))
        { if (grade>=70){
            System.out.println("passed in under graduation");
        }
        else
        {
            System.out.println("Failed in under graduation");
        } }
        else
        {
            System.out.println("Grade should be between 0-100");
        } return true; }
    public void result() { System.out.println("\n\nName:" +name);
        System.out.println("Address:" +address);
        System.out.println("Grade:" +grade);
        System.out.println("Id:" +id);
        System.out.println("Age:" +age);}}
class student1
{
    public static void main(String[] args)
    {
        student obistud=new student("Jon","Chembur",90,33,22);
        student s2= new under("Kiev","Kurla",84,12,23);
        student s1= new graduate("Mukul","Bandra",85,56,22);
        obistud.result();
        s2.result();
        s2.isPassed();
        s1.result();
        s1.isPassed();
    }
}
```

Output:**Failed in graduate**

```
Name:Jon  
Address:Chembur  
Grade:90  
Id:33  
Age:22
```

```
Name:Kiev  
Address:Kurla  
Grade:70  
Id:12  
Age:23  
Failed in under graduation
```

Passed in under graduate

```
Name:Jon  
Address:Chembur  
Grade:90  
Id:33  
Age:22
```

```
Name:Kiev  
Address:Kurla  
Grade:84  
Id:12  
Age:23  
passed in under graduation
```

```
Name:Mukul  
Address:Bandra  
Grade:85  
Id:56  
Age:22  
passed in under graduation
```

C) Sample Program to demonstrate Access Specifier in Packages.

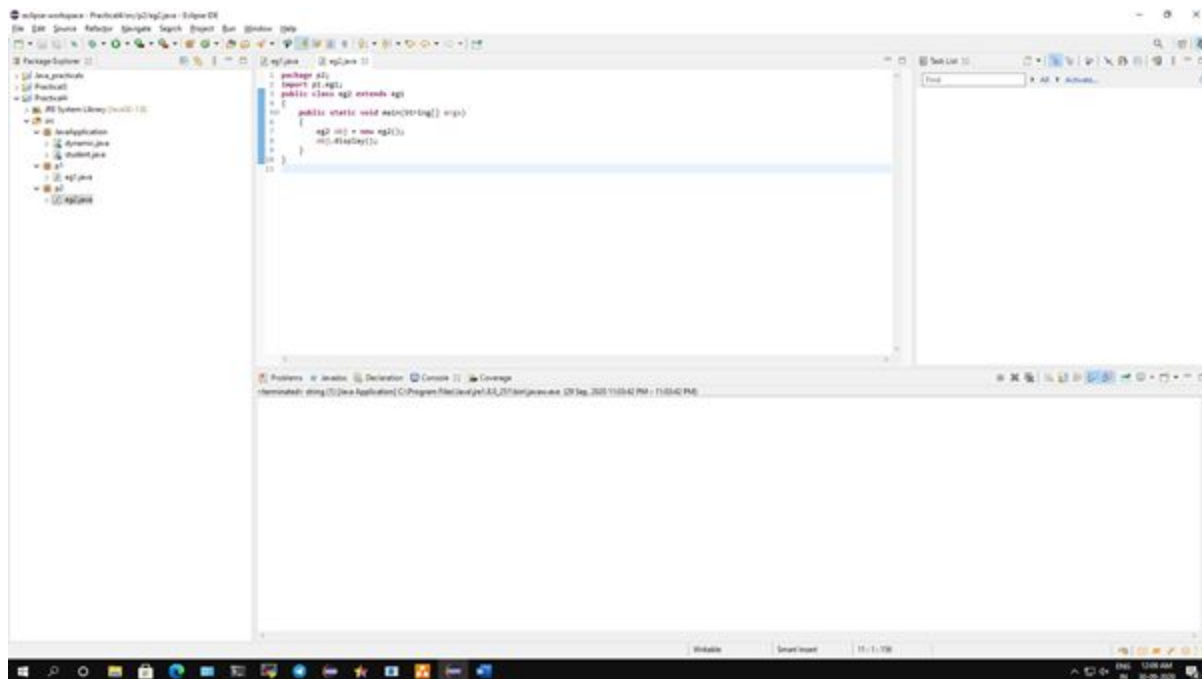
Aim: Write a sample java program to demonstrate Access Specifier in Packages.

Description:

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and classes by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.



Conclusion: We have demonstrated a sample program on Access Specifier in Packages.

A) Protected Modifier

Code:

EG1.java

```
package p1;
public class eg1
{
    protected void display()
    {
        System.out.println("Welcome to java programming");
    }
}
```

EG2.java

```
package p2;
import p1.eg1;
public class eg2 extends eg1
{
    public static void main(String[] args)
    {
        eg2 obj = new eg2();
        obj.display();
    }
}
```

Output:

```
Welcome to java programming
```

B) Public Modifier**Code:****EG1.java**

```
package p1;
public class eg1
{
    public void display()
    {
        System.out.println("Public access modifier is used");
    }
}
```

EG2.java

```
package p2;
import p1.eg1;
public class eg2 extends eg1
{
    public static void main(String[] args)
    {
        eg2 obj = new eg2();
        obj.display();
    }
}
```

Output:

```
Public access modifier is used
```

C) Private Modifier

Code:

EG1.java

```
package p1;
public class eg1
{
    private void display()
    {
        System.out.println("Private access modifier is used");
    }
}
```

EG2.java

```
package p2;
import p1.eg1;
public class eg2 extends eg1
{
    public static void main(String[] args)
    {
        eg2 obj = new eg2();
        obj.display();
    }
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The method display() from the type eg1 is not visible

    at p2.eg2.main(eg2.java:8)
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 05		
Title Of Lab Practical: GENERICS, COLLECTIONS AND LAMBDA EXPRESSION GENERICS				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

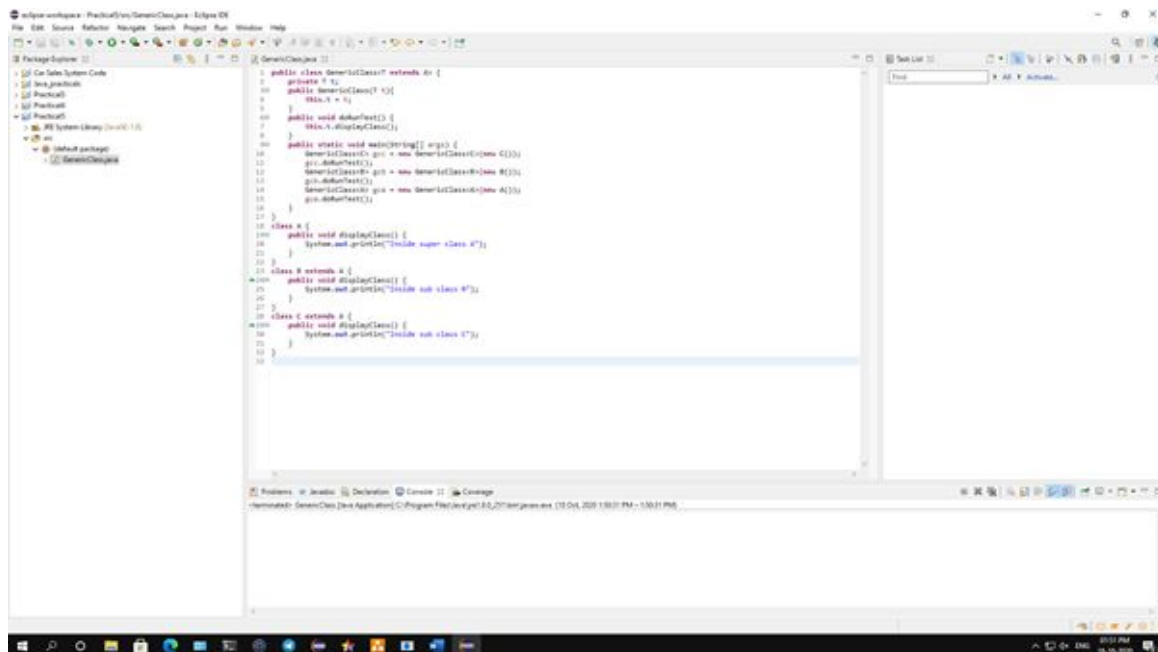
Practical No 5

A) Implement bounded types (extends superclass) with generics.

Aim: Write a program to implement bounded types (extends superclass) with generics.

Description:

Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively. Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time. Using the Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements. We have defined generic class T which extends A and then we defined the private keyword T with public generic class. The t.display class will display the output with other generic classes. Then we have defined class A which will show that class A is inside the sub class A. Then class B has been extended with class A which shows class A is inside of sub class B and for class C the output is repeated same as other classes.



```
1 public class GenericClassT extends A {
2     private T t;
3     public GenericClassT(T x) {
4         this.t = x;
5     }
6     public void displayClass() {
7         this.t.displayClass();
8     }
9     public static void main(String[] args) {
10        GenericClassT gtc = new GenericClassT(new C());
11        gtc.displayClass();
12        GenericClassT gtc2 = new GenericClassT(new B());
13        gtc2.displayClass();
14        GenericClassT gtc3 = new GenericClassT(new A());
15        gtc3.displayClass();
16    }
17 }
18 class A {
19     public void displayClass() {
20         System.out.println("Inside super class A");
21     }
22 }
23 class B extends A {
24     public void displayClass() {
25         System.out.println("Inside sub class B");
26     }
27 }
28 class C extends A {
29     public void displayClass() {
30         System.out.println("Inside sub class C");
31     }
32 }
```

Conclusion: We have written a program to implement bounded types (extends superclass) with generics.

Code:

```
public class GenericClass<T extends A> {
    private T t;
    public GenericClass(T t){
        this.t = t;
    }
    public void doRunTest() {
        this.t.displayClass();
    }
    public static void main(String[] args) {
        GenericClass<C> qcc = new GenericClass<C>(new C());
        qcc.doRunTest();
        GenericClass<B> qcb = new GenericClass<B>(new B());
        qcb.doRunTest();
        GenericClass<A> qca = new GenericClass<A>(new A());
        qca.doRunTest();
    }
}

class A {
    public void displayClass() {
        System.out.println("Inside super class A");
    }
}

class B extends A {
    public void displayClass() {
        System.out.println("Inside sub class B");
    }
}

class C extends A {
    public void displayClass() {
        System.out.println("Inside sub class C");
    }
}
```

Output:

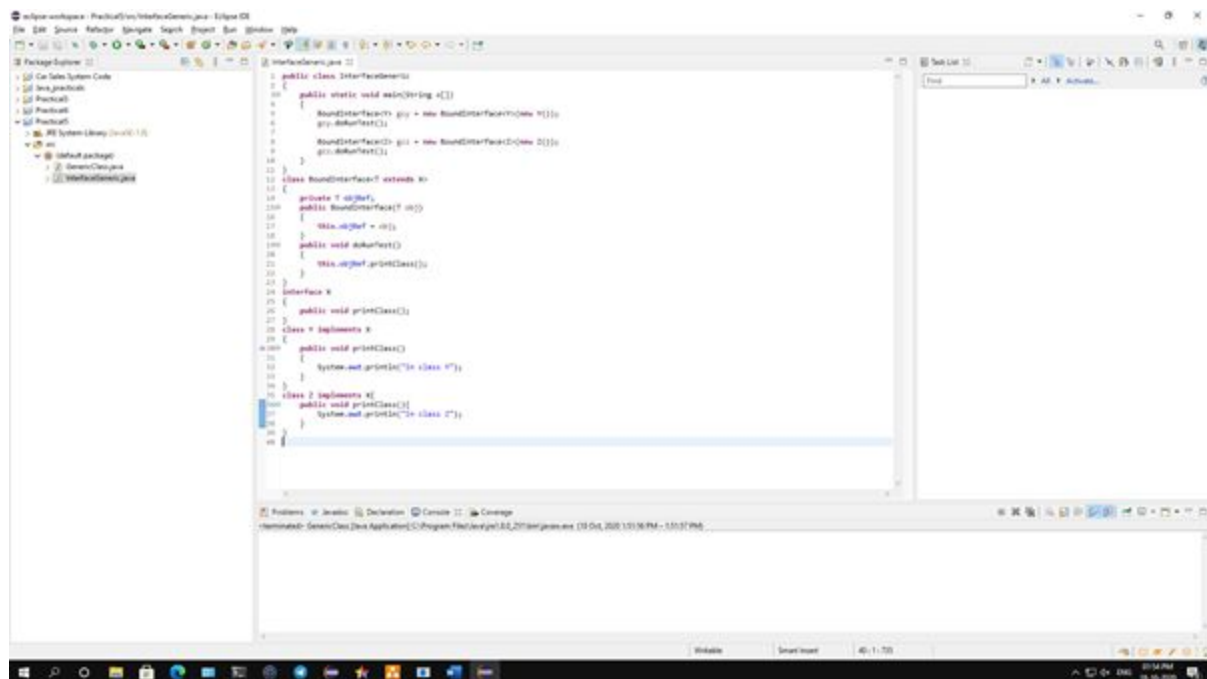
```
Inside sub class C
Inside sub class B
Inside super class A
```

B) Implement bounded types (implements an interface) with generics.

Aim: Write a java program to implement bounded types (implements an interface) with generics.

Description:

There may be times when you'll want to restrict the kinds of types that are allowed to be passed to a type parameter. For example, a method that operates on numbers might only want to accept instances of Number or its subclasses. This is what bounded type parameters are for. To declare a bounded type parameter, list the type parameter's name, followed by the extends keyword, followed by its upper bound. In this program we have defined the bound interface as Y and Z. Then we use class bounds that extend T and X with private obj T with the public BoundInterface class. Then we have defined X interface with class Y that implements class X so it will print that X is in class Y and then we defined class Z which implements X that will show output as class X is in class Z.



```
1 public class InterfaceGenerics {
2     public static void main(String s[]) {
3         BoundInterface<Y> y = new BoundInterface<>(<Y> Y());
4         y.demonTest();
5
6         BoundInterface<Z> z = new BoundInterface<>(<Z> Z());
7         z.demonTest();
8     }
9 }
10
11 class BoundInterface<T extends X> {
12     private T objT;
13     public BoundInterface(T obj) {
14         this.objT = obj;
15     }
16     public void demonTest() {
17         this.objT.printClass();
18     }
19 }
20
21 interface X {
22     public void printClass();
23 }
24
25 class Y implements X {
26     public void printClass() {
27         System.out.println("In class Y");
28     }
29 }
30
31 class Z implements X {
32     public void printClass() {
33         System.out.println("In class Z");
34     }
35 }
```

Conclusion: We have written a program to implement bounded types (implements an interface) with generics.

Code:

```
public class InterfaceGeneric
{
    public static void main(String a[])
    {
        BoundInterface<Y> acv = new BoundInterface<Y>(new Y());
        acv.doRunTest();
        BoundInterface<Z> acz = new BoundInterface<Z>(new Z());
        acz.doRunTest();
    }
}

class BoundInterface<T extends X> {
    private T obiRef;
    public BoundInterface(T obi) {
        this.obiRef = obi;
    }
    public void doRunTest() {
        this.obiRef.printClass();
    }
}

interface X
{
    public void printClass();
}

class Y implements X
{
    public void printClass()
    {
        System.out.println("In class Y");
    }
}

class Z implements X{
    public void printClass(){
        System.out.println("In class Z");
    }
}
```

Output:

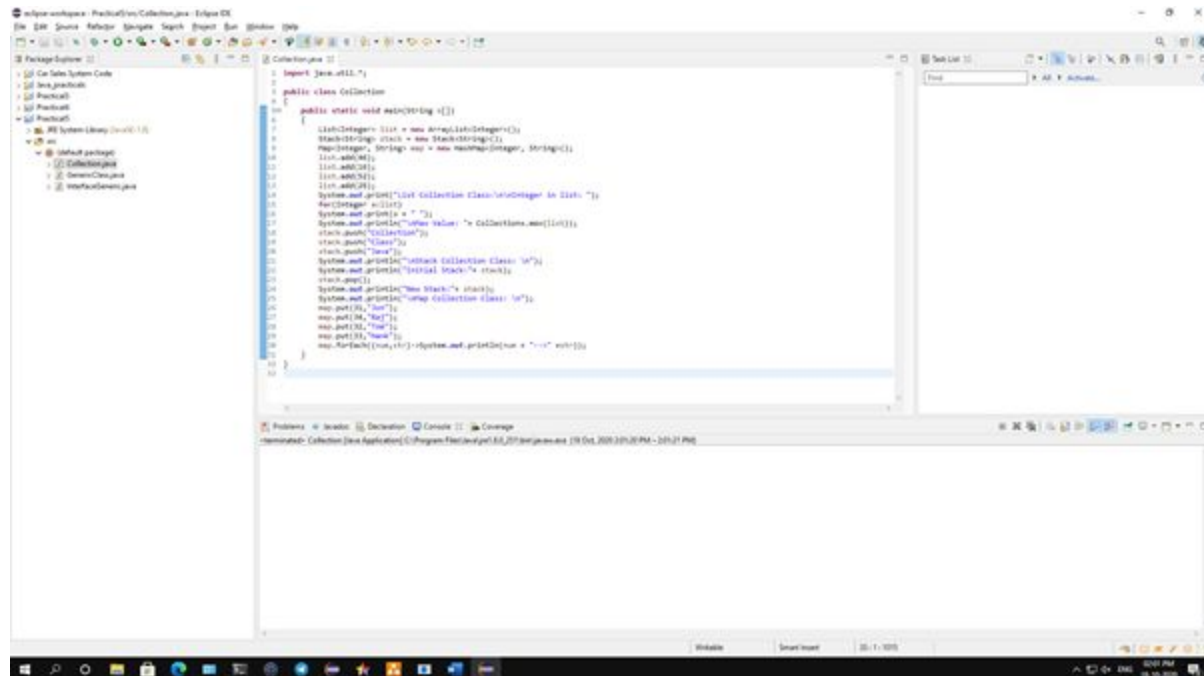
```
In class Y
In class Z
```


C) Implement any three collection classes.

Aim: Write a java program to implement any three collection classes.

Description:

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion. Java Collection means a single unit of objects. The Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet). A framework is a set of classes and interfaces which provide a ready-made architecture. In order to implement a new feature or a class, there is no need to define a framework. However, an optimal object-oriented design always includes a framework with a collection of classes such that all the classes perform the same kind of task. Consistent API, reduces programming effort and Increases program speed and quality are the advantages of collection frameworks. In this program we use Array as a list integer to find the list of integers with collection classes. We have a stack function to list the max collection of classes and then we use a pop function to pop out the list if the stack is full, then we get the map of the collection class output.



```
import java.util.*;

public class Collection {

    public static void main(String s[]) {
        List<Integer> list = new ArrayList<Integer>();
        Stack<String> stack = new Stack<String>();
        Map<Integer, String> map = new HashMap<Integer, String>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        list.add(50);
        System.out.println("List Collection Class/Integer in List: ");
        for(Integer i: list)
            System.out.print(i + " ");
        System.out.println("\nCollection name: " + Collections.name(list));
        stack.push("collection");
        stack.push("class");
        stack.push("test");
        System.out.println("Stack Collection Class: ");
        System.out.println("Initial Stack: " + stack);
        stack.pop();
        System.out.println("New Stack: " + stack);
        System.out.println("Map Collection Class: ");
        map.put(10, "test");
        map.put(20, "map");
        map.put(30, "map");
        map.put(40, "map");
        map.put(50, "map");
        map.forEach((key, value) -> System.out.println(key + " " + value));
    }
}
```

Conclusion: We have written a java program to implement any three collection classes.

Code:

```
import java.util.*;

public class Collection
{
    public static void main(String x[])
    {
        List<Integer> list = new ArrayList<Integer>();
        Stack<String> stack = new Stack<String>();
        Map<Integer, String> map = new HashMap<Integer, String>();
        list.add(46);
        list.add(16);
        list.add(52);
        list.add(25);
        System.out.println("List Collection Class:\n\nInteger in list: ");
        for(Integer a:list)
        System.out.print(a + " ");
        System.out.println("\nMax Value: "+ Collections.max(list));
        stack.push("Collection");
        stack.push("Class");
        stack.push("Java");
        System.out.println("\nStack Collection Class: \n");
        System.out.println("Initial Stack:"+ stack);
        stack.pop();
        System.out.println("New Stack:"+ stack);
        System.out.println("\nMap Collection Class: \n");
        map.put(31,"Jon");
```

```
        map.put(34,"Raj");

        map.put(32,"Tom");

        map.put(33,"Hank");

        map.forEach((num,str)->System.out.println(num + "-->" +str));

    }

}
```

Output:

```
List Collection Class:

Integer in list: 46 16 52 25
Max Value: 52

Stack Collection Class:

Initial Stack:[Collection, Class, Java]
New Stack:[Collection, Class]

Map Collection Class:

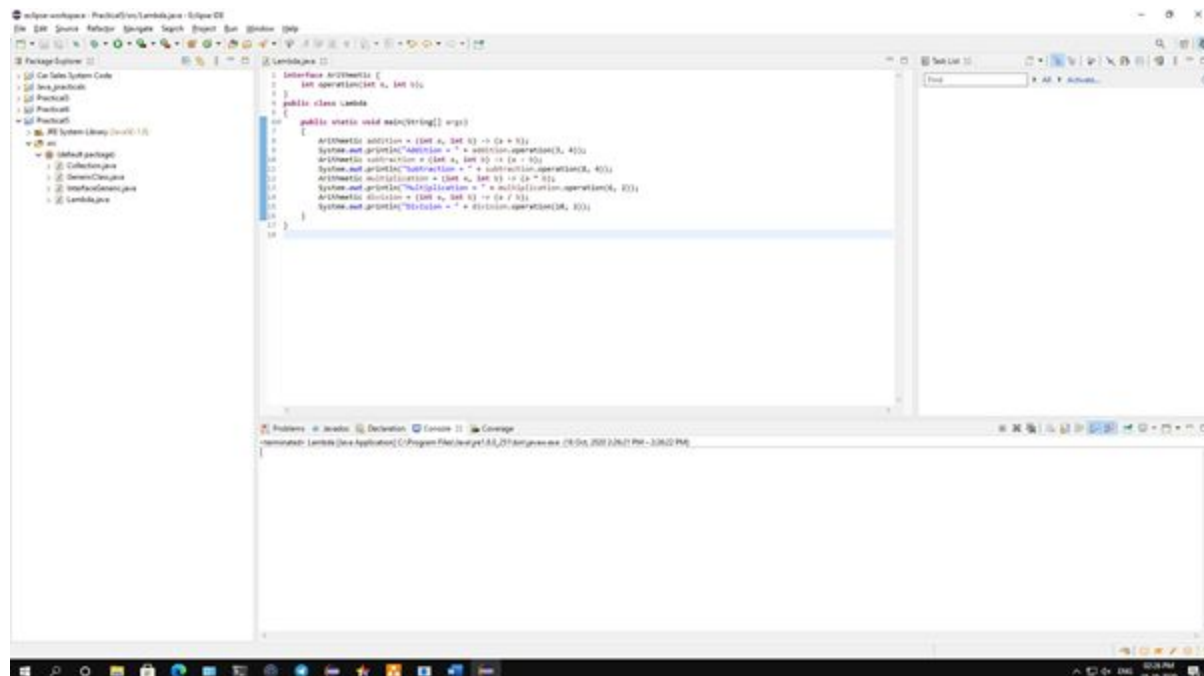
32-->Tom
33-->Hank
34-->Raj
31-->Jon
```

D) Perform addition, subtraction, multiplication as well as division using Lambda Expression.

Aim: Write a java program to perform addition, subtraction, multiplication as well as division using Lambda Expression.

Description:

A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method. Lambda expressions are usually passed as parameters to a function. Lambda expressions can be stored in variables if the variable's type is an interface which has only one method. The lambda expression should have the same number of parameters and the same return type as that method. Java has many of these kinds of interfaces built in, such as the Consumer interface (found in the java.util package) used by lists. In this program we first defined the interface as arithmetic to find arithmetic operations. Then we have defined Lambda as a public class then we use an arithmetic interface to define addition, subtraction, multiplication and division operation. We will get the output once the program is done.



```
1 interface Arithmetic {
2     int operation(int x, int y);
3 }
4 public class Lambda {
5 }
6 public static void main(String[] args) {
7     Arithmetic addition = (x, y) -> x + y;
8     System.out.println("Addition = " + addition.operation(3, 4));
9     Arithmetic subtraction = (x, y) -> x - y;
10    System.out.println("Subtraction = " + subtraction.operation(8, 4));
11    Arithmetic multiplication = (x, y) -> x * y;
12    System.out.println("Multiplication = " + multiplication.operation(4, 3));
13    Arithmetic division = (x, y) -> x / y;
14    System.out.println("Division = " + division.operation(8, 3));
15 }
```

Conclusion: We have written a java program to perform addition, subtraction, multiplication as well as division using Lambda Expression.

Code:

```
interface Arithmetic {  
    int operation(int a, int b);  
}  
  
public class Lambda  
{  
    public static void main(String[] args)  
    {  
        Arithmetic addition = (int a, int b) -> (a + b);  
        System.out.println("Addition = " + addition.operation(3, 4));  
        Arithmetic subtraction = (int a, int b) -> (a - b);  
        System.out.println("Subtraction = " + subtraction.operation(8, 4));  
        Arithmetic multiplication = (int a, int b) -> (a * b);  
        System.out.println("Multiplication = " + multiplication.operation(6, 2));  
        Arithmetic division = (int a, int b) -> (a / b);  
        System.out.println("Division = " + division.operation(10, 2));  
    }  
}
```

Output:

```
Addition = 7  
Subtraction = 4  
Multiplication = 12  
Division = 5
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 06		
Title Of Lab Practical: EXCEPTION HANDLING				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

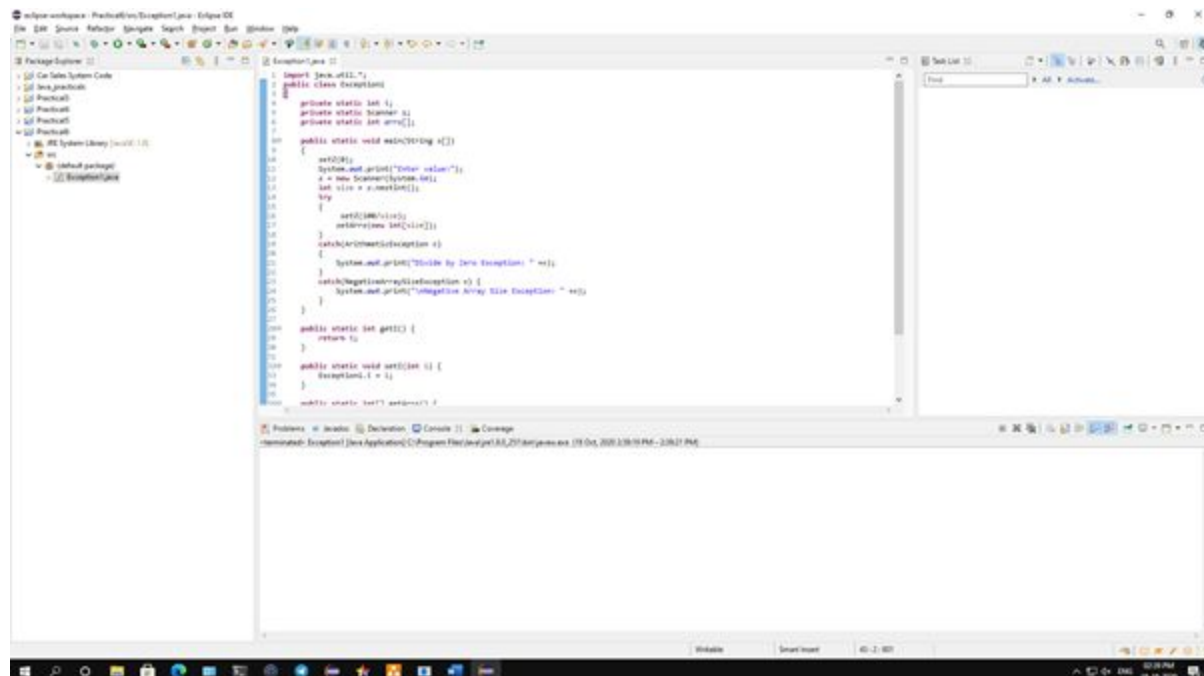
Practical No 6

A) To implement a program to demonstrate exceptions for negative array size and divide by zero etc.

Aim: Write a program to demonstrate exceptions for negative array size and divide by zero etc.

Description:

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled. Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions: Checked Exception, Unchecked Exception and Error. In this program we have defined the scanner class and int i for value and int arrs for exception classes. Then we defined the main method. After that we defined the size of the array. If the array is more than 100 it will throw exceptions such as zero or if we get a negative value it will throw a negative array size expectation.



```
import java.util.*;

public class Exception {

    private static int i;
    private static Scanner s;
    private static int arrs[];

    public static void main(String s[]) {

        s[0] = "0";
        System.out.println("Enter value");
        s = new Scanner(System.in);
        int size = s.nextInt();
        try {
            arrs = new int[size];
            arrs[0] = s.nextInt();
        } catch (ArithmeticException e) {
            System.out.println("Divide by 0 Exception" + e);
        } catch (NegativeArraySizeException e) {
            System.out.println("Negative Array Size Exception" + e);
        }

        public static int getI() {
            return i;
        }

        public static void setI(int i) {
            Exception.i = i;
        }

        public static void setI(String s) {
            Exception.i = s;
        }
    }
}
```

Conclusion: We have written a program to demonstrate exceptions for negative array size and divide by zero etc.

Code:

```
import java.util.*;

public class Exception1
{
    private static int i;

    private static Scanner s;

    private static int arrs[];

    public static void main(String a[])
    {
        setI(0);

        System.out.print("Enter value:");

        s = new Scanner(System.in);

        int size = s.nextInt();

        try
        {
            setI(100/size);

            setArrs(new int[size]);

        }

        catch(ArithmeticException e)
        {
            System.out.print("Divide by Zero Exception: " +e);

        }

        catch(NegativeArraySizeException e) {

            System.out.print("\nNegative Array Size Exception: " +e);

        }

    }
}
```



```
    public static int getI()  
    {  
        return i;  
    }  
  
    public static void setI(int i)  
    {  
        Exception1.i = i;  
    }  
  
    public static int[] getArrs()  
    {  
        return arrs;  
    }  
  
    public static void setArrs(int arrs[])  
    {  
        Exception1.arrs = arrs;  
    }  
}
```

Output:

Negative array size exception:

Enter value: -3

Negative Array Size Exception: java.lang.NegativeArraySizeException

Divide by zero exceptions:

Enter value: 0

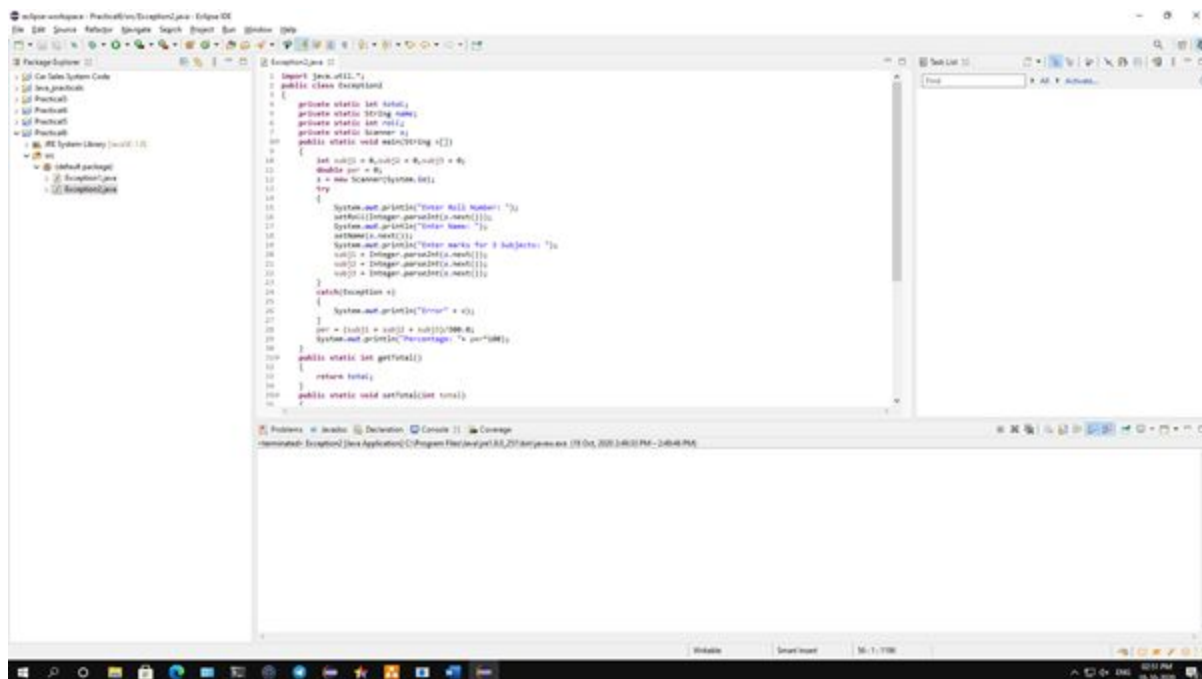
Divide by Zero Exception: [java.lang.ArithmeticException](#): / by zero

B) To implement a program to demonstrate NumberFormatException: Enter Roll No, Name, Marks and Subject from command line and calculate percentage.

Aim: Write a java program to implement a program to demonstrate NumberFormatException: Enter Roll No, Name, Marks and Subject from command line and calculate percentage.

Description:

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled. There are three types of exceptions: Checked Exception, Unchecked Exception and Error. In this program we use private class to define total, name and roll number, then we use the main functions. After that we defined the int for the subject, after that we have defined the try keyword to try the exception if the exception is found in the code the error will throw and the program will terminate, if there are no errors the code will compile and output will be displayed.



```
1 import java.util.*;
2 public class Exception2 {
3 {
4     private static int total;
5     private static String name;
6     private static int roll;
7     private static Scanner s;
8
9     public static void main(String[] args) {
10 {
11     int roll() = 0, sub1() = 0, sub2() = 0;
12     double per = 0;
13     s = new Scanner(System.in);
14     try {
15         System.out.println("Enter Roll Number: ");
16         roll = Integer.parseInt(s.next());
17         System.out.println("Enter Name: ");
18         name = s.next();
19         System.out.println("Enter marks for 3 subjects: ");
20         sub1 = Integer.parseInt(s.next());
21         sub2 = Integer.parseInt(s.next());
22         sub3 = Integer.parseInt(s.next());
23     }
24     catch(Exception e) {
25         System.out.println("Error" + e);
26     }
27     per = (sub1 + sub2 + sub3)/300.0;
28     System.out.println("Percentage: ", per*100);
29 }
30     public static int getTotal() {
31     return total;
32 }
33     public static void setTotal(int total) {
34     }
35 }
```

Conclusion: We have written a program to implement a program to demonstrate NumberFormatException: Enter Roll No, Name, Marks and Subject from command line and calculate percentage.

Code:

```
import java.util.*;
public class Exception2
{
    private static int total;
    private static String name;
    private static int roll;
    private static Scanner s;
    public static void main(String x[])
    {
        int subj1 = 0,subj2 = 0,subj3 = 0;
        double per = 0;
        s = new Scanner(System.in);
        try
        {
            System.out.println("Enter Roll Number: ");
            setRoll(Integer.parseInt(s.next()));
            System.out.println("Enter Name: ");
            setName(s.next());
            System.out.println("Enter marks for 3 Subjects: ");
            subj1 = Integer.parseInt(s.next());
            subj2 = Integer.parseInt(s.next());
            subj3 = Integer.parseInt(s.next());
        }
        catch(Exception e)
        {
            System.out.println("Error" + e);
        }
        per = (subj1 + subj2 + subj3)/300.0;
        System.out.println("Percentage: "+ per*100);
    }
    public static int getTotal()
    {
        return total;
    }
}
```

```
    public static void setTotal(int total)
    {
        Exception2.total = total;
    }
    public static String getName()
    {
        return name;
    }
    public static void setName(String name)
    {
        Exception2.name = name;
    }
    public static int getRoll()
    {
        return roll;
    }
    public static void setRoll(int roll)
    {
        Exception2.roll = roll;
    }
}
```

Output:

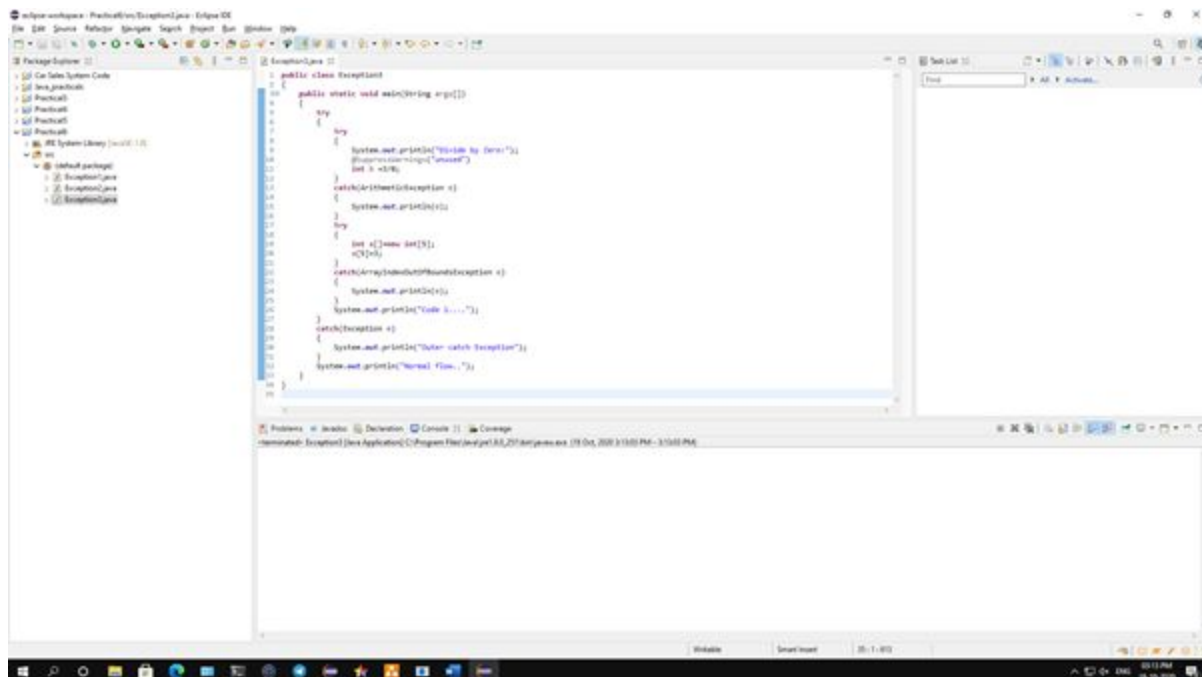
```
Enter Roll Number:
33
Enter Name:
Ninad
Enter marks for 3 Subjects:
80
78
a71
Error java.lang.NumberFormatException: For input string: "a71"
Percentage: 52.666666666666664
```

C) To implement a program to implement nested try, multiply, catch and finally.

Aim: Write a java program to implement nested try, multiply catch and finally.

Description:

There are 5 keywords which are used in handling exceptions in Java. They are Try, Catch, Finally, Throw, Throws. The "try" keyword is used to specify a block where we should place exception code. The "catch" block is used to handle the exception. The "finally" block is used to execute the important code of the program. The "throw" keyword is used to throw an exception. The "throws" keyword is used to declare exceptions. In this program we defined the public class followed by the main method. Then we use try block to specify the block of exceptions divided by zero. Then we use a catch block to handle the exception. Again, we define try block for specifying the new block of exception. The catch block will be used to handle that exception for code 1. And it will catch that exception in the last one and then output will show.



```
public class Exception {  
    public static void main(String args[])  
    {  
        try  
        {  
            System.out.println("Welcome to Java");  
            @SuppressWarnings("unused")  
            int i = 0; // throws  
        }  
        catch (ArithmeticException e)  
        {  
            System.out.println(i);  
        }  
        try  
        {  
            int a[] = new int[5];  
            a[0] = 0; // throws  
        }  
        catch (ArrayIndexOutOfBoundsException e)  
        {  
            System.out.println(i);  
        }  
        System.out.println("Code 3...");  
    }  
    catch (Exception e)  
    {  
        System.out.println("Outer catch Exception");  
    }  
    System.out.println("Normal flow.");  
}
```

Conclusion: We have implemented a program on nested try, multiply catch and finally.

Code:

```
public class Exception3
{
    public static void main(String args[])
    {
        try
        {
            try
            {
                System.out.println("Divide by Zero:");
                @SuppressWarnings("unused")
                int b = 3/0;
            }
            catch(ArithmeticException e)
            {
                System.out.println(e);
            }
            try
            {
                int a[] = new int[5];
                a[5] = 3;
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
                System.out.println(e);
            }
            System.out.println("Code 1....");
        }
        catch(Exception e)
        {
            System.out.println("Outer catch Exception");
        }
        System.out.println("Normal flow..");
    }
}
```

Output:

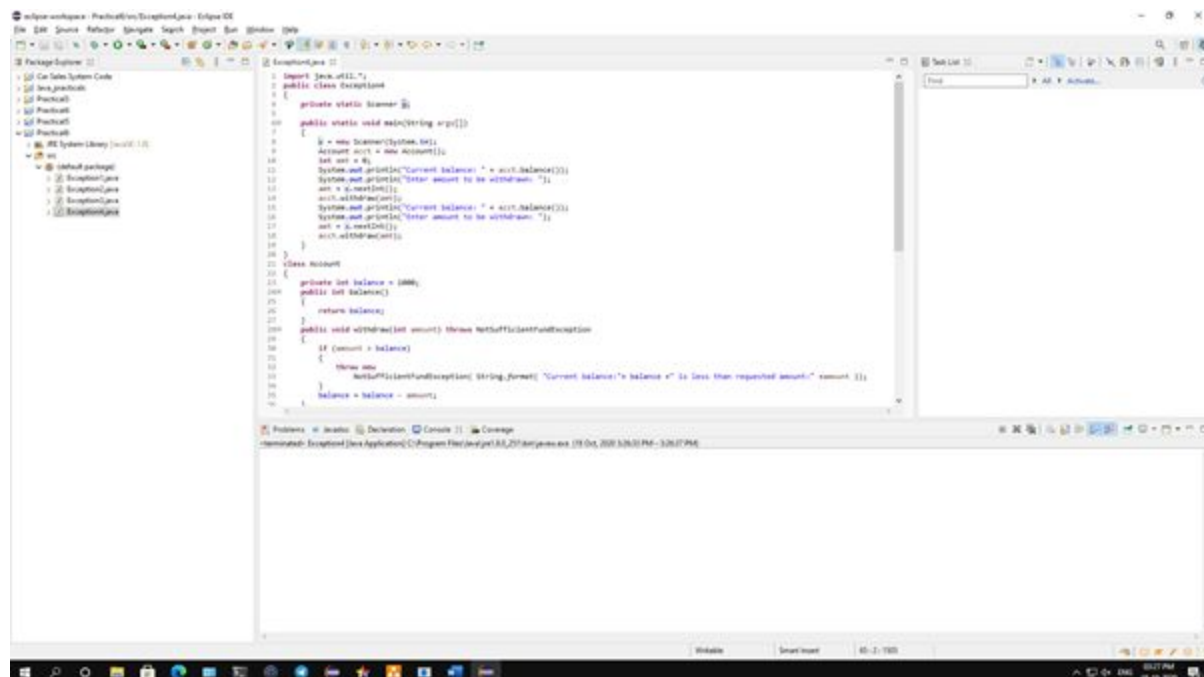
```
Divide by Zero:
java.lang.ArithmeticException: / by zero
java.lang.ArrayIndexOutOfBoundsException: 5
Code 1....
Normal flow..
```

D) To implement a program to create user defined exceptions. Create class Bank and define methods open (), deposit() and withdraw() with minimum balance 500. Create an exception Payoutofbounds and fire exception.

Aim: Write a java program to implement a program to create user defined exceptions. Create class Bank and define methods open (), deposit() and withdraw() with minimum balance 500. Create an exception Payoutofbounds and fire exception.

Description:

There are 5 keywords which are used in handling exceptions in Java. They are Try, Catch, Finally, Throw, Throws. The "try" keyword is used to specify a block where we should place exception code. The "catch" block is used to handle the exception. The "finally" block is used to execute the important code of the program. The "throw" keyword is used to throw an exception. The "throws" keyword is used to declare exceptions. In this program we defined the public class followed by the main method and also the scanner class. Then we define the account and int amount to get the output of current account balance. We use another class as Account which will read the current balance as 1000 and we use throws keyword to declare as exceptions if the balance is more than the remaining balance it will throw the exception. If the balance is less than the remaining balance it will display the output.



```
1 import java.util.*;
2 public class BankException
3 {
4     private static Scanner s;
5
6     public static void main(String args[])
7     {
8         s = new Scanner(System.in);
9         Account acc = new Account();
10        int amt = 0;
11        System.out.println("Current balance: " + acc.balance());
12        System.out.println("Enter amount to be withdrawn: ");
13        amt = s.nextInt();
14        acc.withdraw(amt);
15        System.out.println("Current balance: " + acc.balance());
16        System.out.println("Enter amount to be withdrawn: ");
17        amt = s.nextInt();
18        acc.withdraw(amt);
19    }
20 }
21
22 class Account
23 {
24     private int balance = 1000;
25     public int balance()
26     {
27         return balance;
28     }
29     public void withdraw(int amount) throws BankException
30     {
31         if (amount > balance)
32         {
33             throw new
34                 BankException("Current balance is less than requested amount.");
35         }
36         balance = balance - amount;
37     }
38 }
```

Conclusion: We have implemented a program to create user defined exceptions. Create class Bank and define methods open (), deposit() and withdraw() with minimum balance 500. Create an exception Payoutofbounds and fire exception.

Code:

```
import java.util.*;
public class Exception4
{
    private static Scanner s;
    public static void main(String args[]) {
        s = new Scanner(System.in);
        Account acct = new Account();
        int amt = 0;
        System.out.println("Current balance: " + acct.balance());
        System.out.println("Enter amount to be withdrawn: ");
        amt = s.nextInt();
        acct.withdraw(amt);
        System.out.println("Current balance: " + acct.balance());
        System.out.println("Enter amount to be withdrawn: ");
        amt = s.nextInt();
        acct.withdraw(amt);
    }
}

class Account {
    private int balance = 1000;
    public int balance()
    {
        return balance;
    }
    public void withdraw(int amount) throws NotSufficientFundException
    {
        if (amount > balance)
        {
            throw new
                NotSufficientFundException( String.format( "Current balance:"+ balance +" is less
than requested amount:" +amount ));
        }
    }
}
```



```
    }
    balance = balance - amount;
    }
    public void deposit(int amount)
    {
        if (amount <= 0)    {
amount));    throw new IllegalArgumentException(String.format("Invalid deposit amount:"+
        }
    }
}

class NotSufficientFundException extends RuntimeException
{
    /**
     *
    */
    private static final long serialVersionUID = 1L;
    private String message;
    public NotSufficientFundException(String message) {
        this.message = message; }
    public NotSufficientFundException(Throwable cause, String message)    {
        super(cause);
        this.message = message;
    }
    public String getMessage()
    {
        return message;
    }
}
```

Output:

```
Current balance: 1000
Enter amount to be withdrawn:
800
Current balance: 200
Enter amount to be withdrawn:
400
Exception in thread "main" NotSufficientFundException: Current balance:200 is less than requested amount:400
    at Account.withdraw(Exception4.java:32)
    at Exception4.main(Exception4.java:18)
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 07		
Title Of Lab Practical: MULTITHREADING				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

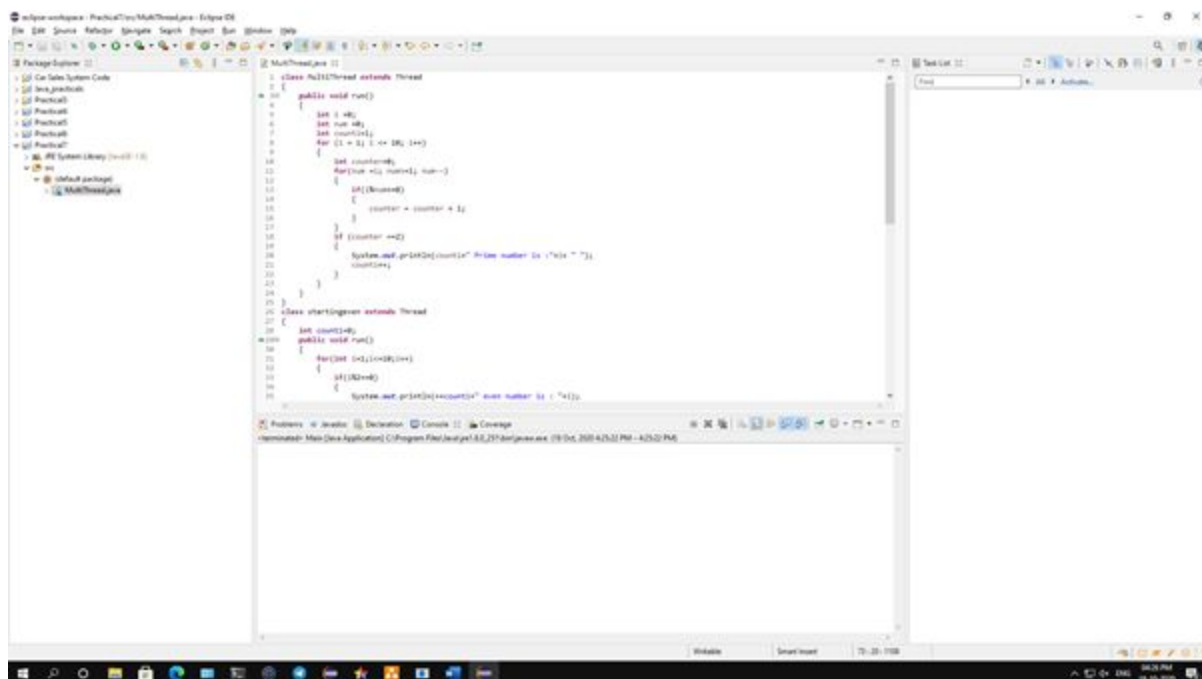
Practical No 7

A) To implement a program to demonstrate multithreading: even numbers between 1 to 100, Prime Numbers between 1 to 100 and Fibonacci Series.

Aim: Write a program to demonstrate multithreading: even numbers between 1 to 100, Prime Numbers between 1 to 100 and Fibonacci Series.

Description:

Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading rather than multiprocessing because threads use a shared memory area. They don't allocate separate memory areas so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc. In this program we have defined the class multithread which extends thread classes. Then we use run as a public method and we define int as I, num and count. It will print the prime number. Then we defined a stringeven class that extends thread which will also count the even number. Then we defined fibo class which will extend thread which will print the fib no followed by the main class and the output will be shown.



```
1 class Multithread extends Thread
2 {
3     public void run()
4     {
5         int i = 0;
6         int num = 0;
7         int count = 0;
8         for (i = 1; i <= 100; i++)
9         {
10             if (isPrime(i))
11             {
12                 num = i;
13                 count++;
14             }
15             if (count % 2 == 0)
16             {
17                 System.out.println(count + " Prime number is: " + num + " ");
18                 count++;
19             }
20         }
21     }
22 }
23
24 class stringeven extends Thread
25 {
26     public void run()
27     {
28         int count = 0;
29         for (i = 1; i <= 100; i++)
30         {
31             if (i % 2 == 0)
32             {
33                 System.out.println(count + " even number is: " + i);
34                 count++;
35             }
36         }
37     }
38 }
39
40 class fibo extends Thread
41 {
42     public void run()
43     {
44         int i = 0;
45         int num = 0;
46         int count = 0;
47         for (i = 1; i <= 100; i++)
48         {
49             if (isFibo(i))
50             {
51                 num = i;
52                 count++;
53             }
54             if (count % 2 == 0)
55             {
56                 System.out.println(count + " Fibonacchi number is: " + num + " ");
57                 count++;
58             }
59         }
60     }
61 }
62
63 public class Main {
64     public static void main(String[] args) {
65         Multithread mt = new Multithread();
66         stringeven se = new stringeven();
67         fibo f = new fibo();
68         mt.start();
69         se.start();
70         f.start();
71         try {
72             mt.join();
73             se.join();
74             f.join();
75         } catch (InterruptedException e) {
76             e.printStackTrace();
77         }
78     }
79 }
```

Conclusion: We have written a program to demonstrate multithreading: even numbers between 1 to 100, Prime Numbers between 1 to 100 and Fibonacci Series.

Code:

```
class MultiThread extends Thread
{
    public void run()
    {
        int i =0;
        int num =0;
        int count1=1;
        for (i = 1; i <= 10; i++)
        {
            int counter=0;
            for(num =i; num>=1; num--)
            {
                if(i%num==0)
                {
                    counter = counter + 1;
                }
            }
            if (counter ==2)
            {
                System.out.println(count1+" Prime number is :"+i+ " ");
                count1++;
            }
        }
    }
}

class startingeven extends Thread
{
    int count1=0;
```

```
        public void run()
        {
            for(int i=1;i<=10;i++)
            {
                if(i%2==0)
                {
                    System.out.println(++count1+" even number is : "+i);
                }
            }
        }
    }

    class fibo extends Thread
    {
        long a=0,b=1,c=0,n=1;
        public void run()
        {
            while(a<10)
            {
                System.out.println(n+"th" +" Fib no: = "+a);
                n++;
                c=a+b;
                a=b;
                b=c;
                try
                {
                    if(a==21)
                    {
                        Thread.sleep(500);
                    }
                }
                catch(Exception e)
                {}
            }
        }
    }
```

```
    }  
}  
class Main  
{  
    public static void main(String[] args)  
    {  
        MultiThread sp = new MultiThread();  
        startingeven se = new startingeven();  
        fibo fb = new fibo();  
        sp.start();  
        se.start();  
        fb.start();  
    }  
}
```

Output:

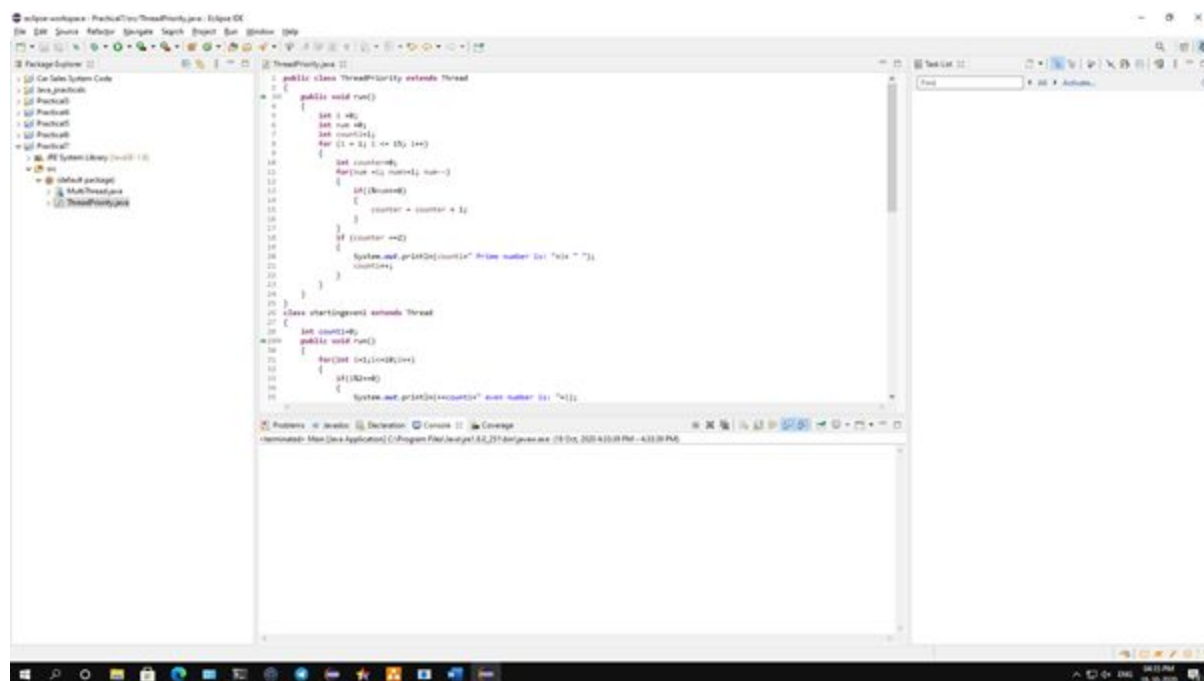
```
1 Prime number is :2  
2 Prime number is :3  
3 Prime number is :5  
4 Prime number is :7  
1 even number is : 2  
2 even number is : 4  
3 even number is : 6  
4 even number is : 8  
5 even number is : 10  
1th Fib no: = 0  
2th Fib no: = 1  
3th Fib no: = 1  
4th Fib no: = 2  
5th Fib no: = 3  
6th Fib no: = 5  
7th Fib no: = 8
```

B) To implement a program to demonstrate Thread Priority in above program.

Aim: Write a java program to implement a program to demonstrate Thread Priority in above program.

Description:

Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification which scheduling it chooses. 3 constants defined in Thread class are public static int MIN_PRIORITY, public static int NORM_PRIORITY, public static int MAX_PRIORITY. Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10. In this program we have defined the class multithread which extends thread classes. Then we use run as a public method and we define int as I, num and count. It will print the prime number. Then we defined a stringeven class that extends thread which will also count the even number. Then we defined fibo class which will extend thread which will print the fib no. Then we defined the prior class which will show the default priority and current priority.



```
public class ThreadPriority extends Thread {
    private static int count = 0;
    private static int num = 0;
    private static int i = 0;

    public ThreadPriority() {
        setPriority(10);
    }

    public void run() {
        while (true) {
            if (isAlive()) {
                num = num + 1;
                count = count + 1;
            }
            if (count % 2 == 0) {
                System.out.println(count + " Even number is: " + num + " ");
                count = 0;
            }
        }
    }
}

class StringEven extends Thread {
    private static int count = 0;
    private static int num = 0;

    public StringEven() {
        setPriority(5);
    }

    public void run() {
        while (true) {
            num = num + 1;
            count = count + 1;
            if (count % 2 == 0) {
                System.out.println(count + " Even number is: " + num + " ");
            }
        }
    }
}

class Fibo extends Thread {
    private static int count = 0;
    private static int num = 0;

    public Fibo() {
        setPriority(1);
    }

    public void run() {
        while (true) {
            num = num + 1;
            count = count + 1;
            if (count % 2 == 0) {
                System.out.println(count + " Even number is: " + num + " ");
            }
        }
    }
}
```

Conclusion: We have implemented a program to demonstrate Thread Priority in the above program.

Code:

```
public class ThreadPriority extends Thread
{
    public void run()
    {
        int i =0;
        int num =0;
        int count1=1;
        for (i = 1; i <= 15; i++)
        {
            int counter=0;
            for(num =i; num>=1; num--)
            {
                if(i%num==0)
                {
                    counter = counter + 1;
                }
            }
            if (counter ==2)
            {
                System.out.println(count1+" Prime number is: "+i+ " ");
                count1++;
            }
        }
    }
}

class startingeven1 extends Thread
{
    int count1=0;
    public void run()
    {
        for(int i=1;i<=10;i++)
        {
            if(i%2==0)
            {
```



```
        System.out.println(++count1+" even number is: "+i);
    }
}
}

class fibo1 extends Thread
{
    long a=0,b=1,c=0,n=1;
    public void run()
    {
        while(a<40)
        {
            System.out.println(n+"th" +" Fib no: = "+a);
            n++;
            c=a+b;
            a=b;
            b=c;
            try
            {
                if(a==21)
                {
                    Thread.sleep(500);
                }
            }
            catch(Exception e)
            {}
        }
    }
}

class thread_prior
{
    public static void main(String[] args)
    {
        ThreadPriority sp = new ThreadPriority();
        startingeven1 se = new startingeven1();
        fibo1 fb = new fibo1();
    }
}
```

```
        sp.start();
        System.out.println("Default priority of "+ sp.getName() + " : " + sp.getPriority());
        se.start();
        System.out.println("Default priority of "+ se.getName() + " : " + se.getPriority());
        fb.start();
        System.out.println("Default priority of "+ fb.getName() + " : " + fb.getPriority());
        se.setPriority(Thread.MAX_PRIORITY);
        sp.setPriority(Thread.MIN_PRIORITY);
        System.out.println("Current priority of "+ sp.getName() + " : " + sp.getPriority());
        System.out.println("Current priority of "+ se.getName() + " : " + se.getPriority());
    }
}
```

Output:

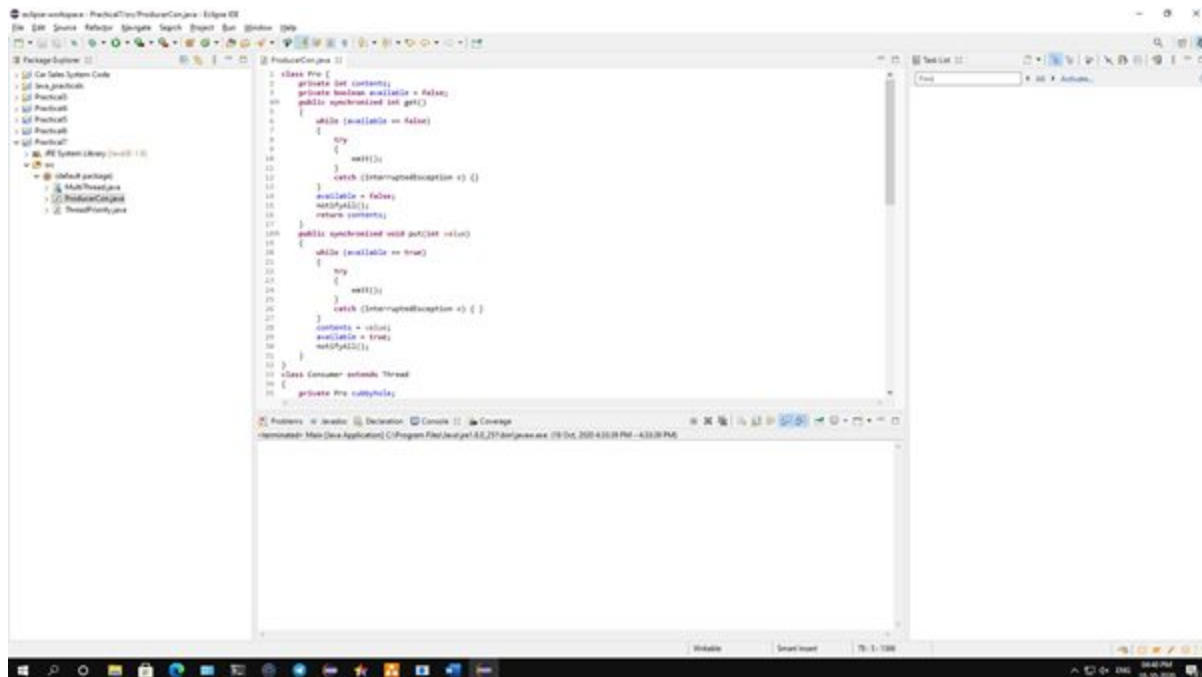
```
1 Prime number is :2
2 Prime number is :3
3 Prime number is :5
4 Prime number is :7
1 even number is : 2
2 even number is : 4
3 even number is : 6
4 even number is : 8
5 even number is : 10
1th Fib no: = 0
2th Fib no: = 1
3th Fib no: = 1
4th Fib no: = 2
5th Fib no: = 3
6th Fib no: = 5
7th Fib no: = 8
```

C) To implement Multi-threaded Producer Consumer Application.

Aim: Write a java program to implement Multi-threaded Producer Consumer Application.

Description:

In computing, the producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time. In this program we have defined pro classes and the use of private and public classes. Then we defined the try block followed by catch block. Then we define class consumer which extends thread class followed by producer class and its main functions and the output will be displayed.



```
1 class Producer {
2     private int contents;
3     private boolean available = false;
4     public synchronized int get() {
5         while (available == false)
6             wait();
7     }
8     public synchronized void put(int value) {
9         while (available == true)
10            wait();
11        contents = value;
12        available = true;
13        notify();
14    }
15 }
16 class Consumer extends Thread {
17     private Producer obj;
18     public Consumer(Producer obj) {
19         this.obj = obj;
20     }
21     public void run() {
22         while (true) {
23             int value = obj.get();
24             if (value != -1) {
25                 System.out.println("Consumer: " + value);
26             }
27         }
28     }
29 }
30 public class ProducerConsumer {
31     public static void main(String[] args) {
32         Producer producer = new Producer();
33         Consumer consumer = new Consumer(producer);
34         producer.start();
35         consumer.start();
36     }
37 }
```

Conclusion: We have implemented a program on Multi-threaded Producer Consumer Application.

Code:

```
class Pro {  
    private int contents;  
    private boolean available = false;  
    public synchronized int get()  
    {  
        while (available == false)  
        {  
            try  
            {  
                wait();  
            }  
            catch (InterruptedException e) {}  
        }  
        available = false;  
        notifyAll();  
        return contents;  
    }  
    public synchronized void put(int value)  
    {  
        while (available == true)  
        {  
            try  
            {  
                wait();  
            }  
            catch (InterruptedException e) { }  
        }  
        contents = value;  
        available = true;  
        notifyAll();  
    }  
}  
class Consumer extends Thread  
{
```

```
    private Pro cubbyhole;
    private int number;
    public Consumer(Pro c, int number)
    {
        cubbyhole = c;
        this.number = number;
    }
    public void run()
    {
        int value = 0;
        for (int i = 0; i < 10; i++)
        {
            value = cubbyhole.get();
            System.out.println("Consumer #" + this.number + " got: " + value);
        }
    }
}

class Producer extends Thread
{
    private Pro cubbyhole;
    private int number;
    public Producer(Pro c, int number)
    {
        cubbyhole = c;
        this.number = number;
    }
    public void run()
    {
        for (int i = 0; i < 10; i++)
        {
            cubbyhole.put(i);
            System.out.println("Producer #" + this.number + " put: " + i);
            try
            {
                sleep((int)(Math.random() * 100));
            }
        }
    }
}
```

```
        catch (InterruptedException e) { }  
    }  
}  
  
public class ProducerCon  
{  
    public static void main(String[] args)  
    {  
        Pro c = new Pro();  
        Producer p1 = new Producer(c, 1);  
        Consumer c1 = new Consumer(c, 1);  
        p1.start();  
        c1.start();  
    }  
}
```

Output:

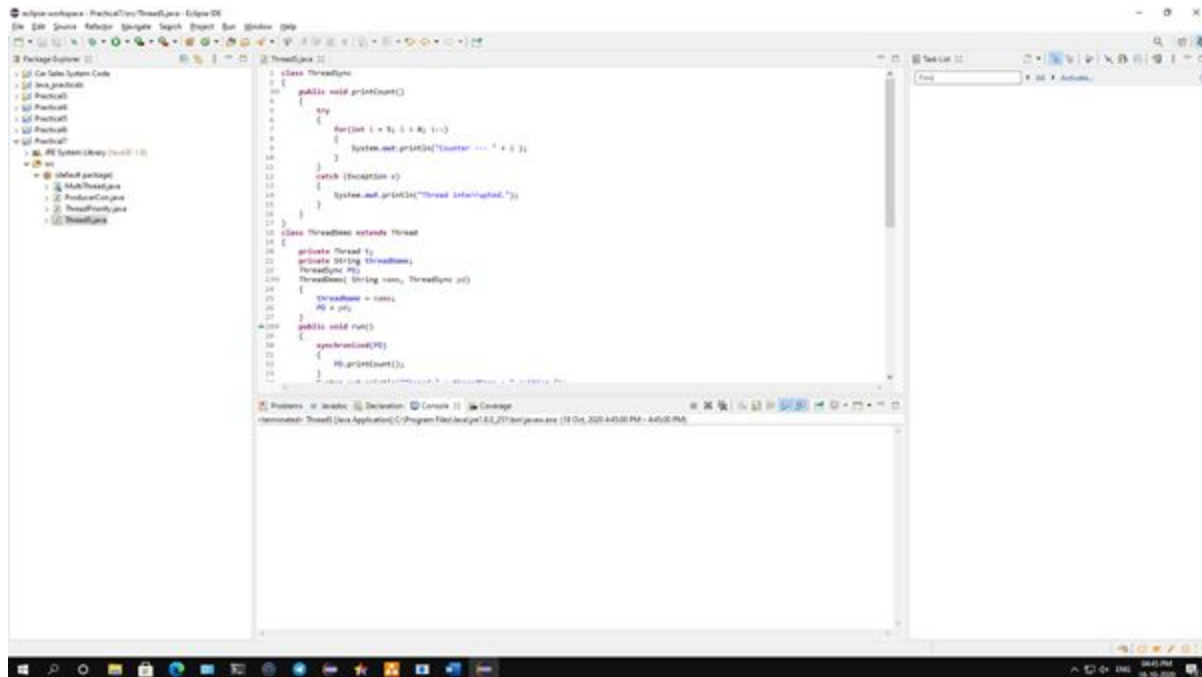
```
Producer #1 put: 0  
Consumer #1 got: 0  
Producer #1 put: 1  
Consumer #1 got: 1  
Producer #1 put: 2  
Consumer #1 got: 2  
Producer #1 put: 3  
Consumer #1 got: 3  
Producer #1 put: 4  
Consumer #1 got: 4  
Producer #1 put: 5  
Consumer #1 got: 5  
Producer #1 put: 6  
Consumer #1 got: 6  
Producer #1 put: 7  
Consumer #1 got: 7  
Producer #1 put: 8  
Consumer #1 got: 8  
Producer #1 put: 9  
Consumer #1 got: 9
```

D) Program on Thread Synchronization.

Aim: Write a java program to implement Thread Synchronization.

Description:

Multi-threaded programs may often come to a situation where multiple threads try to access the same resources and finally produce erroneous and unforeseen results. So, it needs to be made sure by some synchronization method that only one thread can access the resource at a given point of time. Java provides a way of creating threads and synchronizing their task by using synchronized blocks. Synchronized blocks in Java are marked with the synchronized keyword. A synchronized block in Java is synchronized on some object. All synchronized blocks synchronized on the same object can only have one thread executing inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block. We have defined ThreadSync class followed by counter and catch exception, it will throw the exception if the thread is interrupted. The class threaddemo will extend the thread class followed by public and private class. Then the main function will be declared the output will be displayed.



```
1 class ThreadSync
2 {
3     public void printCount()
4     {
5         try
6         {
7             for(int i = 0; i < 5; i++)
8             {
9                 System.out.println("Counter ... " + i);
10            }
11        }
12        catch (Exception e)
13        {
14            System.out.println("Thread interrupted.");
15        }
16    }
17 }
18 class ThreadDemo extends Thread
19 {
20     private Thread t;
21     private String ThreadName;
22     ThreadSync TS;
23     ThreadDemo(String name, ThreadSync ts)
24     {
25         ThreadName = name;
26         TS = ts;
27     }
28     public void run()
29     {
30         synchronized(TS)
31         {
32             TS.printCount();
33         }
34     }
35 }
```

ThreadDemo [Java Application] C:\Program Files\Java\jdk-8.0.520\bin\java.exe (19 Oct, 2020 4:05:02 PM) - Auto-Stop

Conclusion: We have implemented a program on Thread Synchronization.

Code:

```
class ThreadSync
{
    public void printCount()
    {
        try
        {
            for(int i = 5; i > 0; i--)
            {
                System.out.println("Counter --- " + i );
            }
        }
        catch (Exception e)
        {
            System.out.println("Thread interrupted.");
        }
    }
}

class ThreadDemo extends Thread
{
    private Thread t;
    private String threadName;
    ThreadSync PD;
    ThreadDemo( String name, ThreadSync pd)
    {
        threadName = name;
        PD = pd;
    }
    public void run()
    {
        synchronized(PD)
        {
            PD.printCount();
        }
        System.out.println("Thread " + threadName + " exiting.");
    }
    public void start ()
    {
        System.out.println("Starting " + threadName );
        if (t == null)
        {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}
```



```
public class ThreadS
{
    public static void main(String args[])
    {
        ThreadSync PD = new ThreadSync();
        ThreadDemo T1 = new ThreadDemo( "Thread - 1 ", PD );
        ThreadDemo T2 = new ThreadDemo( "Thread - 2 ", PD );
        T1.start();
        T2.start();
        try
        {
            T1.join();
            T2.join();
        }
        catch ( Exception e)
        {
            System.out.println("Interrupted");
        }
    }
}
```

Output:

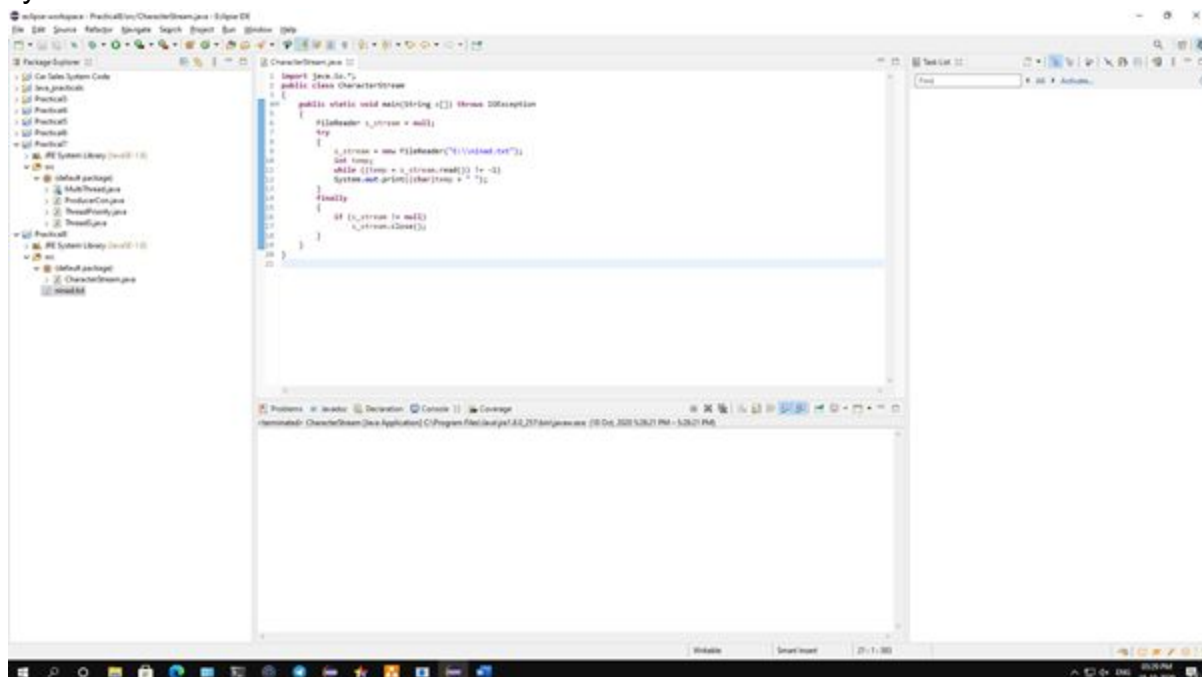
```
Starting Thread - 1
Starting Thread - 2
Counter --- 5
Counter --- 4
Counter --- 3
Counter --- 2
Counter --- 1
Thread Thread - 1 exiting.
Counter --- 5
Counter --- 4
Counter --- 3
Counter --- 2
Counter --- 1
Thread Thread - 2 exiting.
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 08		
Title Of Lab Practical: FILE HANDLING				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

A) To implement a program using console-based IO using character and byte stream.

Description:

A stream is a method to sequentially access a file. I/O Stream means an input source or output destination representing different types of sources e.g. disk files. The java.io package provides classes that allow you to convert between Unicode character streams and byte streams of non-Unicode text. In Java, characters are stored using Unicode conventions. Character stream automatically allows us to read/write data character by character. For example, FileReader and FileWriter are character streams used to read from source and write to destination. Byte streams process data byte by byte (8 bits). For example, FileInputStream is used to read from source and FileOutputStream to write to the destination. In the given program we have used CharacterStream and public class and then we defined main method with throws keyword, then we defined FileReader function to read the file which is defined in txt format with its dir. Then the finally keyword will display the output of the text file. In another program we use ByteStream class followed by public class as fileoutputstream. We defined the main function followed by throws keyword and then we used try keyword to handle the exception, we also defined the file which is in txt format. The output will be displayed in byte format.



Conclusion: We have written a program to implement console-based IO using character and byte stream.

Code:

Character Stream:

```
import java.io.*;

public class CharacterStream
{
    public static void main(String x[]) throws IOException
    {
        FileReader s_stream = null;

        try
        {
            s_stream = new FileReader("E:\\ninad.txt");
            int temp;
            while ((temp = s_stream.read()) != -1)
                System.out.print((char)temp + " ");
        }
        finally
        {
            if (s_stream != null)
                s_stream.close();
        }
    }
}
```

Output:

```
Hello World!!
```

Byte Stream:

```
import java.io.*;

public class ByteStream {

    private static FileOutputStream t_stream;

    public static void main(String[] args) throws IOException{

        FileInputStream s_stream = null;

        setT_stream(null);

        try    {

            s_stream = new FileInputStream("E:\\ninad.txt");

            int temp;

            while ((temp = s_stream.read()) != -1)

                System.out.print((byte)temp+ " ");

        }

        finally {

            if (s_stream != null)

                s_stream.close();

        }

    }

}
```

```
public static FileOutputStream getT_stream() {  
    return t_stream;  
}  
  
public static void setT_stream(FileOutputStream t_stream) {  
    ByteStream.t_stream = t_stream;  
}  
}
```

Output:

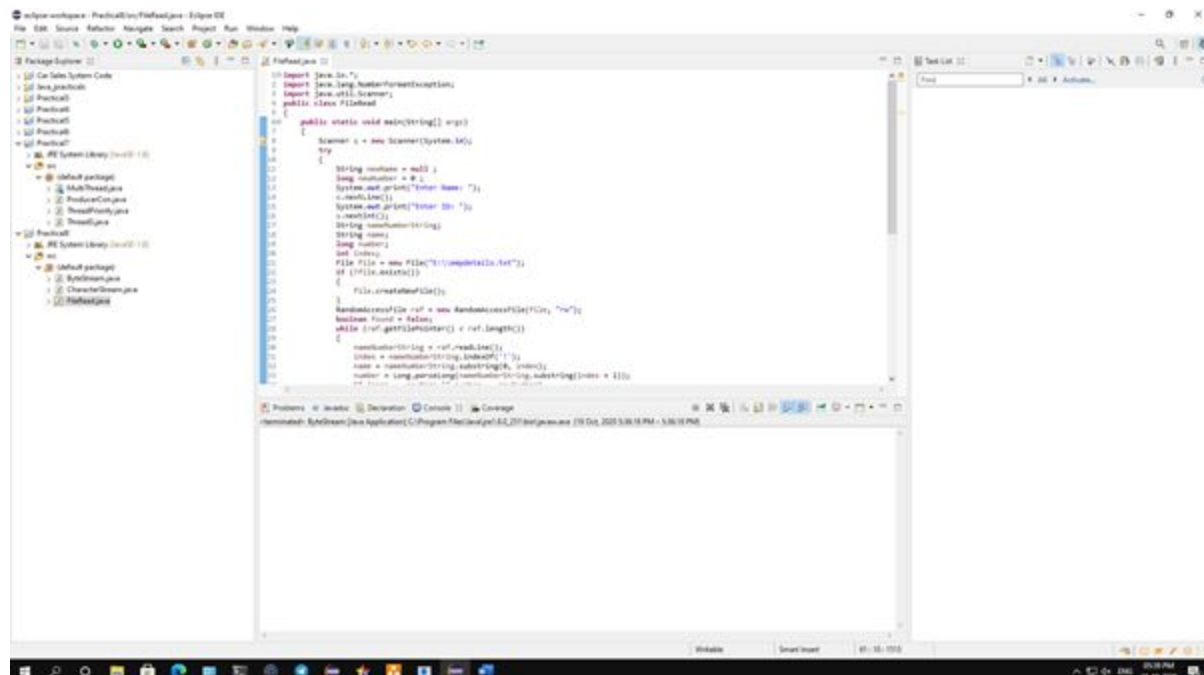
```
72 101 108 108 111 32 87 111 114 108 100 33 33
```

B) To implement a program to store the details of employees in file using file reader and writer class/File Input Stream and Output stream.

Aim: Write a java program to implement a program to store the details of employees in file using file reader and writer class/File Input Stream and Output stream.

Description:

I/O Stream means an input source or output destination representing different types of sources e.g. disk files. The java.io package provides classes that allow you to convert between Unicode character streams and byte streams of non-Unicode text. In Java, characters are stored using Unicode conventions. Character stream automatically allows us to read/write data character by character. For example, FileReader and FileWriter are character streams used to read from source and write to destination. Byte streams process data byte by byte (8 bits). For example, FileInputStream is used to read from source and FileOutputStream to write to the destination. In this program we have imported some java classes such as io, lang.numberformatexception and scanner class. Then we defined the fileread class followed by the main function, then we use scanner class followed by the try key block which is used to handle the exception. We define a long function for a new number and then the file function is used to read the file. The output will be displayed showing employee details.



```
1 import java.io.*;
2 import java.lang.NumberFormatException;
3 import java.util.Scanner;
4 public class Fileread {
5     public static void main(String[] args) {
6         Scanner s = new Scanner(System.in);
7         try {
8             String employee = null;
9             long number = 0;
10             System.out.println("Enter Name: ");
11             s.nextLine();
12             System.out.println("Enter ID: ");
13             s.nextLine();
14             String nameFunction(s.nextLine());
15             String name;
16             long number;
17             File file = new File("C:\\empdata\\a.txt");
18             if (!file.exists()) {
19                 file.createNewFile();
20             }
21             RandomAccessFile raf = new RandomAccessFile(file, "rw");
22             boolean found = false;
23             while (!raf.getFilePointer() < raf.length()) {
24                 numberFunction = raf.readLine();
25                 index = numberFunction.indexOf(",");
26                 name = numberFunction.substring(0, index);
27                 number = Long.parseLong(numberFunction.substring(index + 1));
28             }
29         } catch (Exception e) {
30             e.printStackTrace();
31         }
32     }
33 }
```

Conclusion: We have implemented a program to store the details of employees in file using file reader and writer class/File Input Stream and Output stream.

Code:

```
import java.io.*;
import java.lang.NumberFormatException;
import java.util.Scanner;
public class FileRead
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        try
        {
            String newName = null ;
            long newNumber = 0 ;
            System.out.print("Enter Name: ");
            s.nextLine();
            System.out.print("Enter ID: ");
            s.nextInt();
            String nameNumberString;
            String name;
            long number;
            int index;
            File file = new File("E:\\empdetails.txt");
            if (!file.exists())
            {
                file.createNewFile();
            }
            RandomAccessFile raf = new RandomAccessFile(file, "rw");
            boolean found = false;
            while (raf.getFilePointer() < raf.length())
            {
                nameNumberString = raf.readLine();
                index = nameNumberString.indexOf('!');
                name = nameNumberString.substring(0, index);
                Long.parseLong(nameNumberString.substring(index + 1));
                if (name == newName || number == newNumber)
                {
                    found = true;
                    break;
                }
            }
            if (found == false)
```



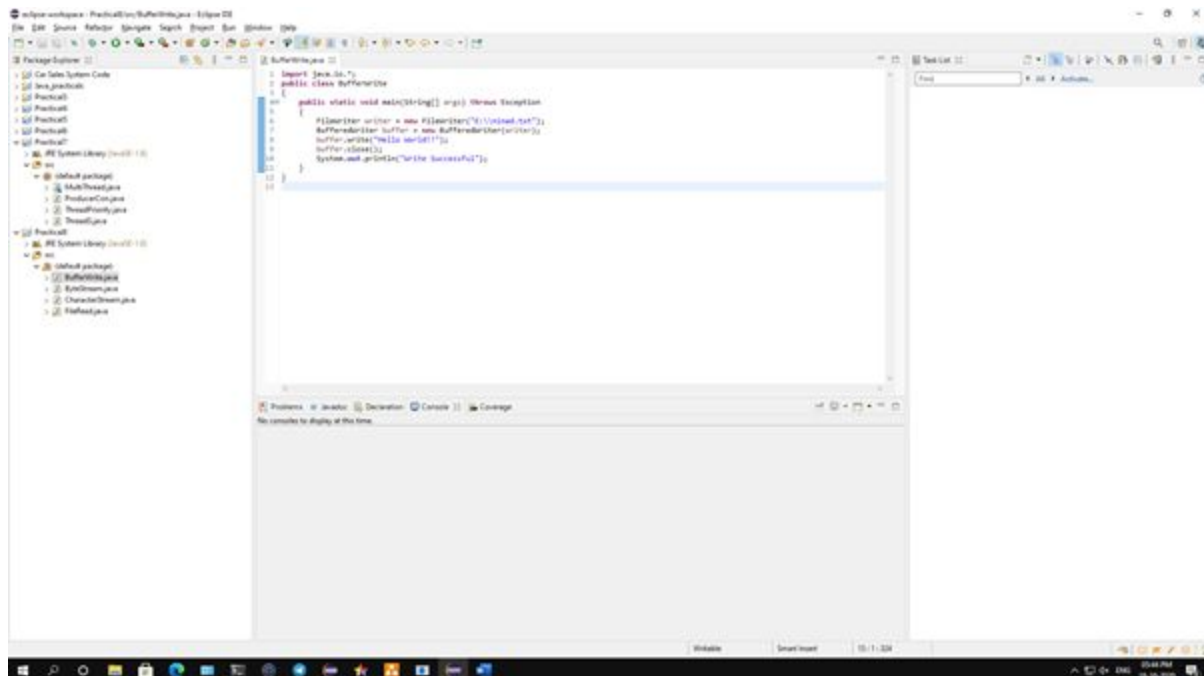
```
        {
String.valueOf(newNumber);    nameNumberString    =    newName+"!"+
                                raf.writeBytes(nameNumberString);
                                raf.writeBytes(System.lineSeparator());
                                System.out.println(" Employee added. ");
                                raf.close();
        }
        else
        {
                                raf.close();
exists. ");    System.out.println(" The entered employee" + " already
        }
    }
    catch (IOException ioe)
    {
        System.out.println(ioe);
    }
    catch (NumberFormatException nef)
    {
        System.out.println(nef);
    }
}
}
```

Output:

```
Enter Name: Ninad Patil
Enter ID: 33
Employee added.
```

Aim: Write a java program to implement buffered reader/writer.

Java `BufferedWriter` class is used to provide buffering for `Writer` instances. It makes the performance fast. It inherits `Writer` class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings. Java `BufferedReader` class is used to read the text from a character-based input stream. It can be used to read data line by line by `readLine()` method. It makes the performance fast. It inherits `Reader` class. In the first program we defined the `bufferwrite` class followed by the `main` function and the `throws` keyword which throws an exception. Then we use a file write function to write the character in the output. In the second program we defined the `bufferread` class followed by the `main` functions which also define the `throws` keyword to throws an exception. Then the file read function is used to read the file and it will print the output which will read the character in the files and display it.



Conclusion: We have implemented a program to read/write buffers.

Code:
Buffered Write:

```
import java.io.*;
public class BufferWrite
{
    public static void main(String[] args) throws Exception
    {
        FileWriter writer = new FileWriter("E:\\ninad.txt");
        BufferedWriter buffer = new BufferedWriter(writer);
        buffer.write("Hello World!!");
        buffer.close();
        System.out.println("Write Successful");
    }
}
```

Output:

```
Write Successful
```

Buffered Read:

```
import java.io.*;
public class BufferRead
{
    public static void main(String args[])throws Exception
    {
        FileReader fr=new FileReader("E:\\ninad.txt");
        BufferedReader br=new BufferedReader(fr);
        int i;
        while((i=br.read())!=-1)
        {
            System.out.print((char)i);
        }
        br.close();
        fr.close();
    }
}
```

Output:

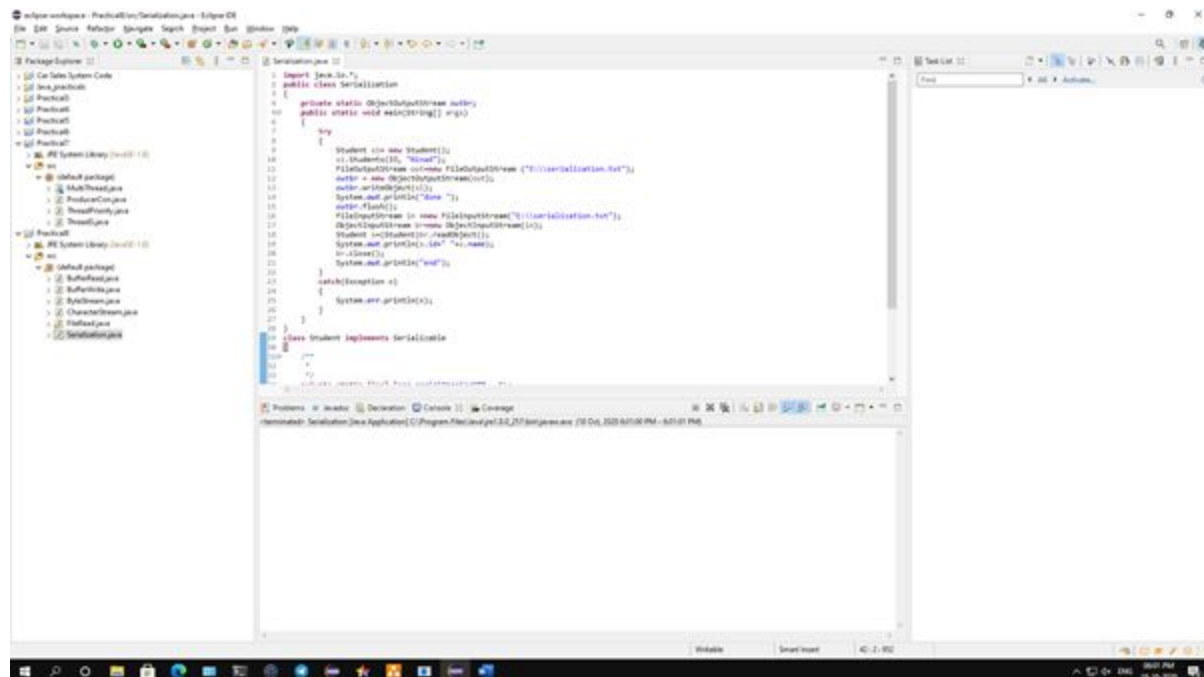
```
Hello World!!
```

D) To demonstrate Object serialization and Deserialization.

Aim: Write a java program to demonstrate Object serialization and Deserialization.

Description:

Serialization in Java is a mechanism of *writing the state of an object into a byte-stream*. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies. The reverse operation of serialization is called *deserialization* where byte-stream is converted into an object. The serialization and deserialization process are platform-independent, it means you can serialize an object in a platform and deserialize in different platforms. For serializing the object, we call the writeObject() method of *ObjectOutputStream*, and for deserialization we call the readObject() method of *ObjectInputStream* class. In the given program we defined the serialization class followed by the object outputstream and main function. Then we defined try keyword to handle the exception, then we define the output stream function to read the output and save it in a file name serialization.txt once it save it will proceed to inputstream function which helps to read and display the output of the content which is in serialization.txt file.



The screenshot shows an IDE with a Java file named 'Serialization.java'. The code defines a 'Student' class with attributes 'id', 'name', and 'age'. It implements the 'Serializable' interface. The main method uses 'ObjectOutputStream' to serialize a 'Student' object to 'serialization.txt' and 'ObjectInputStream' to deserialize it back into a 'Student' object, displaying its details. The IDE interface includes a Package Explorer on the left, a Source Editor in the center, and a Console at the bottom.

```
1 import java.io.*;
2 public class Serialization
3 {
4     private static ObjectOutputStream out;
5     public static void main(String[] args)
6     {
7         try
8         {
9             Student s = new Student();
10            s.setId(10);
11            s.setName("Ninad");
12            FileOutputStream fos = new FileOutputStream("serialization.txt");
13            ObjectOutputStream out = new ObjectOutputStream(fos);
14            out.writeObject(s);
15            out.close();
16            System.out.println("Done");
17            FileInputStream fis = new FileInputStream("serialization.txt");
18            ObjectInputStream in = new ObjectInputStream(fis);
19            Student s1 = (Student) in.readObject();
20            System.out.println(s1.getId() + " " + s1.getName());
21            in.close();
22            System.out.println("End");
23        }
24        catch (Exception e)
25        {
26            System.out.println(e);
27        }
28    }
29 }
30
31 class Student implements Serializable
32 {
33     int id;
34     String name;
35     int age;
36 }
```

Conclusion: We have demonstrated a program on Object serialization and Deserialization.

Code:

```
import java.io.*;

public class Serialization
{
    private static ObjectOutputStream outbr;

    public static void main(String[] args)
    {
        try
        {
            Student s1= new Student();
            s1.Students(33, "Ninad");

            FileOutputStream out=new FileOutputStream
("E:\\serialization.txt");

            outbr = new ObjectOutputStream(out);
            outbr.writeObject(s1);
            System.out.println("done ");
            outbr.flush();

            FileInputStream in =new FileInputStream("E:\\serialization.txt");
            ObjectInputStream br=new ObjectInputStream(in);
            Student s=(Student)br.readObject();
            System.out.println(s.id+" "+s.name);
            br.close();
            System.out.println("end");
        }
        catch(Exception e)
        {
            System.err.println(e);
        }
    }
}
```

```
        }  
    }  
}  
  
class Student implements Serializable  
{  
    /**  
     *  
     */  
    private static final long serialVersionUID = 1L;  
    int id;  
    String name;  
    public void Students(int id, String name)  
    {  
        this.id = id;  
        this.name = name;  
    }  
}
```

Output:

```
done  
33 Ninad  
end
```

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 09		
Title Of Lab Practical: PROGRAM BASED ON APPLETS AND SWINGS APPLETS				
DOP:		DOS:		
CO Mapped: CO1,CO2	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

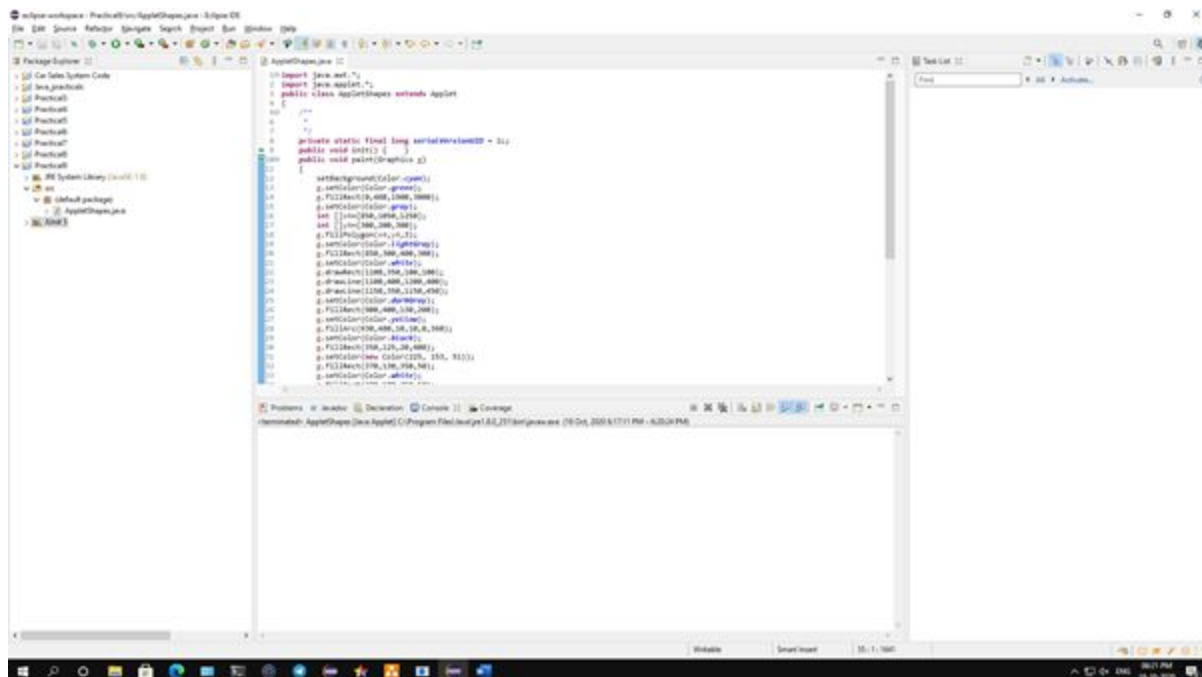
Practical No 9

A) To design an applet to display shapes like: House, Ashok Chakra, Indian flag.

Aim: Write a program to design an applet to display shapes like: House, Ashok Chakra, Indian flag.

Description:

An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. Four methods in the Applet class give you the framework on which you build any serious applet they are: init, start, stop, destroy and paint. In this program we first defined the java awt and applet package followed by appletshapes which extends applet class. Then we define the init function and paint which is used to show shapes and graphics. Then we set the background as cyan and green colours. Then we defined the required colours we need to complete the program. In the last we defined the math functions to make a circle, rectangle and triangle shape. The output will be displayed showing shapes like house, ashok chakra and the Indian flag.



Conclusion: We have designed an applet to display shapes like: House, Ashok Chakra, Indian flag.

Code:

```
import java.awt.*;

import java.applet.*;

public class AppletShapes extends Applet
{
    /**
     *
     */

    private static final long serialVersionUID = 1L;

    public void init() { }

    public void paint(Graphics g)
    {
        setBackground(Color.cyan);

        g.setColor(Color.green);

        g.fillRect(0,480,1900,3000);

        g.setColor(Color.gray);

        int []x4={850,1050,1250};

        int []y4={300,200,300};

        g.fillPolygon(x4,y4,3);

        g.setColor(Color.lightGray);

        g.fillRect(850,300,400,300);

        g.setColor(Color.white);

        g.drawRect(1100,350,100,100);
```

```
g.drawLine(1100,400,1200,400);
g.drawLine(1150,350,1150,450);
g.setColor(Color.darkGray);
g.fillRect(900,400,130,200);
g.setColor(Color.yellow);
g.fillArc(930,480,10,10,0,360);
g.setColor(Color.black);
g.fillRect(350,125,20,400);
g.setColor(new Color(225, 153, 51));
g.fillRect(370,130,350,50);
g.setColor(Color.white);
g.fillRect(370,180,350,50);
g.setColor(new Color(19, 136, 8));
g.fillRect(370,230,350,50);
g.setColor(new Color(0, 0, 128));
g.drawOval(500,180,50,50);

int n1=525;

int d1=205;

double n2,d2;

double angle= 0.0;

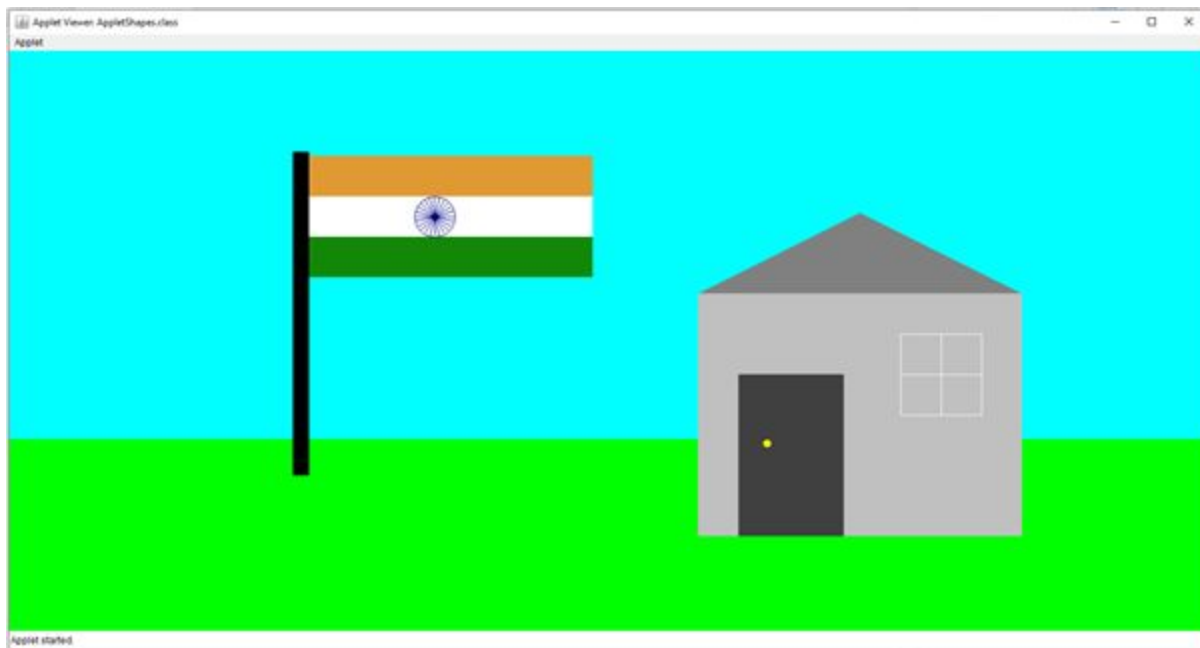
double line=0.0;

int r=23;

for(int i=1;i<=24;i++)
{
    angle=(double)line*(3.14/180);
    n2=n1+(double)r*Math.cos(angle);
    d2=d1+(double)r*Math.sin(angle);
```

```
g.drawLine(n1,d1,(int)n2,(int)d2);  
line=line+(360/24);  
}  
}  
}
```

Output:

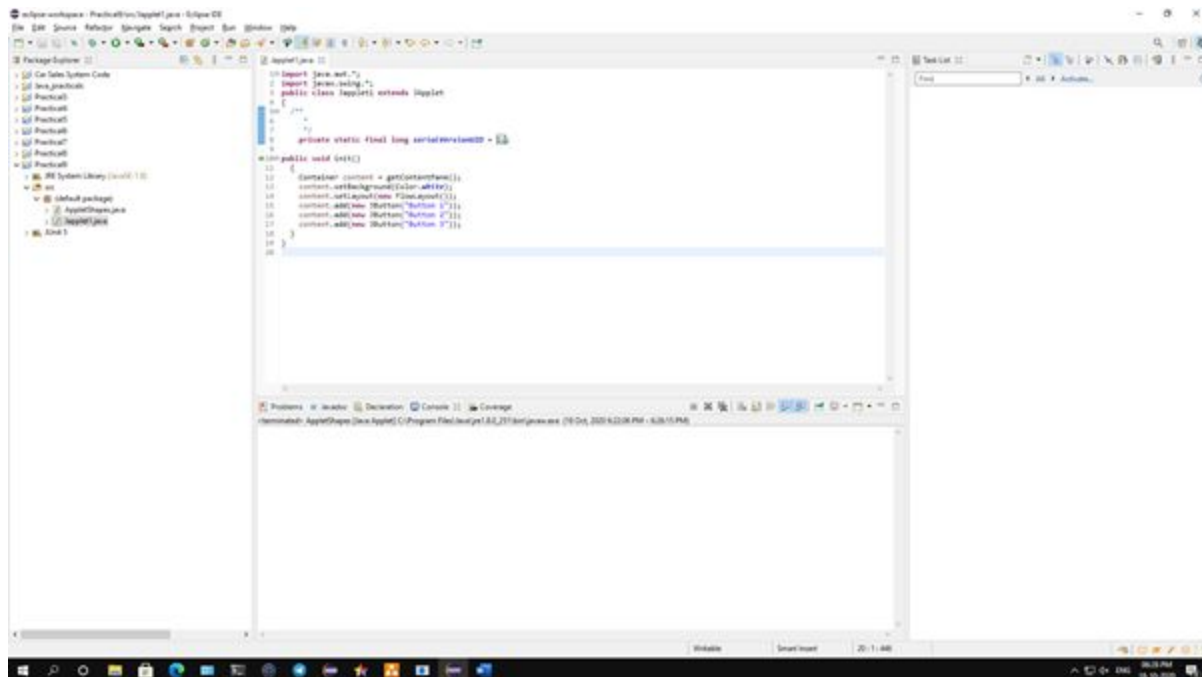


B) To demonstrate JApplet.

Aim: Write a program to demonstrate JApplet.

Description:

A JApplet is a Swing JPanel with a mission. It is a GUI container that has some extra structure to allow it to be used in the “alien” environment of a web browser. Applets also have a lifecycle that lets them act more like an application than a static component. We have imported the java awt and swing class, then we defined Japplet1 class which extends Japplet class, then we define the init function which shows the container and set background as white then we define a new button as button 1, button 2 and button 3. Then the output will show with gui that show Button 1, Button 2 and Button 3.

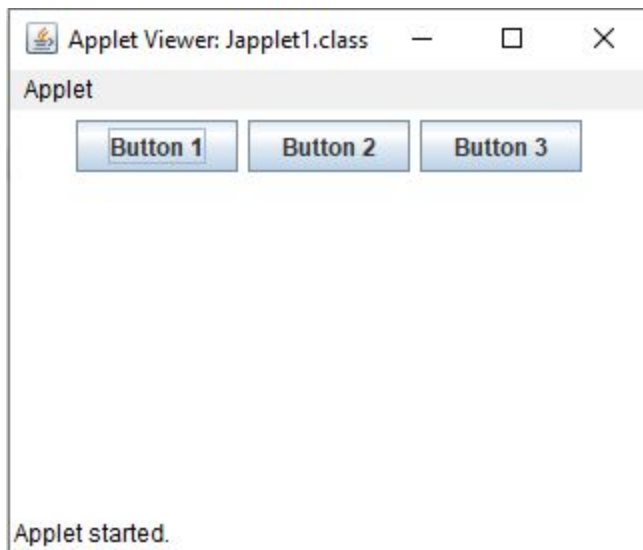


Conclusion: We have demonstrated JApplet in detail.

Code:

```
import java.awt.*;
import javax.swing.*;
public class Japplet1 extends JApplet
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public void init()
    {
        Container content = getContentPane();
        content.setBackground(Color.white);
        content.setLayout(new FlowLayout());
        content.add(new JButton("Button 1"));
        content.add(new JButton("Button 2"));
        content.add(new JButton("Button 3"));
    }
}
```

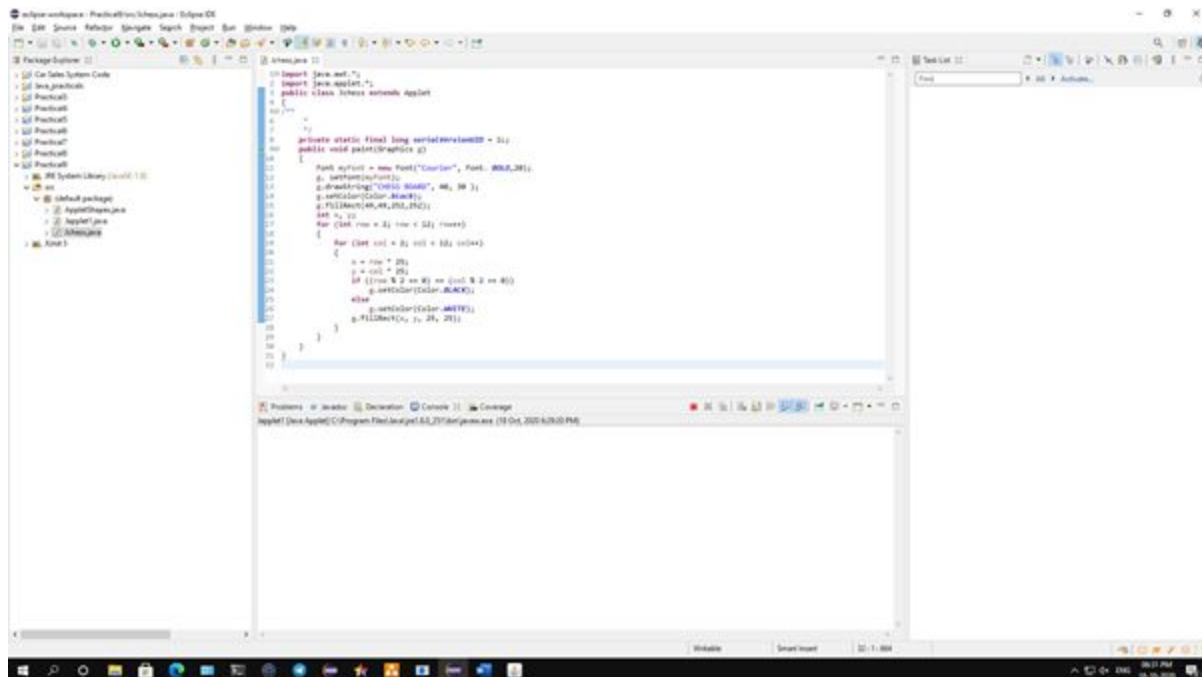
Output:

C) To Develop Chess Board using Applet.

Aim: Write a program to develop a Chess Board using Applet.

Description:

An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. Four methods in the Applet class give you the framework on which you build any serious applet they are: init, start, stop, destroy and paint. In this program we import the awt and swing classes followed by the Jchess class which extends the applet class, then we define the main function as paint which is used to define the shape and graphics. Then we set Courier as default font and colours as black and white in order to make a chess board, then we define the rows and columns and the chess board has been made in java using an applet.



Conclusion: We have developed a Chess Board using Applet.

Code:

```
import java.awt.*;

import java.applet.*;

public class Jchess extends Applet

{

    /**

        *

        */

    private static final long serialVersionUID = 1L;

    public void paint(Graphics g)

    {

        Font myFont = new Font("Courier", Font. BOLD,20);

        g.setFont(myFont);

        g.drawString("CHESS BOARD", 40, 30 );

        g.setColor(Color.black);

        g.fillRect(49,49,252,252);

        int x, y;

        for (int row = 2; row < 12; row++)

        {

            for (int col = 2; col < 12; col++)

            {

                x = row * 25;

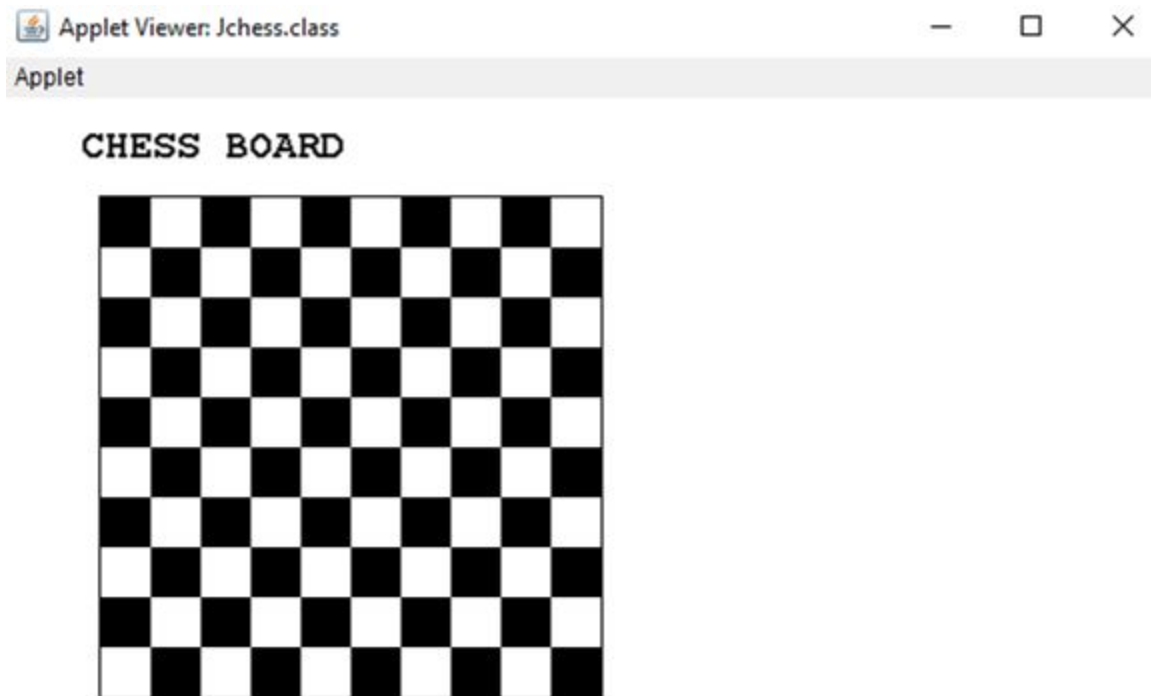
                y = col * 25;

                if ((row % 2 == 0) == (col % 2 == 0))

                    g.setColor(Color.BLACK);
```

```
        else  
            g.setColor(Color.WHITE);  
            g.fillRect(x, y, 25, 25);  
        }  
    }  
}
```

Output:



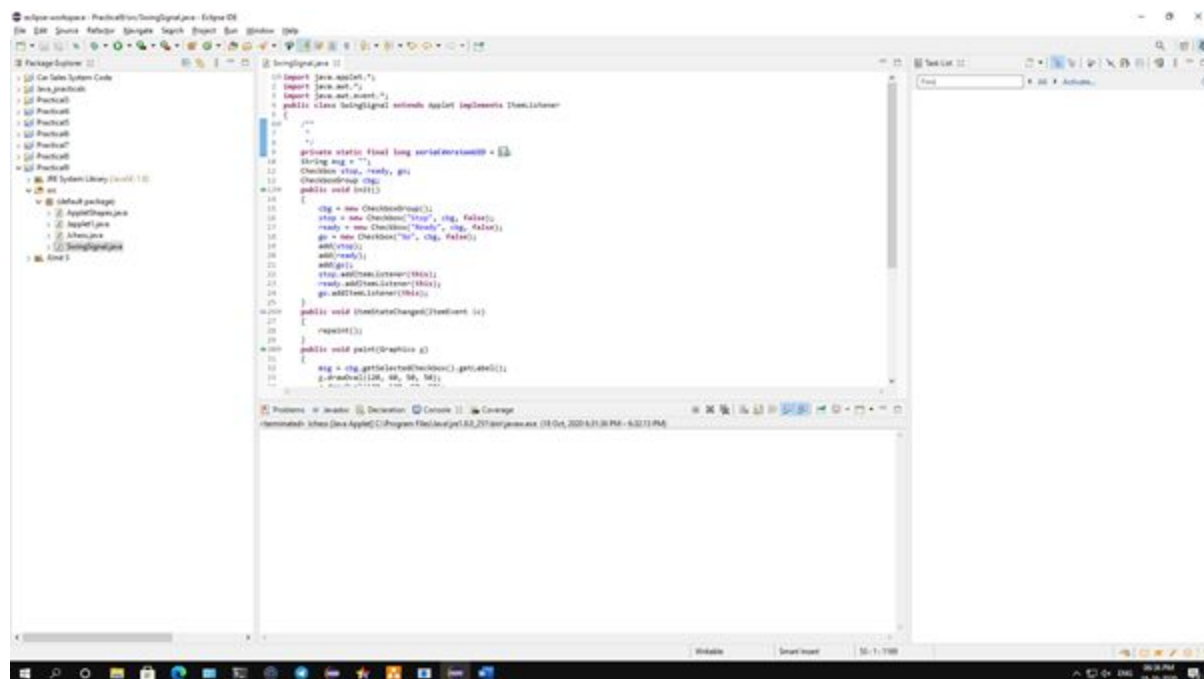
Applet started.

D) Traffic Signal Simulator using Swing Components.

Aim: Write a program to simulate Traffic Signal using Swing Components.

Description:

Swing is a part of the JFC (Java Foundation Classes). Building Graphical User Interface in Java requires the use of Swings. Swing Framework contains a large set of components which allow a high level of customization and provide rich functionalities, and is used to create window-based applications. Java swing components are lightweight, platform-independent, provide powerful components like tables, scroll panels, buttons, list, colour chooser, etc. In this program we imported the applet, awt and the swing classes followed by swingsignal that extends applet and implements itemlistener, then we define the main init functions and then we define 3 oval shapes and give them 3 colours as red, green and blue. The output will show a simulating traffic signal.



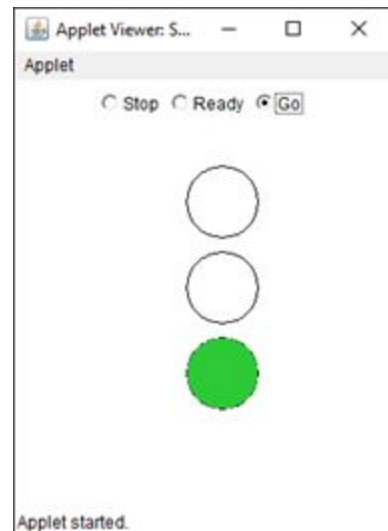
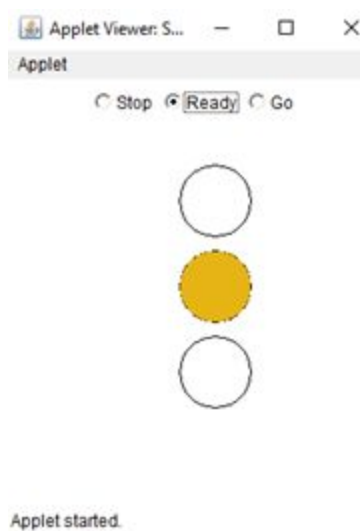
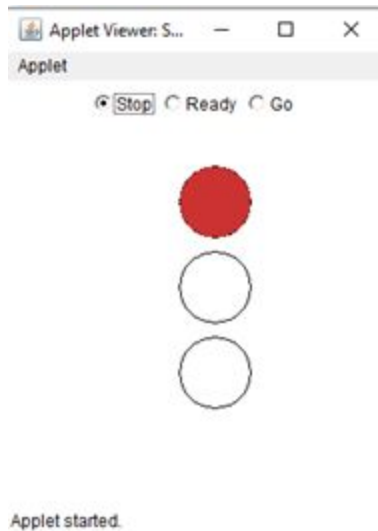
Conclusion: We have simulated a Traffic Signal using Swing Components.

Code:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class SwingSignal extends Applet implements ItemListener
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    String msg = "";
    Checkbox stop, ready, go;
    CheckboxGroup cbg;
    public void init()
    {
        cbg = new CheckboxGroup();
        stop = new Checkbox("Stop", cbg, false);
        ready = new Checkbox("Ready", cbg, false);
        go = new Checkbox("Go", cbg, false);
        add(stop);
        add(ready);
        add(go);
        stop.addItemListener(this);
        ready.addItemListener(this);
        go.addItemListener(this);
    }
    public void itemStateChanged(ItemEvent ie)
    {
        repaint();
    }
    public void paint(Graphics g)
    {
        msg = cbg.getSelectedCheckbox().getLabel();
        g.drawOval(120, 60, 50, 50);
        g.drawOval(120, 120, 50, 50);
    }
}
```

```
g.drawOval(120, 180, 50, 50);  
if(msg.equals("Stop"))  
{  
    g.setColor(new Color(204, 50, 50));  
    g.fillOval(120, 60, 50, 50);  
}  
else if(msg.equals("Ready"))  
{  
    g.setColor(new Color(231, 180, 22));  
    g.fillOval(120, 120, 50, 50);  
}  
else  
{  
    g.setColor(new Color(45, 201, 55));  
    g.fillOval(120, 180, 50, 50);  
}  
}  
}
```

Output:

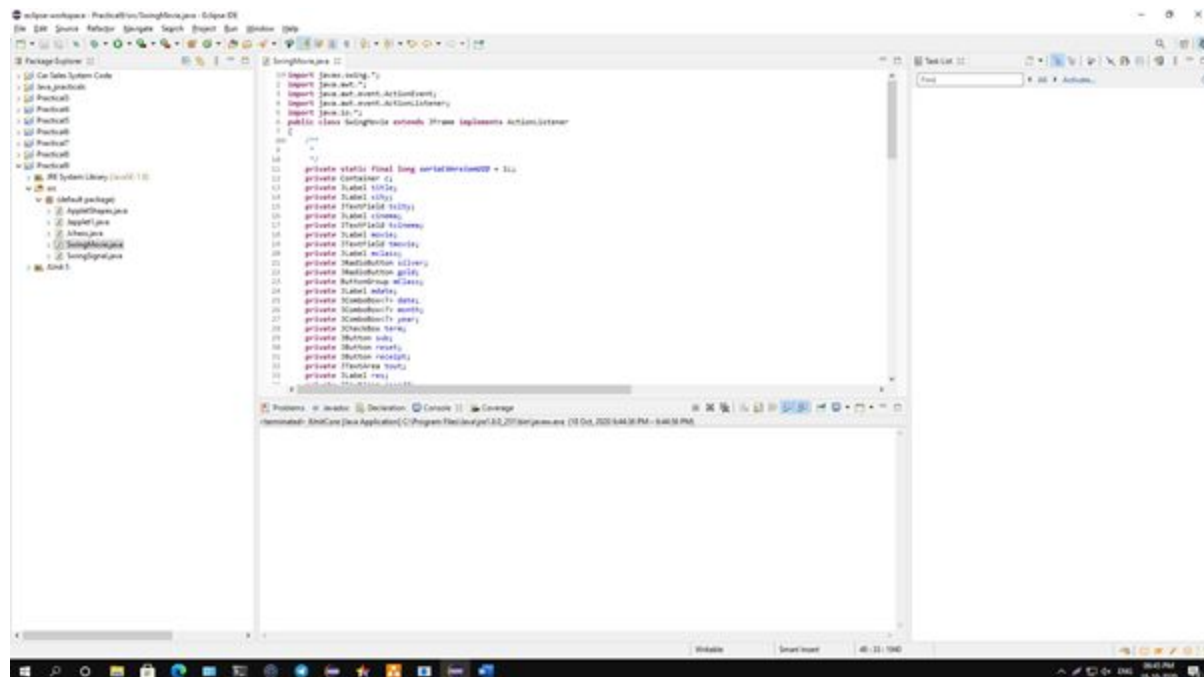


E) To design a form using Swings Component for Movie Ticket Booking application.

Aim: Write a program to design a form using Swings Component for Movie Ticket Booking application.

Description:

Swing is a part of the JFC (Java Foundation Classes). Building Graphical User Interface in Java requires the use of Swings. Swing Framework contains a large set of components which allow a high level of customization and provide rich functionalities, and is used to create window-based applications. Java swing components are lightweight, platform-independent, provide powerful components like tables, scroll panels, buttons, list, colour chooser, etc. In this program we imported java swing, awt and io classes followed by swingmovie class which extends JFrame that implements actionlistener classes, then we defined the main private classes which are required to make a box and list. We then defined the containers and text required to make an application. We also defined the title, name, city, class and so on. We also use try and catch keywords for exception handling then we use the new class for obtaining the data. The application will run and we can book a ticket.



Conclusion: We have designed a form using Swings Component for Movie Ticket Booking application.

Code:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.*;

public class SwingMovie extends JFrame implements ActionListener

{

    /**
     *
     */

    private static final long serialVersionUID = 1L;

    private Container c;

    private JLabel title;

    private JLabel city;

    private JTextField tcity;

    private JLabel cinema;

    private JTextField tcinema;

    private JLabel movie;

    private JTextField tmovie;

    private JLabel mclass;

    private JRadioButton silver;

    private JRadioButton gold;

    private ButtonGroup mClass;
```

```
private JLabel mdate;

private JComboBox<?> date;

private JComboBox<?> month;

private JComboBox<?> year;

private JCheckBox term;

private JButton sub;

private JButton reset;

private JButton receipt;

private JTextArea tout;

private JLabel res;

private JTextArea resadd;

private String dates[] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16",
"17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"};

private String months[] = { "Jan", "feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sup", "Oct",
"Nov", "Dec" };

private String years[] = { "1995", "1996", "1997", "1998", "1999", "2000", "2001",
"2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014", "2015", "2016", "2017", "2018", "2019", "2020" };

public SwingMovie() { setTitle("Movies"); setBounds(300, 90, 900, 600);
setDefaultCloseOperation(EXIT_ON_CLOSE); setResizable(false);

c = getContentPane(); c.setLayout(null);

title = new JLabel("Movie Booking");

title.setFont(new Font("Arial", Font.PLAIN, 30));

title.setSize(300, 30);

title.setLocation(300, 30);

c.add(title);

city = new JLabel("City");

city.setFont(new Font("Arial", Font.PLAIN, 20));

city.setSize(100, 20);
```

```
city.setLocation(100, 100);

c.add(city);

tcity = new JTextField(); tcity.setFont(new Font("Arial", Font.PLAIN, 15)); tcity.setSize(190, 20);
tcity.setLocation(200, 100); c.add(tcity);

cinema = new JLabel("Cinema"); cinema.setFont(new Font("Arial", Font.PLAIN, 20));
cinema.setSize(100, 20); cinema.setLocation(100, 150); c.add(cinema);

tcinema = new JTextField(); tcinema.setFont(new Font("Arial", Font.PLAIN, 15));
tcinema.setSize(150, 20); tcinema.setLocation(200, 150); c.add(tcinema);

movie = new JLabel("Movie"); movie.setFont(new Font("Arial", Font.PLAIN, 20));
movie.setSize(100, 20); movie.setLocation(100, 200); c.add(movie);

tmovie = new JTextField(); tmovie.setFont(new Font("Arial", Font.PLAIN, 15));
tmovie.setSize(150, 20); tmovie.setLocation(200, 200); c.add(tmovie);

mclass = new JLabel("Class"); mclass.setFont(new Font("Arial", Font.PLAIN, 20));
mclass.setSize(100, 20); mclass.setLocation(100, 250); c.add(mclass);

silver = new JRadioButton("Silver"); silver.setFont(new Font("Arial", Font.PLAIN, 15));
silver.setSelected(true); silver.setSize(75, 20); silver.setLocation(200, 250); c.add(silver);

gold = new JRadioButton("Gold"); gold.setFont(new Font("Arial", Font.PLAIN, 15));
gold.setSelected(false); gold.setSize(80, 20); gold.setLocation(275, 250); c.add(gold);

mClass = new ButtonGroup(); mClass.add(silver); mClass.add(gold);

mdate = new JLabel("Date"); mdate.setFont(new Font("Arial", Font.PLAIN, 20));
mdate.setSize(100, 20); mdate.setLocation(100, 300); c.add(mdate);

date = new JComboBox<Object>(dates);

date.setFont(new Font("Arial", Font.PLAIN, 15)); date.setSize(50, 20); date.setLocation(200,
300); c.add(date);

month = new JComboBox<Object>(months);

month.setFont(new Font("Arial", Font.PLAIN, 15)); month.setSize(60, 20);
month.setLocation(250, 300); c.add(month);

year = new JComboBox<Object>(years);

year.setFont(new Font("Arial", Font.PLAIN, 15)); year.setSize(60, 20); year.setLocation(320,
300); c.add(year);

term = new JCheckBox("Accept Terms And Conditions."); term.setFont(new Font("Arial",
Font.PLAIN, 15)); term.setSize(250, 20); term.setLocation(150, 400); c.add(term);

sub = new JButton("Submit");
```

```
        sub.setFont(new Font("Arial", Font.PLAIN, 15)); sub.setSize(100, 20);
        sub.setBackground(Color.green); sub.setLocation(150, 450); sub.addActionListener(this); c.add(sub);

        reset = new JButton("Reset");

        reset.setFont(new Font("Arial", Font.PLAIN, 15)); reset.setSize(100, 20);
        reset.setBackground(Color.white); reset.setLocation(270, 450); reset.addActionListener(this);
        c.add(reset);

        receipt = new JButton("Generate receipt"); receipt.setFont(new Font("Arial", Font.PLAIN, 15));
        receipt.setSize(200, 20);

        receipt.setBackground(Color.orange); receipt.setLocation(540, 450);
        receipt.addActionListener(this); c.add(receipt);

        tout = new JTextArea();

        tout.setFont(new Font("Arial", Font.PLAIN, 15)); tout.setSize(300, 250); tout.setLocation(500,
        100); tout.setLineWrap(true); tout.setEditable(false); c.add(tout);

        res = new JLabel("");

        res.setFont(new Font("Arial", Font.PLAIN, 20)); res.setSize(500, 25); res.setLocation(100, 500);
        c.add(res);

        resadd = new JTextArea();

        resadd.setFont(new Font("Arial", Font.PLAIN, 15)); resadd.setSize(200, 75);
        resadd.setLocation(580, 175); resadd.setLineWrap(true); c.add(resadd);

        setVisible(true);
    }

    @Override

    public void actionPerformed(ActionEvent e) { String receipt = ""; if (e.getSource() == sub) { if
    (term.isSelected()) { String data1;

    String data = "City : " + tcity.getText() + "\n" + "Cinema : " + tcinema.getText() + "\n" + "Movie : " +
    tmovie.getText() + "\n";

    if (silver.isSelected()) data1 = "Class : Silver" + "\n";

    else data1 = "Class : Gold" + "\n"; String data2 = "Date : " + (String)date.getSelectedItem()
    + "/" + (String)month.getSelectedItem()
    + "/" + (String)year.getSelectedItem() + "\n";

    receipt = data + data1 + data2; tout.setText(receipt); tout.setEditable(false);
```



```
res.setText("Registration Successfully..");

}

else { tout.setText(""); resadd.setText("");

res.setText("Please accept the" + " terms & conditions..");

}}

else if (e.getSource() == reset) { String def = ""; tcity.setText(def); tcinema.setText(def);
tmovie.setText(def);

tout.setText(def); term.setSelected(false); date.setSelectedIndex(0); month.setSelectedIndex(0);
year.setSelectedIndex(0); resadd.setText(def);

}

else if (e.getSource() == receipt) { FileWriter writer = null; try { writer = new
FileWriter("movie-receipt.txt");

} catch (IOException ioException) { ioException.printStackTrace();

}

BufferedWriter buffer = new BufferedWriter(writer); try { buffer.write(receipt);

} catch (IOException ioException) { ioException.printStackTrace();

}

try { buffer.close();

} catch (IOException ioException) { ioException.printStackTrace();

}}}}

class Registration

{

    private static SwingMovie f;


    public static void main(String[] args) throws Exception

    {

        setF(new SwingMovie());

    }

}
```

```
public static SwingMovie getF() {  
    return f;  
}  
  
public static void setF(SwingMovie f) {  
    Registration.f = f;  
}  
}
```

Output:

The screenshot shows a Java Swing window titled "Movies" with a "Movie Booking" form. The form contains the following fields and controls:

- City: Text field with "Chembur" entered.
- Cinema: Text field with "Fun Cinema" entered.
- Movie: Text field with "Avengers" entered.
- Class: Radio buttons for "Silver" (selected) and "Gold".
- Date: Three dropdown menus showing "17", "Aug", and "2017".
- A checkbox labeled "Accept Terms And Conditions." which is checked.
- Buttons: "Submit" (green), "Reset" (white), and "Generate receipt" (yellow).
- A message at the bottom: "Registration Successfully..".
- A summary box on the right displays: "City : Chembur", "Cinema : Fun Cinema", "Movie : Avengers", "Class : Silver", and "Date : 17/Aug/2017".

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 10		
Title Of Lab Practical: DATABASE PROGRAMMING				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

Practical No 10

A) To implement an online Exam or Shopping Cart application using swings and JDBC Type 1 Driver.

Aim: Write a program to implement an online Exam application using swings and JDBC Type 1 Driver.

Description:

JDBC drivers are client-side adapters that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers they are:

Type 1: JDBC-ODBC Bridge driver (Bridge)

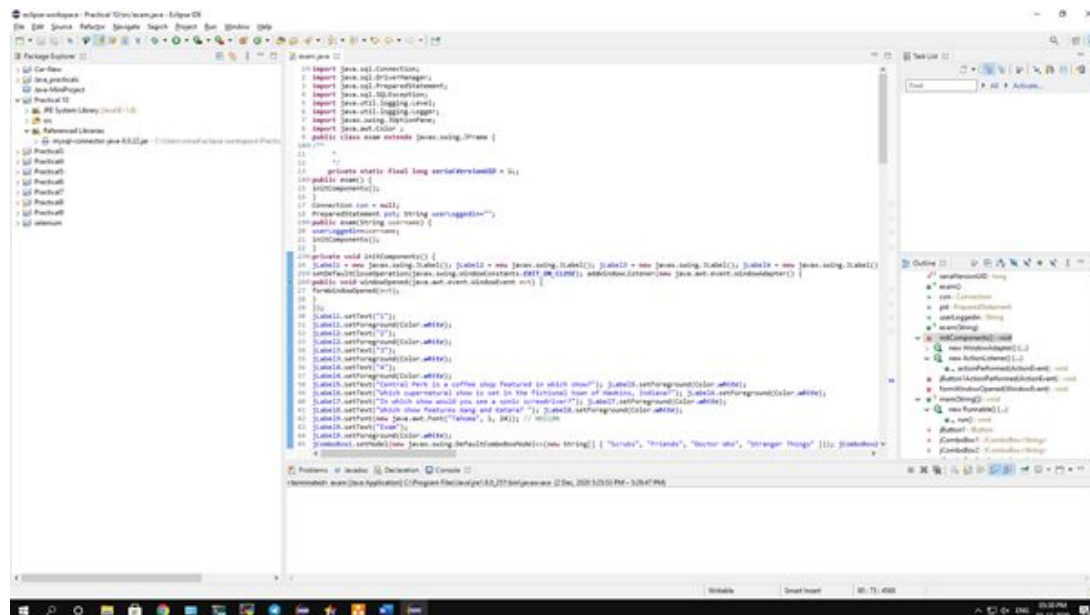
Type 2: Native-API/partly Java driver (Native)

Type 3: All Java/Net-protocol driver (Middleware)

Type 4: All Java/Native-protocol driver (Pure)

Type 1 JDBC Driver : JDBC-ODBC Bridge driver

The Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available. In this program we used Type J JDBC driver and imported all the required libraries and then we created a mcq type exam form for a single user and then we used a username which is already fixed and then we used the marks to show the marks obtained by the students based on correct mcq. Once you submit the exam form the scores will show and display on the screen to the user and the same will update in the database.



Conclusion: We have implemented an online Exam application using swings and JDBC Type 1 Driver.

Code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import java.awt.Color ;

public class exam extends javax.swing.JFrame {
    private static final long serialVersionUID = 1L;
    public exam() {
        initComponents();
    }
    Connection con = null;
    PreparedStatement pst; String userLoggedIn="";
    public exam(String username) {
        userLoggedIn=username;
        initComponents();
    }
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel(); jLabel2 = new javax.swing.JLabel(); jLabel3 = new
        javax.swing.JLabel(); jLabel4 = new javax.swing.JLabel(); jLabel5 = new javax.swing.JLabel(); jLabel6 =
        new javax.swing.JLabel(); jLabel7 = new javax.swing.JLabel(); jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel(); jComboBox1 = new javax.swing.JComboBox<>(); jComboBox2 =
        new javax.swing.JComboBox<>(); jComboBox3 = new javax.swing.JComboBox<>(); jComboBox4 = new
        javax.swing.JComboBox<>(); jButton1 = new javax.swing.JButton(); jLabel10 = new
        javax.swing.JLabel();
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE); addWindowListener(new
        java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });
        jLabel1.setText("1");
        jLabel1.setForeground(Color.white);
        jLabel2.setText("2");
        jLabel2.setForeground(Color.white);
        jLabel3.setText("3");
        jLabel3.setForeground(Color.white);
        jLabel4.setText("4");
        jLabel4.setForeground(Color.white);
        jLabel5.setText("Central Perk is a coffee shop featured in which show?");
        jLabel5.setForeground(Color.white);
        jLabel6.setText("Which supernatural show is set in the fictional town of Hawkins, Indiana?");
        jLabel6.setForeground(Color.white);
        jLabel7.setText("In which show would you see a sonic screwdriver?");
        jLabel7.setForeground(Color.white);
        jLabel8.setText("Which show features Aang and Katara? "); jLabel8.setForeground(Color.white);
```

```
jLabel9.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N
jLabel9.setText("Exam");
jLabel9.setForeground(Color.white);
jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Scrubs", "Friends",
"Doctor Who", "Stranger Things" }));
jComboBox2.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Doctor Who",
"Scrubs", "Friends", "Stranger Things" }));
jComboBox3.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Stranger Things",
"Doctor Who", "Scrubs", "Friends" }));
jComboBox4.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Friends", "Doctor
Who", "Scrubs", "Avatar - the last Airbender" })); jButton1.setText("SUBMIT");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
jLabel10.setFont(new java.awt.Font("Tahoma", 1, 36)); jLabel10.setText("jLabel10");
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setBackground(new Color(90, 113, 148));
getContentPane().setLayout(layout); layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1)
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel5)))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel2)
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel6)))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel3)
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel7)))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jButton1)
                    .addGap(18, 18, 18)
                    .addComponent(jLabel4)))
            .addContainerGap())
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```

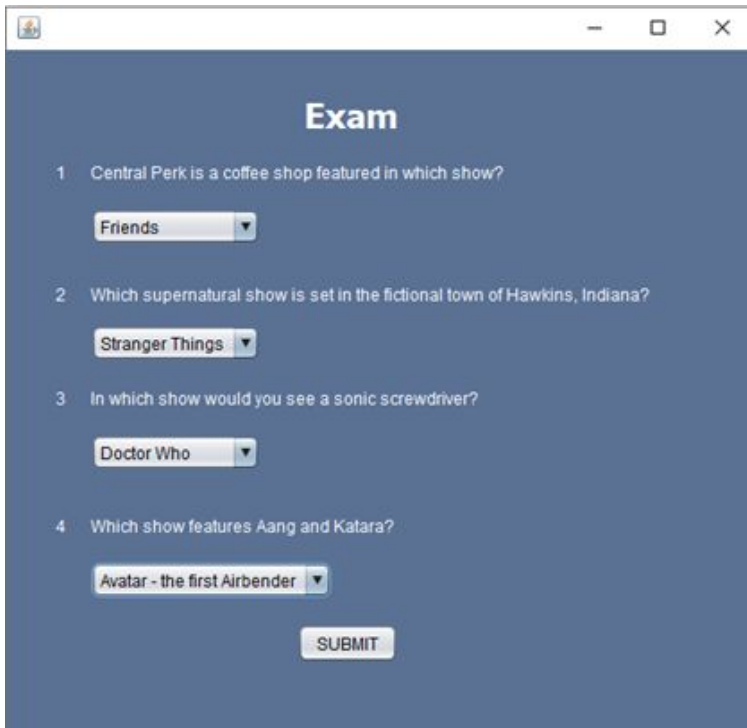
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabel8))))))
.addGroup(layout.createSequentialGroup())
.addGap(212, 212, 212)
.addComponent(jLabel9))
.addGroup(layout.createSequentialGroup())
.addGap(196, 196, 196)
.addComponent(jLabel10)))
.addContainerGap(75, Short.MAX_VALUE)
);
layout.setVerticalGroup( layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup())
.addContainerGap()
.addComponent(jLabel10)
.addGap(26, 26, 26)
.addComponent(jLabel9)
.addGap(18, 18, 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel1)
.addComponent(jLabel5))
.addGap(18, 18, 18)
.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(27, 27, 27)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel2)
.addComponent(jLabel6))
.addGap(14, 14, 14)
.addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel3)
.addComponent(jLabel7))
.addGap(18, 18, 18)
.addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(31, 31, 31)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel4)
.addComponent(jLabel8))
.addGap(18, 18, 18)
.addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)
.addComponent(jButton1)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE));
pack();
setLocationRelativeTo(null); }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
String answer1,answer2,answer3,answer4; int score=0;
answer1 = jComboBox1.getSelectedItem().toString(); answer2 =
jComboBox2.getSelectedItem().toString(); answer3 = jComboBox3.getSelectedItem().toString(); answer4
= jComboBox4.getSelectedItem().toString(); if(answer1.equals("Friends")){

```

```

score++; }
if(answer2.equals("Stranger Things")){ score++; }
if(answer3.equals("Doctor Who")){ score++; }
if(answer4.equals("Avatar - the last Airbender")){ score++; }
JOptionPane.showMessageDialog(this,userLoggedIn+" your score is : "+score); try {
Class.forName("com.mysql.cj.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://****/SAC?useSSL=false&serverTimezone=UTC", "****",
"****");
String query="update user set score = ? where username=?"; pst = con.prepareStatement(query);
pst.setInt(1,score); pst.setString(2,userLoggedIn); int r=pst.executeUpdate();
if(r==1){
JOptionPane.showMessageDialog(null, "Value has been updated");
}else
JOptionPane.showMessageDialog(null, "DB connection issue");
} catch (ClassNotFoundException | SQLException ex) {
Logger.getLogger(exam.class.getName()).log(Level.SEVERE, null, ex);
}
}
private void formWindowOpened(java.awt.event.WindowEvent evt) { jLabel10.setText(userLoggedIn);
}public static void main(String args[]) { try {
for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
if ("Nimbus".equals(info.getName())) {
javax.swing.UIManager.setLookAndFeel(info.getClassName()); break; }}
} catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(exam.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(exam.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(exam.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(exam.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
}
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() { new exam().setVisible(true); } });
private javax.swing.JButton jButton1;
private javax.swing.JComboBox<String> jComboBox1; private javax.swing.JComboBox<String>
jComboBox2; private javax.swing.JComboBox<String> jComboBox3; private
javax.swing.JComboBox<String> jComboBox4; private javax.swing.JLabel jLabel1; private
javax.swing.JLabel jLabel10; private javax.swing.JLabel jLabel2; private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4; private javax.swing.JLabel jLabel5; private javax.swing.JLabel
jLabel6; private javax.swing.JLabel jLabel7; private javax.swing.JLabel jLabel8; private
javax.swing.JLabel jLabel9;
}

```


Output:

The screenshot shows a web browser window with a dark blue background. At the top, the word "Exam" is centered in white. Below it, there are four numbered questions, each followed by a dropdown menu. The first question is "Central Perk is a coffee shop featured in which show?" with "Friends" selected. The second is "Which supernatural show is set in the fictional town of Hawkins, Indiana?" with "Stranger Things" selected. The third is "In which show would you see a sonic screwdriver?" with "Doctor Who" selected. The fourth is "Which show features Aang and Katara?" with "Avatar - the first Airbender" selected. At the bottom right, there is a "SUBMIT" button.

Exam

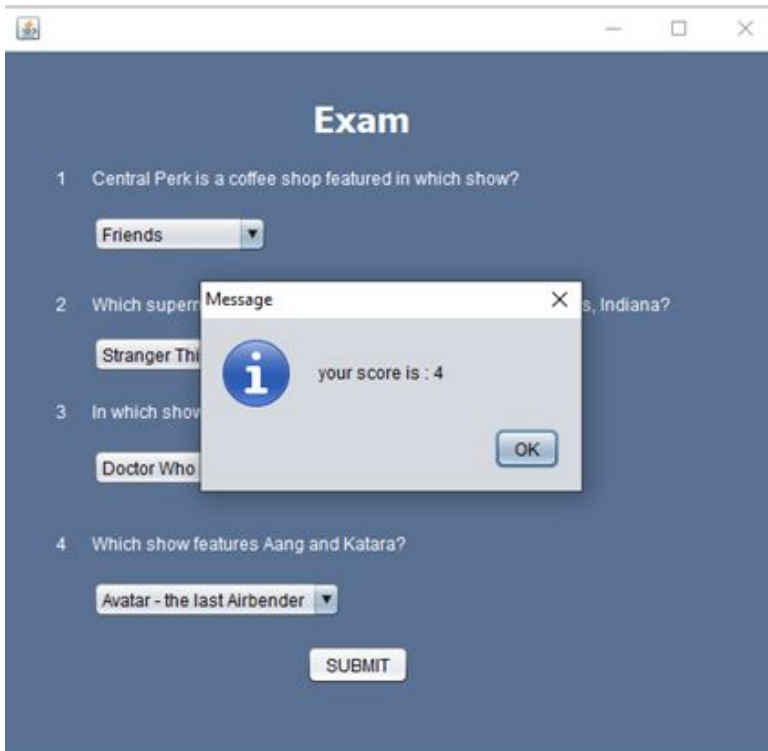
1 Central Perk is a coffee shop featured in which show?
Friends

2 Which supernatural show is set in the fictional town of Hawkins, Indiana?
Stranger Things

3 In which show would you see a sonic screwdriver?
Doctor Who

4 Which show features Aang and Katara?
Avatar - the first Airbender

SUBMIT



This screenshot shows the same "Exam" web application as the first image, but with a message dialog box open in the center. The dialog box has a title bar that says "Message" and a close button (X). It contains an information icon (i) and the text "your score is : 4". There is an "OK" button at the bottom right of the dialog box. The background application is slightly dimmed.

Exam

1 Central Perk is a coffee shop featured in which show?
Friends

2 Which supernatural show is set in the fictional town of Hawkins, Indiana?
Stranger Things

3 In which show would you see a sonic screwdriver?
Doctor Who

4 Which show features Aang and Katara?
Avatar - the last Airbender

SUBMIT

Message
your score is : 4
OK

B) To implement a program using JDBC Type 4 driver and mysql connector.

Aim: Write a program to implement a program using JDBC Type 4 driver and mysql connector.

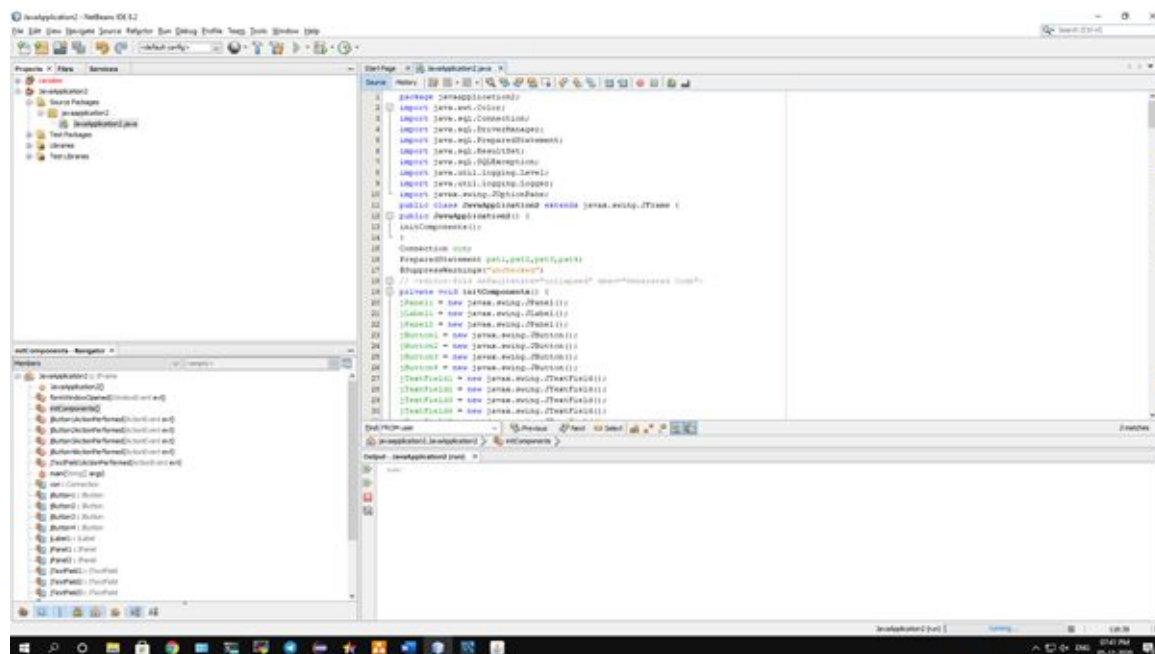
Description:

JDBC drivers are client-side adapters that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers they are:

- Type 1: JDBC-ODBC Bridge driver (Bridge)
- Type 2: Native-API/partly Java driver (Native)
- Type 3: All Java/Net-protocol driver (Middleware)
- Type 4: All Java/Native-protocol driver (Pure)

Type 4 JDBC Driver : All Java/Native-protocol driver (Pure)

The Type 4 JDBC Driver uses java networking libraries to communicate directly with the database server. In this program we imported Type 4 JDBC driver and all its required libraries and then we have created a window where all the major database operations are performed. Then we have used the insert button function in which it will insert the id and the username in the database, where the select button will show the username for the id we have inserted. Then we have defined the update button which will update the inserted id and then delete button will delete the records from the database.



Conclusion: We have implemented a program using JDBC Type 4 driver and mysql connector.

Code:

```
package javadb;
import java.awt.Color;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
public class javadb extends javax.swing.JFrame {
    private static final long serialVersionUID = 1L;
    public javadb() {
        initComponents();
        Connection con;
        PreparedStatement pst1,pst2,pst3,pst4;
        private void initComponents() {
            jPanel1 = new javax.swing.JPanel();
            jLabel1 = new javax.swing.JLabel();
            jPanel2 = new javax.swing.JPanel();
            jButton1 = new javax.swing.JButton();
            jButton2 = new javax.swing.JButton();
            jButton3 = new javax.swing.JButton();
            jButton4 = new javax.swing.JButton();
            jTextField1 = new javax.swing.JTextField();
            jTextField2 = new javax.swing.JTextField();
            jTextField3 = new javax.swing.JTextField();
            jTextField4 = new javax.swing.JTextField();
            jTextField5 = new javax.swing.JTextField();
            jTextField6 = new javax.swing.JTextField();
            setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
            addWindowListener(new java.awt.event.WindowAdapter() {
                @Override
                public void windowOpened(java.awt.event.WindowEvent evt)
                { formWindowOpened(evt); }
            });
            jLabel1.setFont(new java.awt.Font("Tahoma", 1, 28));
            jLabel1.setText("DATABASE OPERATIONS");
            jLabel1.setForeground(Color.darkGray);
            javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
            jPanel1.setLayout(jPanel1Layout);
            jPanel1Layout.setHorizontalGroup(
                jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(76, 76, 76)
                        .addComponent(jLabel1)
                        .addContainerGap())
            );
            jPanel1Layout.setVerticalGroup(
                jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(76, 76, 76)
                        .addComponent(jLabel1)
                        .addContainerGap())
            );
        }
    }
}
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup())
.addContainerGap()
.addComponent(jLabel1)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jButton1.setText("SELECT");
jButton1.addActionListener((java.awt.event.ActionEvent evt) -> {
jButton1ActionPerformed(evt);
});
jButton2.setText("UPDATE");
jButton2.addActionListener((java.awt.event.ActionEvent evt) -> {
jButton2ActionPerformed(evt);
});
jButton3.setText("DELETE");
jButton3.addActionListener((java.awt.event.ActionEvent evt) -> {
jButton3ActionPerformed(evt);
});
jButton4.setText("INSERT");
jButton4.addActionListener((java.awt.event.ActionEvent evt) -> {
jButton4ActionPerformed(evt);
});
jTextField1.setText("Please enter the id for name");
jTextField1.addActionListener((java.awt.event.ActionEvent evt) -> {
jTextField1ActionPerformed(evt);
});
jTextField2.setText("Please enter id");
jTextField3.setText("Please enter the id to be deleted");
jTextField4.setText("Name to be updated");
jTextField5.setText("Enter the id");
jTextField6.setText("Enter the username");
javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel2Layout.createSequentialGroup()
.addGroup(jPanel2Layout.createSequentialGroup()
.addGap(44, 44, 44)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jButton1)
.addComponent(jButton2)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
.addComponent(jButton4, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
.addComponent(jButton3, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
.addGap(24, 24, 24)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jTextField1)
.addComponent(jTextField3)
.addGroup(jPanel2Layout.createSequentialGroup()
.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jTextField4))
.addGroup(jPanel2Layout.createSequentialGroup())
.addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jTextField6))
));
jPanel2Layout.setVerticalGroup(
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel2Layout.createSequentialGroup())
.addGap(43, 43, 43)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton1)
.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton2)
.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton3)
.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton4)
.addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addContainerGap(32, Short.MAX_VALUE))
);
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup())
.addContainerGap()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup())
.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```

javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addContainerGap())
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup())
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE) .addContainerGap(19,
Short.MAX_VALUE));
pack();
setLocationRelativeTo(null); }
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{ try { int id=Integer.parseInt(jTextField3.getText());
Class.forName("com.mysql.cj.jdbc.Driver");
con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/practical10?serverTimezone=UTC","root","");
String query3="DELETE FROM users WHERE id = ?";
pst3 = con.prepareStatement(query3);
pst3.setInt(1,id);
int r3=pst3.executeUpdate();
if(r3==1){
JOptionPane.showMessageDialog(null,"The details have been deleted"); }
else
JOptionPane.showMessageDialog(null,"Some connection issue"); }
catch (ClassNotFoundException | SQLException ex)
{
Logger.getLogger(javadb.class.getName()).log(Level.SEVERE, null, ex); } }
private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{ try {
int id=Integer.parseInt(jTextField1.getText());
Class.forName("com.mysql.cj.jdbc.Driver");
con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/practical10?serverTimezone=UTC","root","");
String query="select username from users where id = ?";
pst1 = con.prepareStatement(query);
pst1.setInt(1,id);
ResultSet r=pst1.executeQuery();
while(r.next()){
JOptionPane.showMessageDialog(null,"The username of id : "+id+" is : "+r.getString(1)); } }
catch (ClassNotFoundException | SQLException ex)
{
Logger.getLogger(javadb.class.getName()).log(Level.SEVERE, null, ex); } }
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{

```



```

try {
int id=Integer.parseInt(jTextField2.getText());
String updatedName=jTextField4.getText();
Class.forName("com.mysql.cj.jdbc.Driver");
con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/practical10?serverTimezone=UTC","root","");
String query2="update users set username = ? where id = ?";
pst2 = con.prepareStatement(query2);
pst2.setInt(2,id);
pst2.setString(1,updatedName);
int r1=pst2.executeUpdate();
if(r1==1){
JOptionPane.showMessageDialog(null,"Username has been updated");
}
else
JOptionPane.showMessageDialog(null,"Some connection issue");
}
catch (ClassNotFoundException | SQLException ex) {
Logger.getLogger(javadb.class.getName()).log(Level.SEVERE, null, ex);
}
}

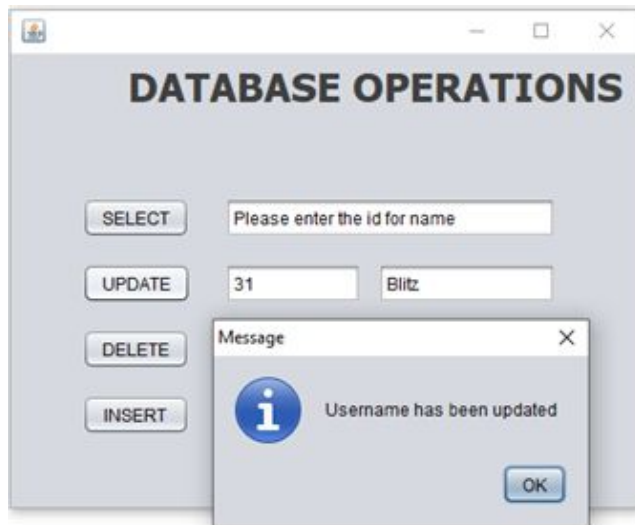
private void formWindowOpened(java.awt.event.WindowEvent evt) {}
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
try {
int ID=Integer.parseInt(jTextField5.getText());
String name=jTextField6.getText();
Class.forName("com.mysql.cj.jdbc.Driver");
con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/practical10?serverTimezone=UTC","root","");
String query2="INSERT INTO users VALUES(" + ID + "," + name + ")";
pst4 = con.prepareStatement(query2);
int r1=pst4.executeUpdate();
if(r1==1){
JOptionPane.showMessageDialog(null,"Records inserted"); }
else
JOptionPane.showMessageDialog(null,"Some connection issue"); }
catch (ClassNotFoundException | SQLException ex) {
Logger.getLogger(javadb.class.getName()).log(Level.SEVERE, null, ex); } }

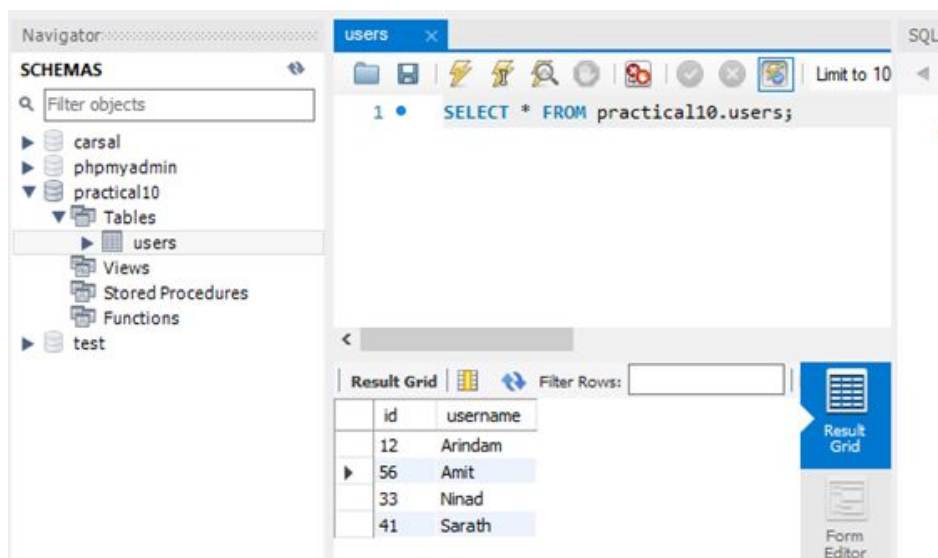
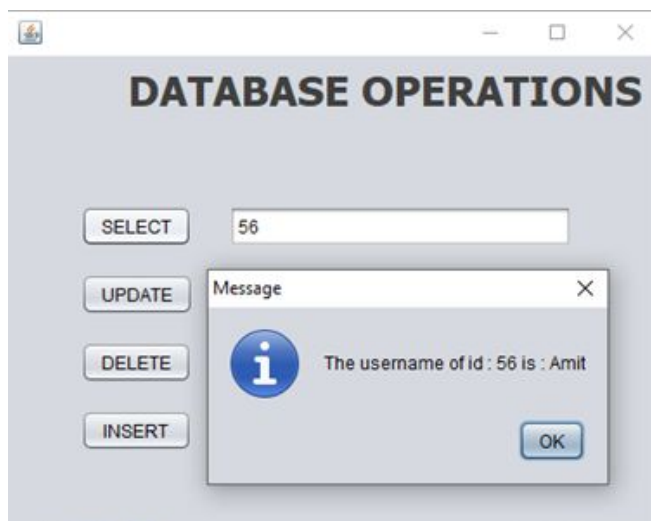
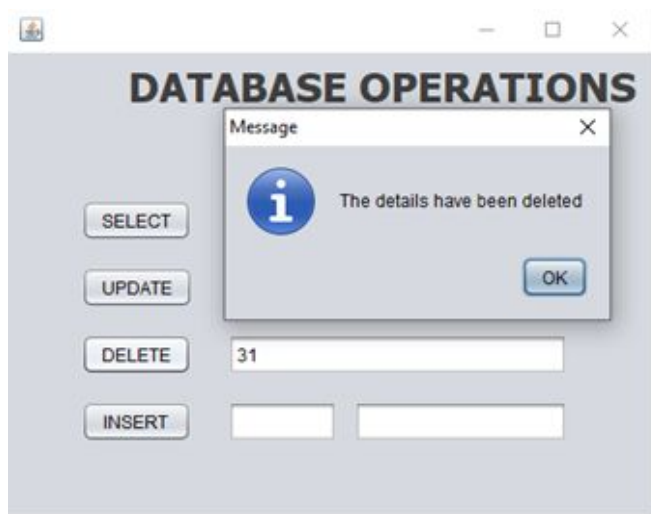
public static void main(String args[])
{ try {
for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels())
{
if ("Nimbus".equals(info.getName()))
{ javax.swing.UIManager.setLookAndFeel(info.getClassName());
break; } } }
catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
javax.swing.UnsupportedLookAndFeelException ex)
{
java.util.logging.Logger.getLogger(javadb.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
java.awt.EventQueue.invokeLater(() -> {
new javadb().setVisible(true);

```

```
});  
}  
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton2;  
private javax.swing.JButton jButton3;  
private javax.swing.JButton jButton4;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JPanel jPanel1;  
private javax.swing.JPanel jPanel2;  
private javax.swing.JTextField jTextField1;  
private javax.swing.JTextField jTextField2;  
private javax.swing.JTextField jTextField3;  
private javax.swing.JTextField jTextField4;  
private javax.swing.JTextField jTextField5;  
private javax.swing.JTextField jTextField6;  
}
```

Output:





Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 11		
Title Of Lab Practical: WEB DEVELOPMENT USING SERVLETS				
DOP:		DOS:		
CO Mapped: CO1,CO2	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

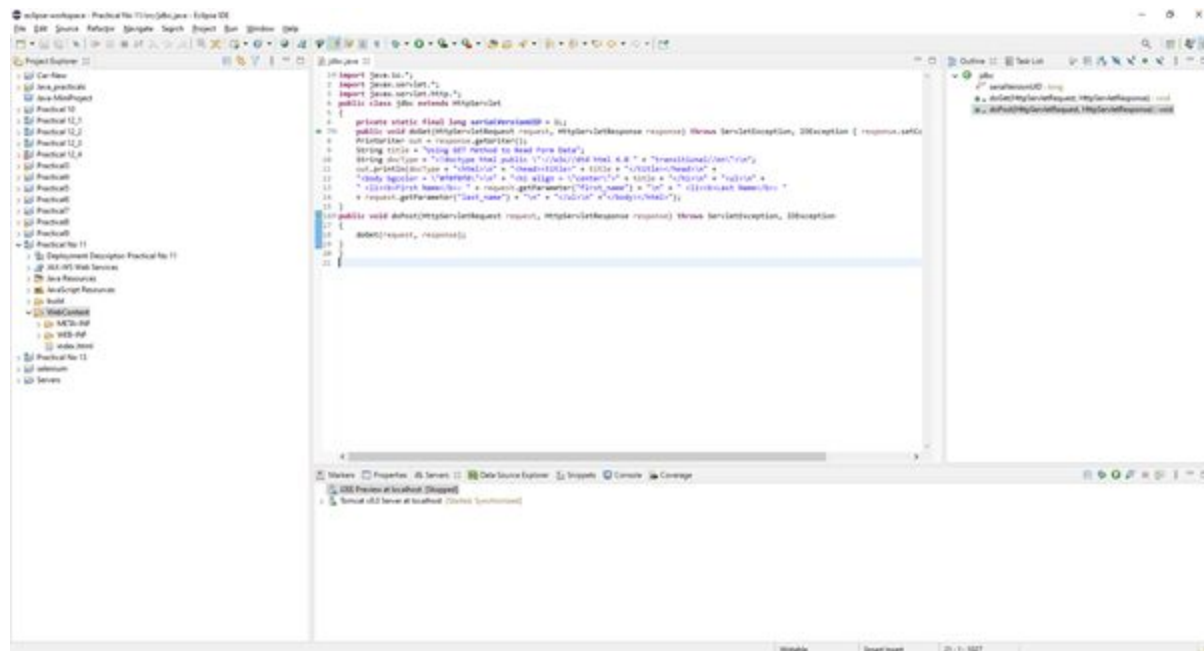
Practical No 11

A) To design a simple web-based interface to a currency converter application. The interface should consist of a title, suitable instructions, and a form for entering the amount to be converted and an optional currency rate. Use text fields for entering the amount and rate. Use the POST method to submit the form.

Aim: Write a program to design a simple web-based interface to a currency converter application.

Description:

Java Web Application is used to create dynamic websites. Java provides support for web applications through Servlets and JSPs. We can create a website with static HTML pages but when we want information to be dynamic, we need web application. In the following program we designed a simple web-based interface for converting currency. We defined the main functionality of html and java. We first imported java.io and then the java servlet package which helps in building this application then we extended the jdbc class with httpsservlet class. Then we defined the public void as doget and throw exceptions which will throw an exception if the program fails to run. Then we use index.html which contains mains html functions as head body and then we use a script which helps in displaying the webpage.



Conclusion: We have designed a simple web-based interface to a currency converter application.

Code:

jdbc.java

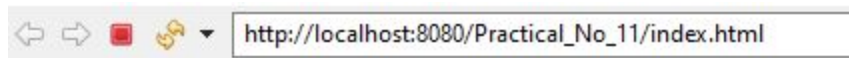
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class jdbc extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException { response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Using GET Method to Read Form Data";
        String docType = "<!doctype html public \"-//w3c//dtd html 4.0 \" + \"transitional//en\">\n";
        out.println(docType + "<html>\n" + "<head><title>" + title + "</title></head>\n" + "<body bgcolor =
\"#f0f0f0\">\n" + "<h1 align = 'center'>" + title + "</h1>\n" + "<ul>\n" + "<li><b>First Name</b>: " +
request.getParameter("first_name") + "\n" + "<li><b>Last Name</b>: " +
request.getParameter("last_name") + "\n" + "</ul>\n" + "</body></html>");
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
    {
        doGet(request, response);
    }
}
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Currency Converter</title>
</head>
<body>
<h3>Currency Converter</h3>
<h3>Rates for Conversion</h3>
<h2>Dollar : 74.12</h2>
<h2>Egyptian Pound: 4.29</h2>
<h2>Qatari Rial : 20.36</h2>
<p>Type amount in RS to convert the amount:</p>
<p><label>Rupees</label>
<input id="input_RS" type="number" placeholder="Insert Amount in RS"
oninput="CurrencyConverter(this.value)" onchange="CurrencyConverter(this.value)">
<p>Dollars : <span id="USDollar"></span></p>
<script>
function CurrencyConverter(valNum)
{
```

```
        document.getElementById("USDollar").innerHTML=valNum*74.12;
    }
    function CurrencyConverter1(valNum1)
    {
        document.getElementById("Egyption_Pound").innerHTML=valNum1*4.29;
    }
    function CurrencyConverter2(valNum2)
    {
        document.getElementById("Qatari_Rial").innerHTML=valNum2*20.36;
    }
</script>
</body>
</html>
```

Output:



Currency Converter

Rates for Conversion

Dollar : 74.12

Egyption Pound: 4.29

Qatari Rial : 20.36

Type amount in RS to convert the amount:

Rupees

Dollars : 370.6

Code:**cookies.java**

```

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class cookies extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    private int hitCount;
    public void init()
    {
        hitCount = 0;
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        hitCount++;
        PrintWriter out = response.getWriter();
        String title = "Total Number of Hits";
        String docType = "<!doctype html public '-//w3c//dtd html 4.0 ' +
\"transitional//en\">\n";
        out.println(docType + "<html>\n" + "<head><title>" + title + "</title></head>\n" +
"<body bgcolor = \"#f0f0f0\">\n" +
        "<h1 align = \"center\">" + title + "</h1>\n" + "<h2 align = \"center\">" + hitCount +
"</h2>\n" + "</body></html>" );
    }
    public void destroy()
    {
    }
}

```

servlet.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
    <display-name>Practical No 11_2</display-name>
    <servlet>
        <servlet-name>visitCounter</servlet-name>
        <servlet-class>visitCounter</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>visitCounter</servlet-name>
        <url-pattern>/visitCounter</url-pattern>
    </servlet-mapping>
</web-app>

```

Output:



Total Number of Hits

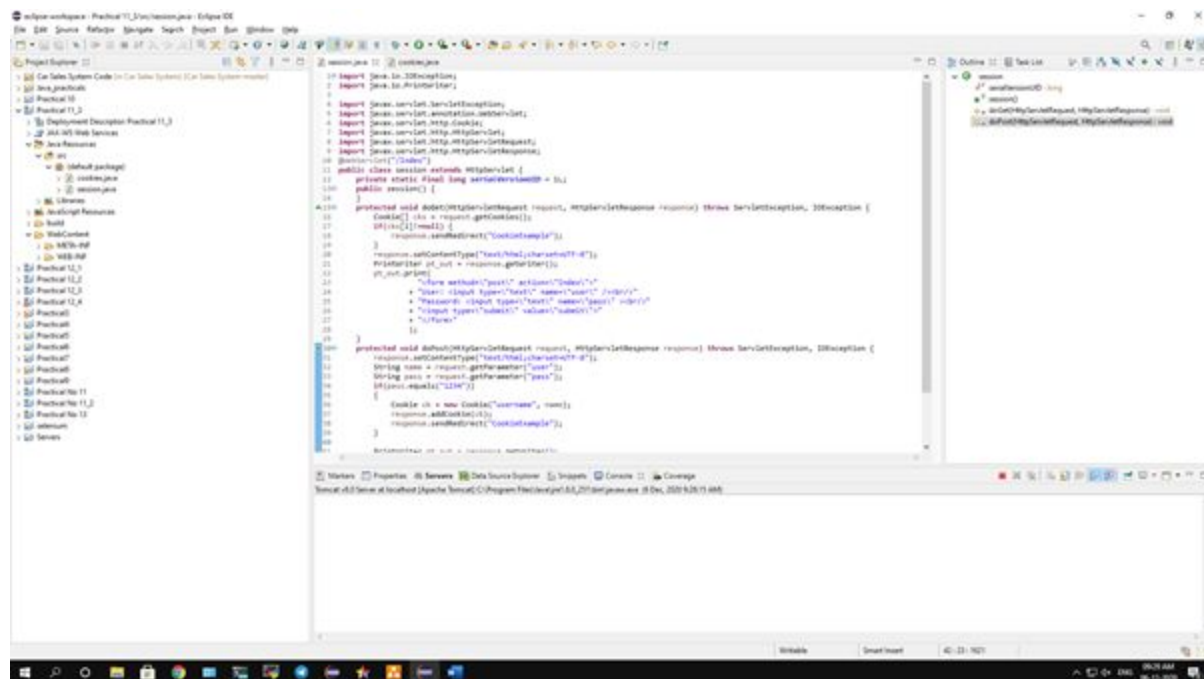
6

C) To implement a program for Session management in a servlet where the session is maintained under username.

Aim: Write a program to implement Session management in servlet.

Description:

A Cookie is a small piece of data that is exchanged between a server and a client. Whenever a client sends a request, the server will send a cookie containing the required data and the client can send back the cookie with its next request. In session management, Tomcat creates a session id whenever the client's first request gets to the server (However, other servlet containers may behave differently). Then it inserts this session id into a cookie with a name JSESSIONID and sends along with the response. After receiving the response with the cookie, the client can send the received cookie in its next request so that the server will identify the session using session id that resides in the JSESSIONID cookie. In this program we created 2 files as cookies.java and session.java. We imported all the sessions and defined its main functions then we created a public class that will extend the servlet class. Then we used a form method to define users and passwords. Once we type the username and password and if the username is correct it will display the output, if the password is not correct it will display as wrong password.



Conclusion: We have implemented a program for Session management in a servlet where the session is maintained under username.

Code:

session.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/Index")
public class session extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public session() {
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        Cookie[] cks = request.getCookies();
        if(cks[1]!=null) {
            response.sendRedirect("CookieExample");
        }
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter pt_out = response.getWriter();
        pt_out.print(
            "<form method='post' action='Index'>"
            + "User: <input type='text' name='user' /><br/>"
            + "Password: <input type='text' name='pass' /><br/>"
            + "<input type='submit' value='submit'>"
            + "</form>"
        );
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String name = request.getParameter("user");
        String pass = request.getParameter("pass");
        if(pass.equals("1234"))
        {
            Cookie ck = new Cookie("username", name);
            response.addCookie(ck);
            response.sendRedirect("CookieExample");
        }

        PrintWriter pt_out = response.getWriter();
        pt_out.append(
            "<h3>Wrong password buddy!!!</h3>"
        );
    }
}
```

```
}  
}
```

cookies.java

```
import java.io.IOException;  
import java.io.PrintWriter;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.Cookie;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
@WebServlet("/CookieExample")  
public class cookies extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
    public cookies() {  
        super();  
    }  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
        response.setContentType("text/html;charset=UTF-8");  
        PrintWriter out = response.getWriter();  
        Cookie[] cks = request.getCookies();  
        out.println("Welcome " + cks[1].getValue());  
    }  
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
  
        doGet(request, response);  
    }  
}
```

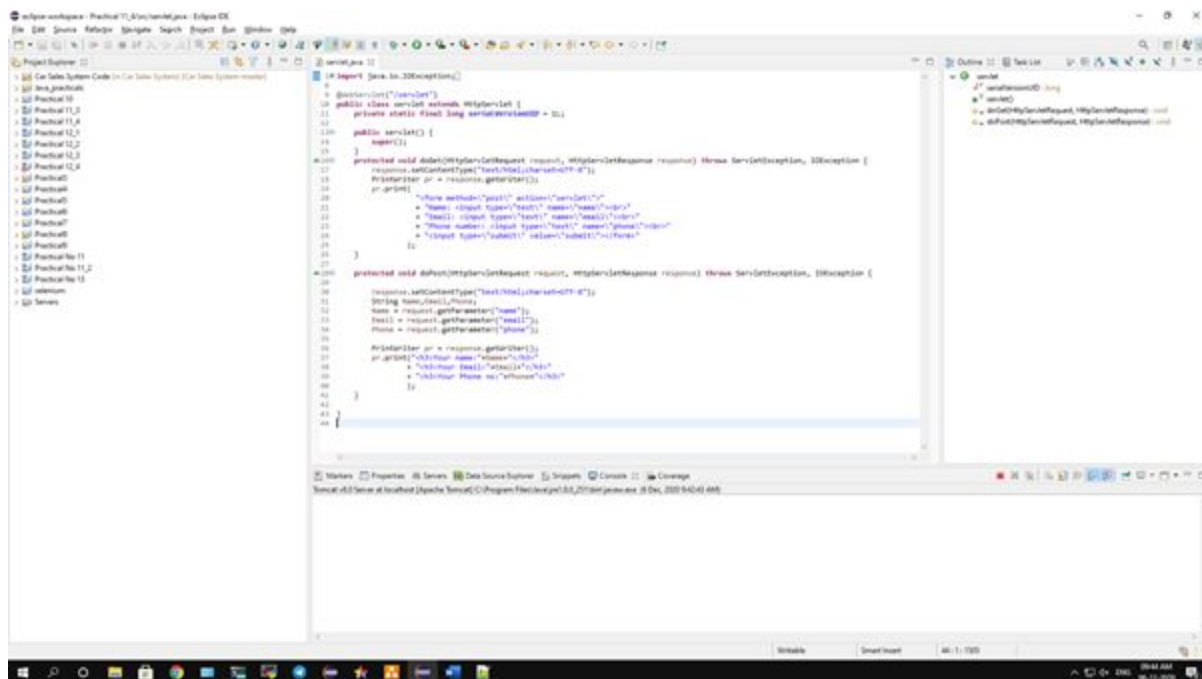
Output:

D) To implement a sample program to handle post methods in servlet.

Aim: Write a program to implement a sample program to handle post methods in servlet.

Description:

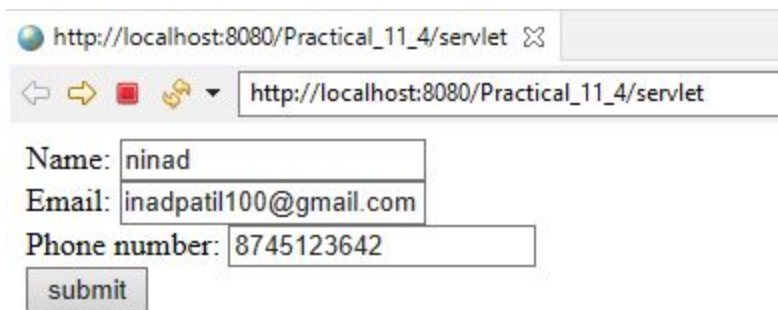
A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. In this program we have imported the servlet classes and then we define the servlet path using `@webserlet` method then we defined the public class and then we use form function. Then we define a protected function and then the output is displayed.



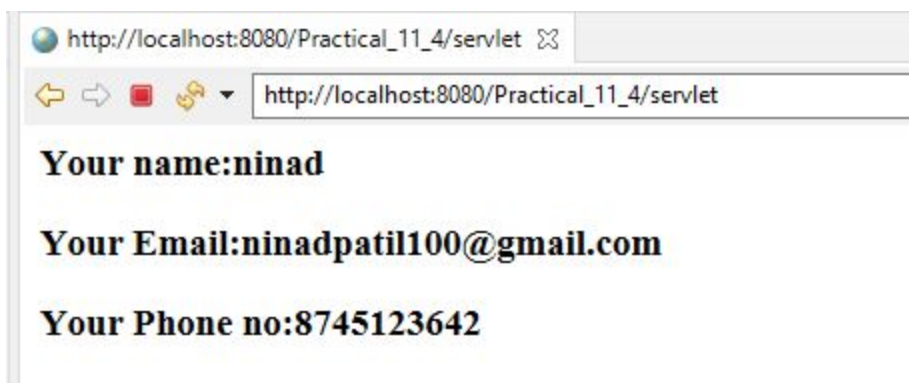
Conclusion: We have implemented a sample program to handle post methods in servlet.

Code:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/servlet")
public class servlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public servlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter pr = response.getWriter();
        pr.print(
            "<form method='post' action='servlet'>"
            + "Name: <input type='text' name='name'><br>"
            + "Email: <input type='text' name='email'><br>"
            + "Phone number: <input type='text' name='phone'><br>"
            + "<input type='submit' value='submit'></form>"
        );
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String Name,Email,Phone;
        Name = request.getParameter("name");
        Email = request.getParameter("email");
        Phone = request.getParameter("phone");
        PrintWriter pr = response.getWriter();
        pr.print("<h3>Your name:"+Name+"</h3>"
            + "<h3>Your Email:"+Email+"</h3>"
            + "<h3>Your Phone no:"+Phone+"</h3>"
        );
    }
}
```

Output:

A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/Practical_11_4/servlet`. Below the address bar, there are three input fields: "Name:" with the value "ninad", "Email:" with the value "inadpatil100@gmail.com", and "Phone number:" with the value "8745123642". Below these fields is a "submit" button.



A screenshot of a web browser window showing the output of the form submission. The address bar shows the URL `http://localhost:8080/Practical_11_4/servlet`. Below the address bar, there are three lines of text: "Your name:ninad", "Your Email:ninadpatil100@gmail.com", and "Your Phone no:8745123642".

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 12		
Title Of Lab Practical: WEB DEVELOPMENT USING JSP				
DOP:		DOS:		
CO Mapped: CO1,CO2	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

Practical No 12

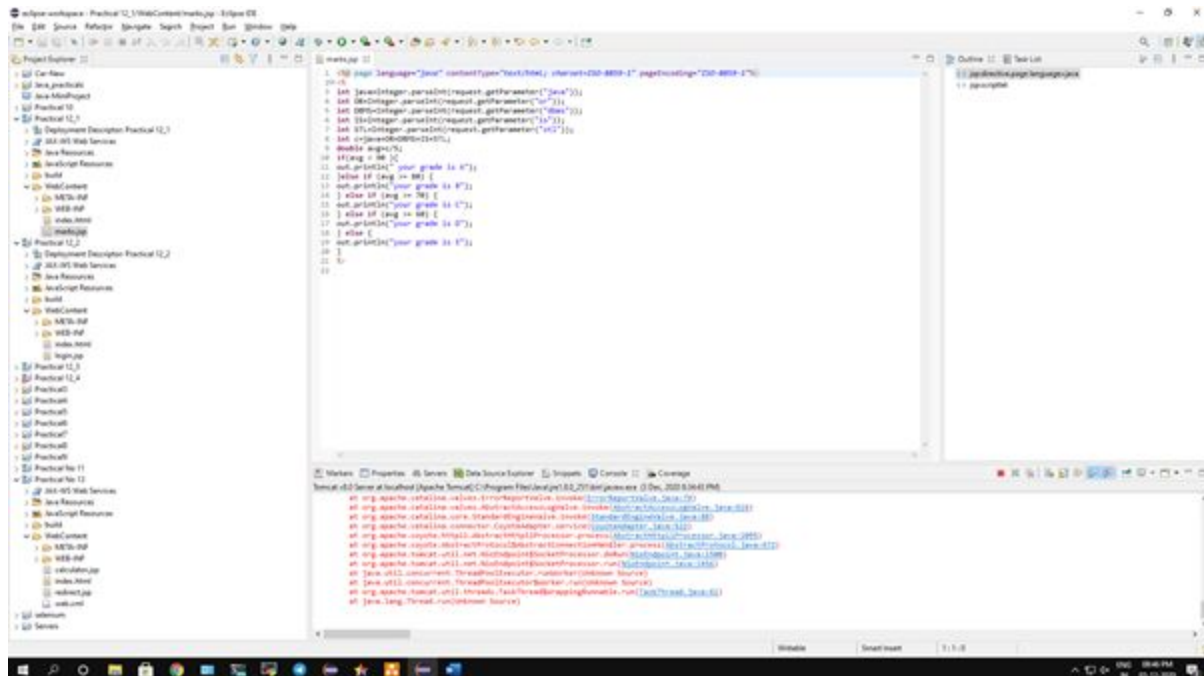
A) To design a form and use of JSP Scripting Element and JSP Directive. Display Grade of a student by accepting marks in five subjects.

Aim: Write a program to design a form and use of JSP Scripting Element and JSP Directive. Display Grade of a student by accepting marks in five subjects.

Description:

JSP scripting allows you to insert Java code into Java Servlet generated from JSP page. These are the scripting elements: comment, expression, scriptlet, declaration and expression language. A JSP directive affects the overall structure of the servlet class. Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.

In the following program we have defined the html functions along with forms and input type in index.html file. Then we have defined directive and scripting elements in marks.jsp file which start with the int subject name defines the subject for an integer so it will read the integer and get the output. Once we input the marks for a particular subject it will read and calculate the total marks and then it will show the grade based on the average of marks and then it will display the grade on the screen.



Conclusion: We have designed a form and use of JSP Scripting Element and JSP Directive. Display Grade of a student by accepting marks in five subjects.

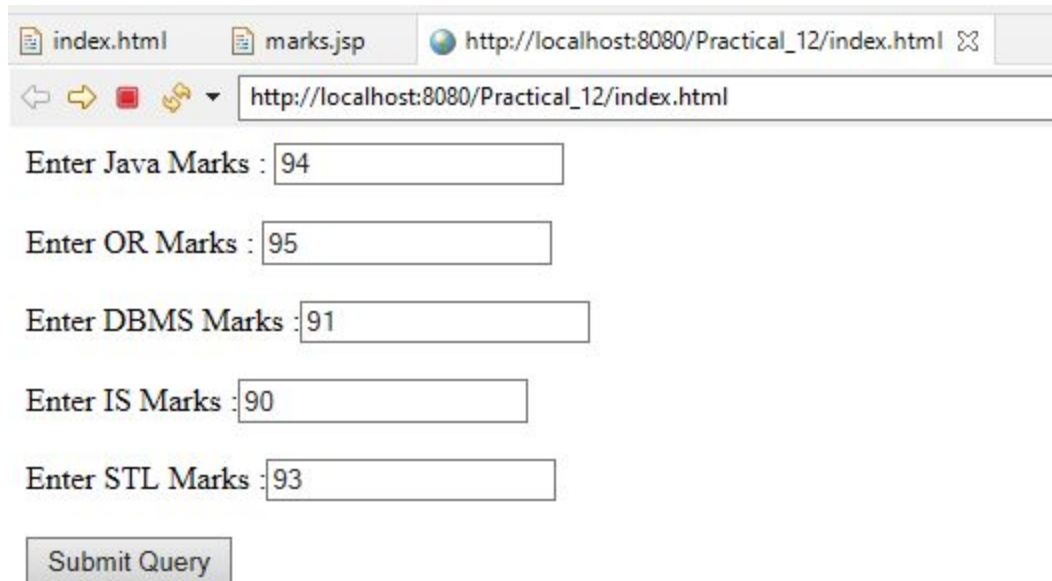
Code:

marks.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%
int java=Integer.parseInt(request.getParameter("java"));
int OR=Integer.parseInt(request.getParameter("or"));
int DBMS=Integer.parseInt(request.getParameter("dbms"));
int IS=Integer.parseInt(request.getParameter("is"));
int STL=Integer.parseInt(request.getParameter("stl"));
int c=java+OR+DBMS+IS+STL;
double avg=c/5;
if(avg > 90 ){
out.println(" your grade is A");
}else if (avg >= 80) {
out.println("your grade is B");
} else if (avg >= 70) {
out.println("your grade is C");
} else if (avg >= 60) {
out.println("your grade is D");
} else {
out.println("your grade is E");
}
%>
```

index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<form action="marks.jsp" method="get">
Enter Java Marks : <input type="text" name="java"> <br><br>
Enter OR Marks : <input type="text" name="or"><br><br>
Enter DBMS Marks :<input type="text" name="dbms"><br><br>
Enter IS Marks :<input type="text" name="is"><br><br>
Enter STL Marks :<input type="text" name="stl"><br><br>
<input type="submit">
</form>
</body>
</html>
```

Output:

index.html marks.jsp http://localhost:8080/Practical_12/index.html

http://localhost:8080/Practical_12/index.html

Enter Java Marks : 94

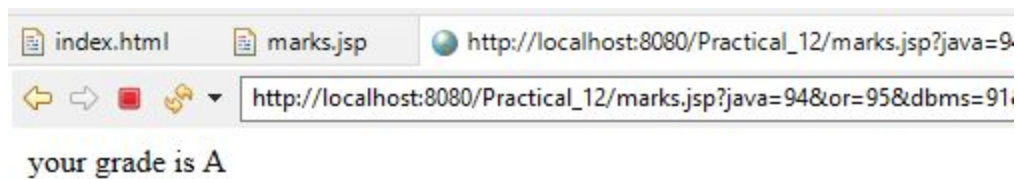
Enter OR Marks : 95

Enter DBMS Marks : 91

Enter IS Marks : 90

Enter STL Marks : 93

Submit Query



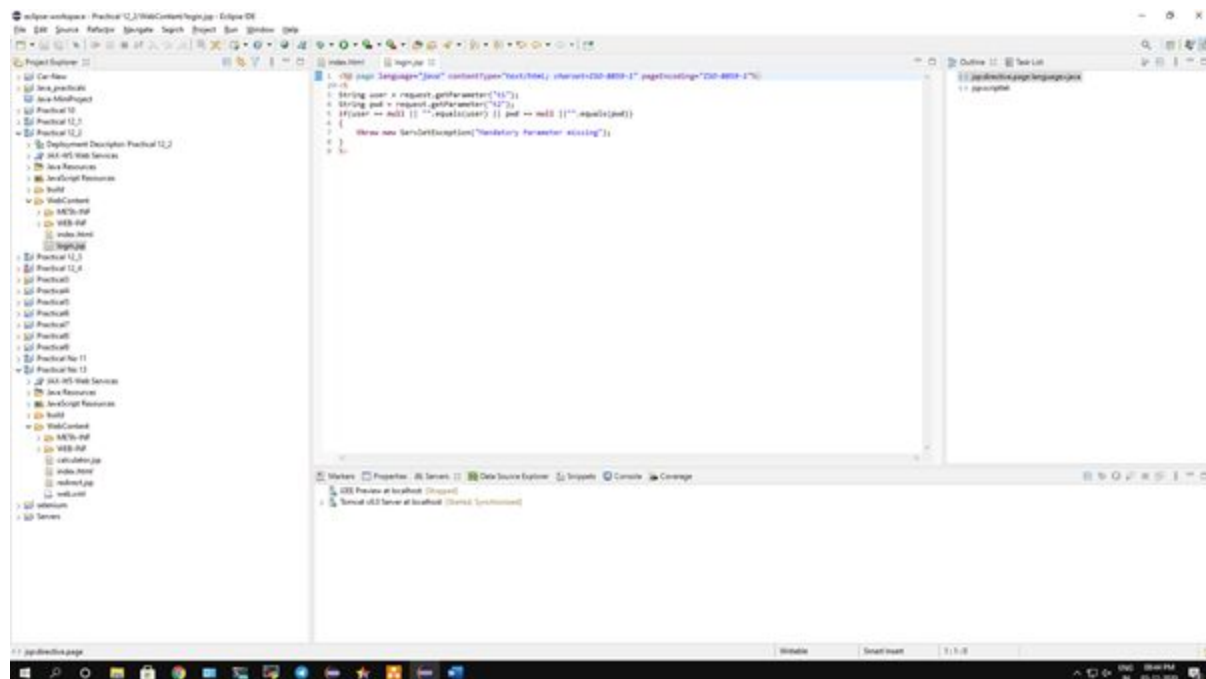
index.html marks.jsp http://localhost:8080/Practical_12/marks.jsp?java=94&or=95&dbms=91&is=90&stl=93

http://localhost:8080/Practical_12/marks.jsp?java=94&or=95&dbms=91&is=90&stl=93

your grade is A

Aim: Write a program to implement error and error Objects.

In the given program we have defined the jsp directive function and the we defined string user and pwd to get a request. If the user enters the password it will login and if the user fails then it will throw an error as the throw keyword is defined which displays as a mandatory parameter missing. In index.html we have defined the html function and then we use form action as login.jsp and then we use input function to get username and password field which will input username and password and then the output will be displayed.



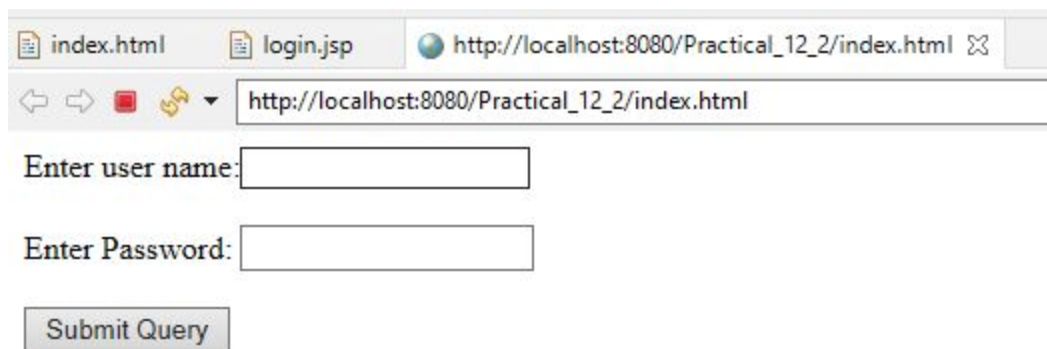
Conclusion: We have implemented error and error Objects.

Code:**login.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%
String user = request.getParameter("t1");
String pwd = request.getParameter("t2");
if(user == null || "".equals(user) || pwd == null || "".equals(pwd))
{
    throw new ServletException("Mandatory Parameter missing");
}
%>
```

index.html

```
<!DOCTYPE html>
<html>
<body>
<form action="login.jsp" method="post">
Enter user name:<input type="text" name="t1"><br><br>
Enter Password:<input type="password" name="t2"><br><br>
<input type="submit" name="s1"><br><br>
</form>
</body>
</html>
```

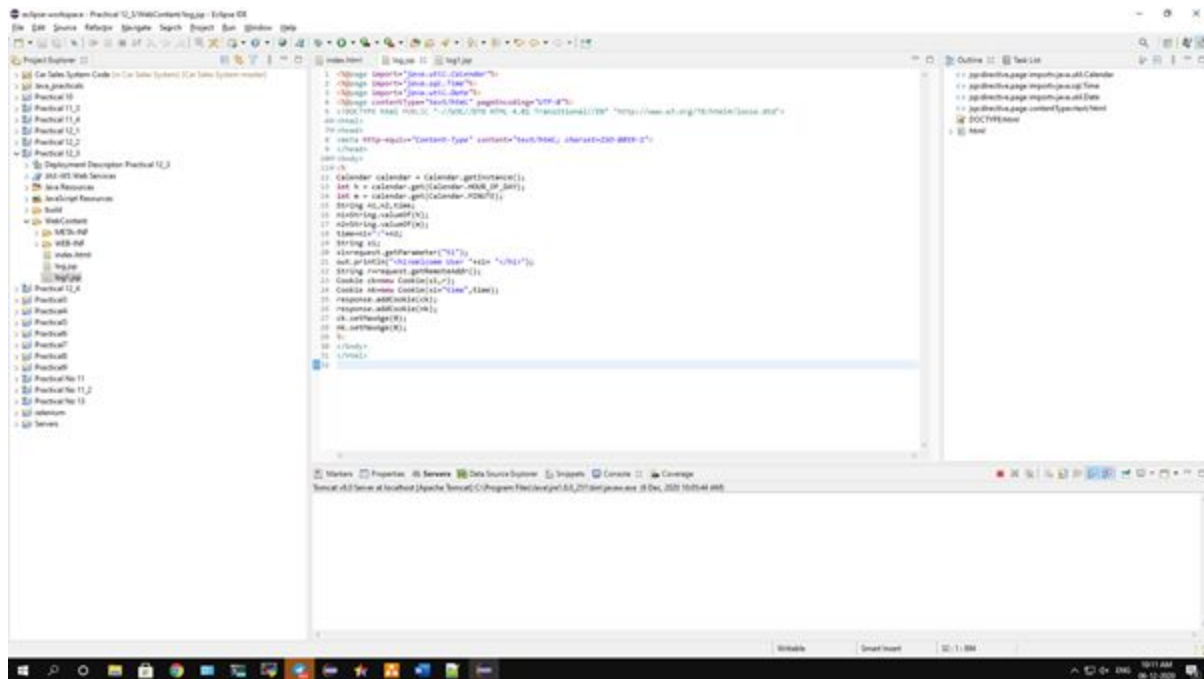
Output:

The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Practical_12_2/index.html`. The browser tabs include `index.html` and `login.jsp`. The page content consists of a form with two input fields: "Enter user name:" and "Enter Password:". Below these fields is a button labeled "Submit Query".



Aim: Write a program to implement a program to create a Visitor Log that reports the IP Address of each User, and the time they visited the page.

In this program we used the html functions with forms such as enter username and submit button. Then we imported all the java utils for logging the users. Then we defined the calendar function with integer hours and mins to get hours and minutes of the day. Then we defined a string to get the username and display the user message. Then the output will display once the user logged in and the ip address will also shown on the output.



Conclusion: We have implemented a program to create a Visitor Log that reports the IP Address of each User, and the time they visited the page.

Code:**index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
<form method="post" action="log.jsp">
Enter user name:<input type="text" name="t1"><br><br>
<input type="submit" name="s1"><br><br>
</form>
</body>
</html>
```

log.jsp

```
<%@page import="java.util.Calendar"%>
<%@page import="java.sql.Time"%>
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<%
Calendar calendar = Calendar.getInstance();
int h = calendar.get(Calendar.HOUR_OF_DAY);
int m = calendar.get(Calendar.MINUTE);
String n1,n2,time;
n1=String.valueOf(h);
n2=String.valueOf(m);
time=n1+":"+n2;
String s1;
s1=request.getParameter("t1");
out.println("<h1>Welcome User "+s1+ "</h1>");
String r=request.getRemoteAddr();
Cookie ck=new Cookie(s1,r);
Cookie nk=new Cookie(s1+"time",time);
response.addCookie(ck);
response.addCookie(nk);
ck.setMaxAge(0);
nk.setMaxAge(0);
%>
</body>
</html>
```

log1.jsp

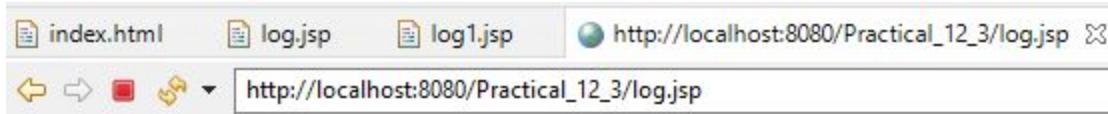
```
<% Cookie nk[]=request.getCookies();
Cookie ck[]=request.getCookies();
String[] name = new String[10];
String[] ip = new String[10];
String[] t = new String[10];
int c=1; %>
<table border="1">
<tr><th>User</th>
<th>IPAddress</th>
<th>Time</th> </tr>
<% for(int i=1;i<ck.length;i++){
if(!nk[i].getName().contains("time")){
name[c]= ck[i].getName();
ip[c] = ck[i].getValue();
c++; } }
c=1;
for(int i=1;i<nk.length;i++){
if(nk[i].getName().contains("time")){
t[c] = nk[i].getValue();
c++; } }
for(int i=1;i<c;i++){
out.println("<tr>");
out.println("<td>" + name[i] + "</td><td>" + ip[i] + "</td><td>" + t[i]
+ "</td>");
out.println("</tr>");} %></table>
```

Output:

index.html log.jsp log1.jsp http://localhost:8080/Practical_12_3/index.html

http://localhost:8080/Practical_12_3/index.html

Enter user name:



Welcome User ninad

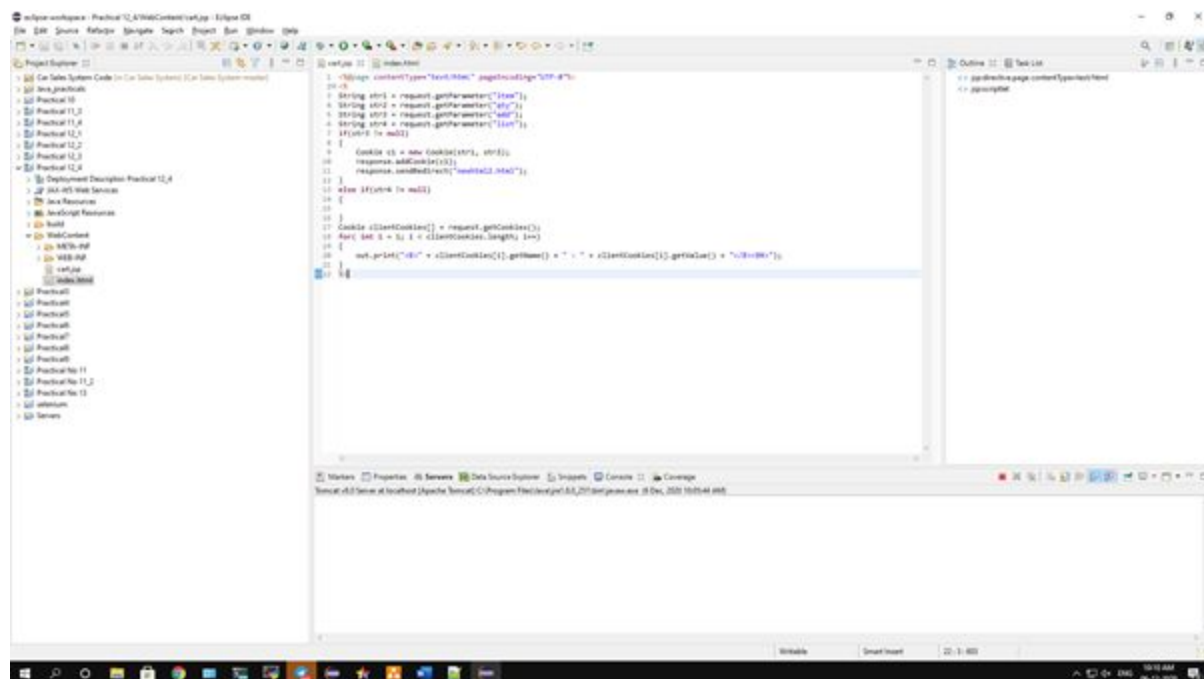


D) To implement a program for Session management in JSP for Shopping cart Application.

Aim: Write a program to implement a program for Session management in JSP for Shopping cart Application.

Description:

A Cookie is a small piece of data that is exchanged between a server and a client. Whenever a client sends a request, the server will send a cookie containing the required data and the client can send back the cookie with its next request. In session management, Tomcat creates a session id whenever the client's first request gets to the server (However, other servlet containers may behave differently). Then it inserts this session id into a cookie with a name JSESSIONID and sends along with the response. After receiving the response with the cookie, the client can send the received cookie in its next request so that the server will identify the session using session id that resides in the JSESSIONID cookie. In this program we have defined the html input functions for entering items and quality. Then we defined the form method and then we added 2 submit buttons for the list and added cookies. Then we use the jsp file for output in which we defined the string functions and then cookies. The output has been displayed based on user inputs.



Conclusion: We have implemented a program for Session management in JSP for Shopping cart Application.

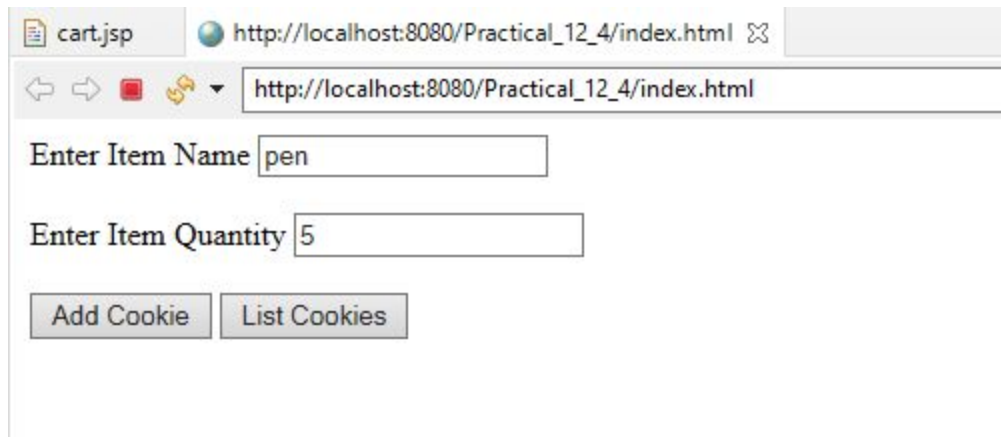
Code:

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
</head>
<body>
<form method="post" action="cart.jsp">
Enter Item Name <input type="text" name="item"><br><br>
Enter Item Quantity <input type="text" name="qty"><br><br>
<input type="submit" value="Add Cookie" name="add">
<input type="submit" value="List Cookies" name="list">
</form>
</body>
</html>
```

cart.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
String str1 = request.getParameter("item");
String str2 = request.getParameter("qty");
String str3 = request.getParameter("add");
String str4 = request.getParameter("list");
if(str3 != null)
{
    Cookie c1 = new Cookie(str1, str2);
    response.addCookie(c1);
    response.sendRedirect("newhtml2.html");
}
else if(str4 != null)
{
}
Cookie clientCookies[] = request.getCookies();
for( int i = 1; i < clientCookies.length; i++)
{
    out.print("<B>" + clientCookies[i].getName() + " : " + clientCookies[i].getValue() +
"</B><BR>");
}
%>
```

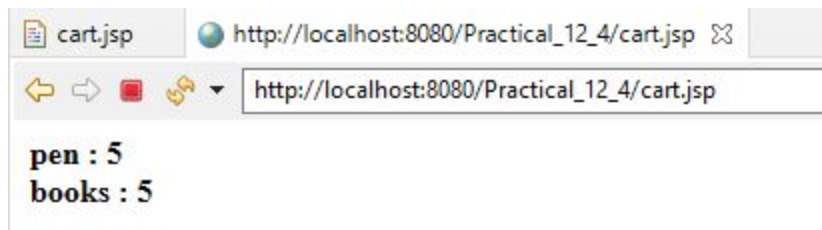
Output:

cart.jsp http://localhost:8080/Practical_12_4/index.html

http://localhost:8080/Practical_12_4/index.html

Enter Item Name

Enter Item Quantity



cart.jsp http://localhost:8080/Practical_12_4/cart.jsp

http://localhost:8080/Practical_12_4/cart.jsp

pen : 5
books : 5

Name Of Student: NINAD AVINASH PATIL				
Roll No: 33		Lab Practical Number: 13		
Title Of Lab Practical: INTRODUCTION TO SPRING FRAMEWORKS				
DOP:		DOS:		
CO Mapped: CO1	PO Mapped: PO3,PO5,PO7	PSO Mapped: PSO1,PSO2	Faculty Signature:	Marks:

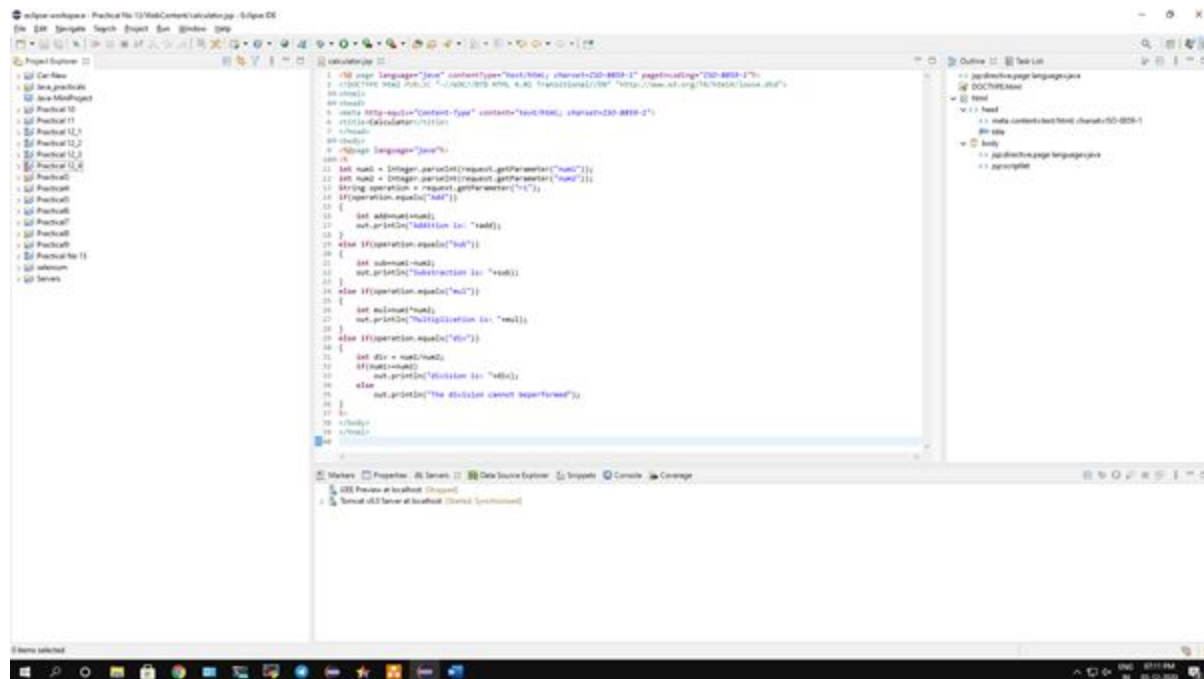
Practical No 13

To implement a basic program on Spring Framework.

Aim: Write a program to implement a basic program on spring framework.

Description:

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code. Spring framework is an open source Java platform. It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003. Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB. So, in this program we first created a file name called calculator.jsp and then we used the html body and then defined the page language as java, then we defined the int functions and string operators for requesting parameters as arithmetic operations. Then we define the if else if functions for operators and then we created another file called index.html in which all the html related functions are stored such as form actions, radio buttons and input type. Then we created a redirect.jsp file which will redirect the index.html file for showing the response once we clicked the submit button. Then in web.xml it consists of all the manifest of the current project.



Conclusion: We have implemented a basic program on spring framework.

Code:

Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome to Spring Web MVC project</title>
</head>
<body>
<form action="calculator.jsp" method="get">
<label for="num1"><b>VALUE 1</b></label>
<input type="text" name="num1"><br><br>
<label for="num2"><b>VALUE 2</b></label>
<input type="text" name="num2"><br><br>
<input type="radio" name="r1" value="Add">ADD
<input type="radio" name="r1" value="Sub">SUB
<input type="radio" name="r1" value="mul">MUL
<input type="radio" name="r1" value="div">DIV
<br><br><input type="submit" value="ANSWER">
</form>
</body>
</html>
```

calculator.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Calculator</title>
</head>
<body>
<%@page language="java"%>
<%
int num1 = Integer.parseInt(request.getParameter("num1"));
int num2 = Integer.parseInt(request.getParameter("num2"));
String operation = request.getParameter("r1");
if(operation.equals("Add"))
{
    int add=num1+num2;
```

```

        out.println("Addition is: "+add);
    }
    else if(operation.equals("Sub"))
    {
        int sub=num1-num2;
        out.println("Substraction is: "+sub);
    }
    else if(operation.equals("mul"))
    {
        int mul=num1*num2;
        out.println("Multiplication is: "+mul);
    }
    else if(operation.equals("div"))
    {
        int div = num1/num2;
        if(num1>=num2)
            out.println("division is: "+div);
        else
            out.println("The division cannot beperformed");
    }
}
%>
</body>
</html>

```

redirect.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
response.sendRedirect("index.html");
%>

```

web.xml

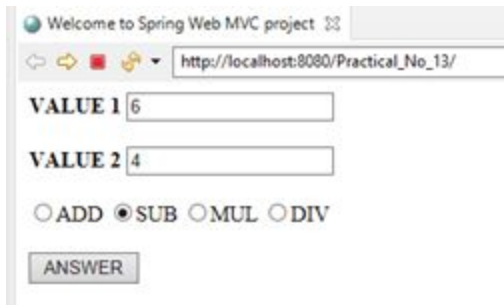
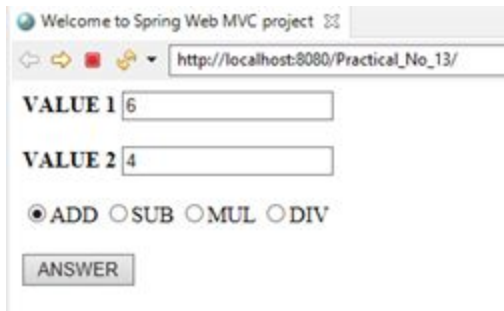
```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<servlet>
<servlet-name>dispatcher</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<load-on-startup>2</load-on-startup>

```

```
</servlet>
<servlet-mapping>
<servlet-name>dispatcher</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>
<session-config>
<session-timeout>30</session-timeout>
</session-config>
<welcome-file-list>
<welcome-file>redirect.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Output:



Welcome to Spring Web MVC project

http://localhost:8080/Practical_No_13/

VALUE 1

VALUE 2

☐ ADD ☐ SUB ☒ MUL ☐ DIV

Calculator

http://localhost:8080/Practical_

Multiplication is: 24

Welcome to Spring Web MVC project

http://localhost:8080/Practical_No_13/

VALUE 1

VALUE 2

☐ ADD ☐ SUB ☐ MUL ☒ DIV

Calculator

http://localhost:8080,

division is: 2