Topic : ..... Date. : ..... Page No. : .....

Experiment 1

Aim

1. WAP to calculate factorial of a no.
2. WAP to check whether a no. is Armstrong or not
3. WAP to print Fibonacci series upto N
4. WAP to take 4 nos. from command line & print the largest, smallest & average.

Software Used

VS Code

Theory

Java is a programming language & a platform. It is a high level, robust, object oriented & secure programming language

Features of java:

1. Simple
2. Object oriented
3. Portable
4. Platform independent
5. Secure
6. Robust
7. Interpreted
8. Multi threaded

Basic Keywords

1. class - Used to declare a class in Java
2. public - Access modifier which represents visibility. It means it is visible to all.

Topic : ..... Date. : ..... Page No. : .....

3. Static - In this method there is no need to create an object to invoke the static method

4. void - Return type of the method. It means it doesn't return any value

5. main - Starting point of program

6. String [] args - Used for command line arguments

7. System.out.println() - Used to print statements.
System is a class.
out is an object of PrintStream class.
println() is method of PrintStream.

```
class Sample {
    public static void main (String args[])
    {
        System.out.println ("Hello World");
    }
}
```

Output: Hello World

Java User Input (Scanner)

The scanner class is used to get user input & it is found in java.util package.

To use the scanner class, create an object of class & use the methods available.

Method

Collection of instructions that performs a specific type of function

Topic : ..... Date. : ..... Page No. : .....

A method is a block of code or collection of statements to perform a certain task. It is used for reusability of code

Method Declaration:

```
public int FactorialNum (int n)
{
}
```

Access specifier ↓ Return type ↓ Method name

There are 2 types of methods
- Predefined
- User-defined

Predefined methods have already been defined in the Java class, libraries. They are also called standard library method. Ex - length(), equals()

The method written by the user or programmer is a user-defined method. These methods are modified according to the requirement.

Recursion

Technique of making a function call itself. Provides a way to break complicated problems down to simpler problems.

It has the following components:

① Base Case

A recursive function must have a terminating condition when the function stops calling itself.

② Recursive Call

The function will recursively invoke itself

Topic : ..... Date. : ..... Page No. : .....

③ Small Calculation

Generally we perform some calculations steps in each recursive call. We can perform this calculation step before or after the recursive call depending upon the nature of the problem.

Source Codes

(I) Factorial of a given number

```
import java.util.Scanner;
public class factorial {
	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		int i = sc.nextInt();
		int j = i;
		int prod = 1;
		while(j>0)
		{
			prod = prod*j;
			j--;
		}
		System.out.println("Factorial:" + prod);
		sc.close();
	}
}
```

(II) Check whether a number is Armstrong or not

```
import java.util.Scanner;
public class armstrong {
	public static void main(String args[])
	{
		Scanner sc = new Scanner(System.in);
		int i = sc.nextInt();
		int j = i;
		int sum=0;
		while(j!=0)
		{
			int k = j%10;
			sum+=(k*k*k);
			j/=10;
		}
		if(sum==i)
			System.out.println("Armstrong Number");
		else
			System.out.println("Not Armstrong Number");
		sc.close();
	}
}
```

(III) Print Fibonacci series upto n

```
import java.util.Scanner;
public class fibo {
	public static void main(String args[])
	{
		Scanner sc = new Scanner(System.in);
		int a = sc.nextInt();
		int b=0,c=1,d,i=2;
		System.out.println(b);
		System.out.println(c);
		while(i<a)
		{
			d=b+c;
			b=c;
			c=d;
			System.out.println(d);
			i++;
		}
		sc.close();
	}
}
```

(IV) Find largest, smallest and average of four numbers

```
import java.util.Scanner;
public class largest {
	public static int large(int a, int b, int c, int d)
	{
		if(a>=b && a>=c && a>=d)
			return a;
		else
		if(b>=a && b>=c && b>=d)
			return b;
		else
		if(c>=a && c>=b && c>=d)
			return c;
		else
			return d;
	}
	public static int small(int a, int b, int c, int d)
	{
		if(a<=b && a<=c && a<=d)
			return a;
		else
		if(b<=a && b<=c && b<=d)
			return b;
		else
		if(c<=a && c<=b && c<=d)
			return c;
		else
			return d;
	}
	public static float avg(int a, int b, int c, int d)
	{
		float sum = a+b+c+d;
		return (sum/4);
	}
	public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();
        int d = sc.nextInt();
        int ans1 = large(a, b, c, d);
        int ans2 = small(a, b, c, d);
        float ans3 = avg(a, b, c, d);
        System.out.println("Largest:" + ans1);
        System.out.println("Smallest:" + ans2);
        System.out.println("Average:" + ans3);
        sc.close();
    }
}
```

Outputs

```
5
Factorial:120
abhinav@jarvis:~/clg_3/jp$
```

```
153
Armstrong Number
abhinav@jarvis:~/clg_3/jp$
```

```
8
0
1
1
2
3
5
8
13
```

```
10 32 76 1
Largest:76
Smallest:1
Average:29.75
```

Topic : ..... Date. : ..... Page No. : .....

Experiment 2

Aim

1. Create class Shape & declare variables. Extend shape class into 2 classes i.e. circle & square. Write methods CalcArea() & CalcPeri(). Test functionality.
2. Create class Person & extend it to Student & Teacher class

Software Used

Visual Studio Code

Theory

OOPs

Object means a real world entity such as a pen, chair, table etc.

Object oriented programming is a methodology or paradigm to design a program using classes & objects.

Properties of OOPs

→ Inheritance - When one object acquires all the properties & behaviours of a parent object. It provides code reusability

→ Polymorphism - If one task is performed in multiple ways. In Java, method overloading & method overriding is used.

→ Abstraction - Hiding internal details & showing functionality is known as abstraction. In Java, abstract class & interface are used.

Topic : ..... Date. : ..... Page No. : .....

⇒ Encapsulation - Binding (or wrapping) code & data together into a single unit
A Java class is an example of encapsulation

Inheritance in Java
It is a mechanism in which one object acquires all properties & behaviours of a parent object.

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods & fields of the parent class.

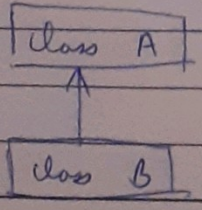
Syntax:

```
class Subclass-name extends Superclass-name
{
    //methods
}
```

Types of inheritance

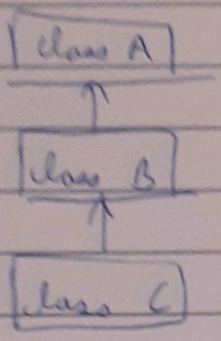
On the basis of class, there can be 3 types of inheritance in Java
- Single
- Multilevel
- Hierarchical

1. Single

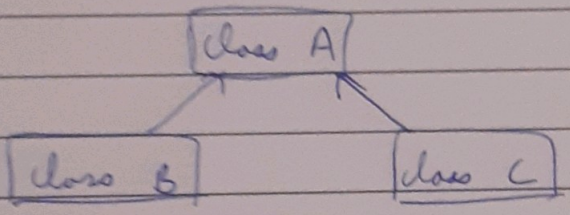


Topic : ..... Date. : ..... Page No. : .....

2. Multilevel



3. Hierarchical

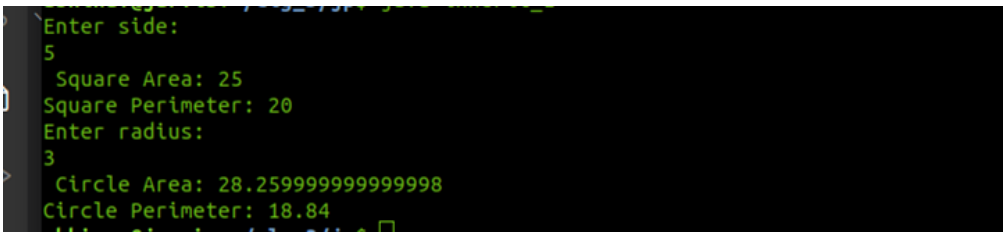


Source Code

```
import java.util.Scanner;
class Shape
{
	int radius, side;
}
class Square extends Shape
{
	Scanner s = new Scanner(System.in);
	public void getdata()
	{
		System.out.println("Enter side: ");
		side = s.nextInt();
		getArea();
		getPeri();
	}
	public void getArea()
	{
		System.out.println(" Square Area: " + side*side);
	}
	public void getPeri()
	{
		System.out.println("Square Perimeter: " + 4*side);
	}
}

class Circle extends Shape
{
	Scanner s = new Scanner(System.in);
	public void getdata()
	{
		System.out.println("Enter radius: ");
		radius = s.nextInt();
```

```
                getArea();
                getPeri();
        }
        public void getArea()
        {
                System.out.println(" Circle Area: " + 3.14*radius*radius);
        }
        public void getPeri()
        {
        System.out.println("Circle Perimeter: " + 2*3.14*radius);
        }
}
public class inherit_1{
        public static void main(String[] args) {
                Square sq = new Square();
                Circle ci = new Circle();
                sq.getdata();
                ci.getdata();
        }
}
```

Output



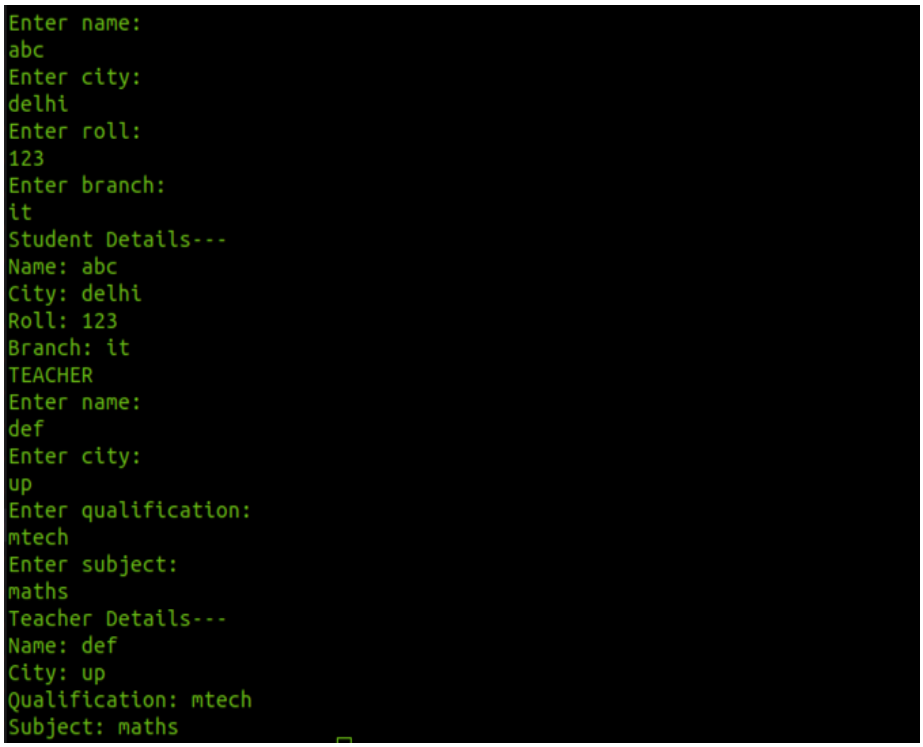
Source Code

```
import java.util.Scanner;

class People
{
	String name, city;
}


class Student extends People
{
	Scanner sc = new Scanner(System.in);
	int roll;
	String branch;
	public void getData()
	{
		System.out.println("Enter name: ");
		name = sc.nextLine();
		System.out.println("Enter city: ");
		city = sc.nextLine();
		System.out.println("Enter roll: ");
		roll = sc.nextInt();
		String temp = sc.nextLine();
		System.out.println("Enter branch: ");
		branch = sc.nextLine();
		System.out.println("Student Details---");
		display();
	}
	public void display()
	{
		System.out.println("Name: " + name);
		System.out.println("City: " + city);
		System.out.println("Roll: " + roll);
		System.out.println("Branch: " + branch);
	}
```

```
}

class Teacher extends People
{
	Scanner sc = new Scanner(System.in);
	String qual, subj;
	public void getData()
	{
		System.out.println("Enter name: ");
		name = sc.nextLine();
		System.out.println("Enter city: ");
		city = sc.nextLine();
		System.out.println("Enter qualification: ");
		qual = sc.nextLine();
		System.out.println("Enter subject: ");
		subj = sc.nextLine();
		System.out.println("Teacher Details---");
		display();
	}
	public void display()
	{
		System.out.println("Name: " + name);
		System.out.println("City: " + city);
		System.out.println("Qualification: " + qual);
		System.out.println("Subject: " + subj);
	}

}

public class inherit_2 {
	public static void main(String[] args) {
		Student st = new Student();
		Teacher te = new Teacher();
		System.out.println("STUDENT");
```

```
            st.getData();
            System.out.println("TEACHER");
            te.getData();
        }
}
```

Output



Date :
Page No. :

Experiment 3

Aim

1. Write a class MathOperation which accepts 5 integers through command line. Create an array, loop through array & obtain sum & average. Various exceptions should be handled.

2. WAP with a division method which receives two integers nos. & perform division operation. Arithmetic Exception should be declared in method, & should be handled in main method.

3. Create a user defined exception class, overload the respective constructors. Create a main class UserRegistration & add the given methods

Software Used

Visual Studio Code

Theory

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

Exceptions In Java, an exception is an event that disrupts the normal flow of the program. It is an object

Date :
Page No. :

thrown at runtime

Types of Java Exceptions
① Checked Exception
② Unchecked exception
③ Errors

There are 5 keyword used:
→ try → catch → finally → throw → throws

Try Catch Block
Try block is used to enclose the code that might throw an exception. It must be used within the method.
If an exception occurs at the particular statement of try block, the rest of the block code will not execute.
Java try block must be followed by either catch or finally block.

Syntax: try {
// code that may throw an exception
} catch (Exception_class_name ref) {}

Java catch block is used to handle the Exception by declaring the type of exception within the parameter.

Date :
Page No. :

Java finally block:
Java 'finally' block is a block that is used to execute important code such as closing connection, stream etc.
It is always executed whether exception is handled or not.
Finally block follows try or catch block

Java throw block:
Used to explicitly throw an exception.
We can throw either checked or unchecked exception in java by throw keyword.
→ throw exception;

There are some scenarios where unchecked exceptions may occur:

→ Arithmetic Exception
int a = 50/0; //Exception

→ NullPointer Exception
String s = null
System.out.println(s.length()); //Exception

→ Number Format Exception
String s = "abc"
int i = Integer.parseInt(s); //Exception

Date :
Page No. :

→ Array Index Out Of Bounds Exception

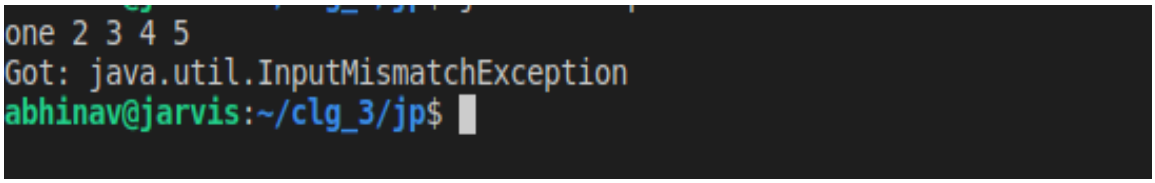
```
int a[] = new int [5];
a[10] = 50;                    // Exception
```

(I)

Source Code

```
import java.util.InputMismatchException;

import java.util.Scanner;

public class MathOperation

{

	public static void main(String[] args)

	{

		Scanner s = new Scanner(System.in);

		int[] arr = new int[5];

		int sum, avg;

		try {

			for(int i=0;i<5;i++)

			{

				arr[i] = s.nextInt();

			}

			sum = arr[0]+arr[1]+arr[2]+arr[3]+arr[4];

			avg = sum/5;

			System.out.println("Sum: " + sum);

			System.out.println("Avg: " + avg);

		}

		catch (ArithmeticException ae)

		{

			System.out.println("Got: " + ae);

		}

		catch (InputMismatchException ne){

			System.out.println("Got: " + ne);

		}

	s.close();

	}

}
```

Output

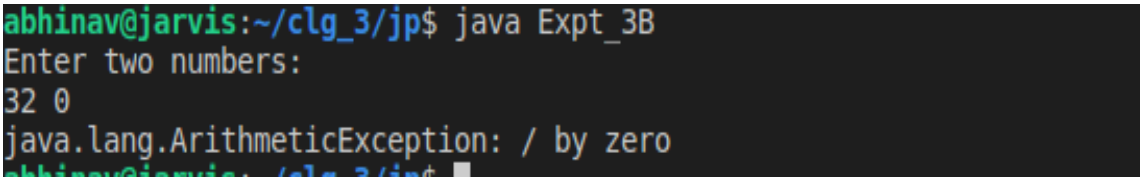

one 2 3 4 5
Got: java.util.InputMismatchException
abhinav@jarvis:~/clg_3/jp$

(II)

Source Code

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class Expt_3B {
	public static double division(int a, int b) throws ArithmeticException
	{
		return a / b;
	}
	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		System.out.println("Enter two numbers: ");
		int x = s.nextInt();
		int y = s.nextInt();
		try{
			double r = division(x,y);
			System.out.println(r);
		}
		catch(InputMismatchException e)
		{
			System.out.println("java.util.InputMismatchException");
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	s.close();
	}
}
```

Output


abhinav@jarvis:~/clg_3/jp$ java Expt_3B
Enter two numbers:
32 0
java.lang.ArithmeticException: / by zero

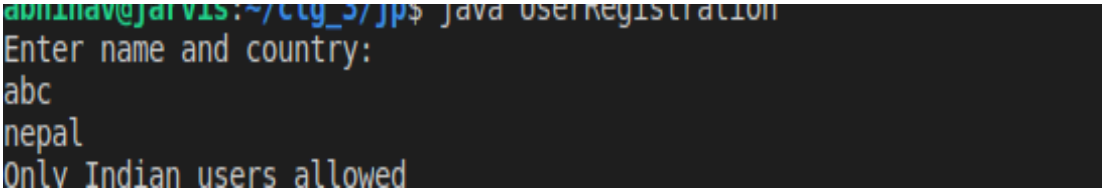
(III)

Source Code

```
import java.util.Scanner;
public class UserRegistration {
	void registerUser(String username,String userCountry) throws InvalidCountryException
	{
		if(userCountry!= "India")
			throw new InvalidCountryException();
		else
			System.out.println("User Registered");
	}
	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		UserRegistration user = new UserRegistration();
		System.out.println("Enter name and country: ");
		String u_name = s.nextLine();
		String country = s.nextLine();
		try
		{
			user.registerUser(u_name, country);
		}
		catch(InvalidCountryException e)
		{}
	s.close();
	}
}
```

```
public class InvalidCountryException extends Exception
{
	public InvalidCountryException() {
		super();
		System.out.println("Only Indian users allowed");
	}
}
```

Output



Date :
Page No. :

EXPERIMENT 4

Aim

1) Create a thread which prints 1 to 10. After 5, there should be a delay of 5000 ms before printing 6
2) WAP (1) using stop method
3) Create two threads, one thread to display all even nos. b/w 1 & 20, another to display odd nos. b/w 1 & 20

Software Used
Visual studio Code

Theory
Multithreading in java is a process of executing multiple threads simultaneously.

Threads share the same address space
A thread is lightweight.
cost of communication b/w the threads is low.
Exception in one thread doesn't affect the other.

Java provides Thread class to achieve thread programming.
Thread class extends Object class & implements Runnable interface.

Date :
Page No. :

Creating a Thread

There are 2 ways to create a Thread :

1. Extending Thread class
2. Implementing Runnable Interface

THREAD CLASS

Provides constructors & methods to create & perform operations on a thread

Constructors : Thread ( )
Thread (String name)
Thread (Runnable r)
Thread (Runnable r, String name)

Methods :

① public void run ( ) : Used to perform action for a thread

② public void start ( ) : Starts execution of thread. JVM calls the run method on the thread

③ public void sleep (long milliseconds) : Causes the currently executing thread to sleep for specified time

④ public void join ( ) : Waits for thread to die

Date :
Page No. :

⑤ public void stop() : Stops the Thread

RUNNABLE INTERFACE

It should be implemented by any class whose instances are intended to be executed by a thread.

1. Thread class

```
class Multi extends Thread {
    public void run(){}
}

p.s.v.m.(String args[]){
    Multi t1 =new Multi();
    t1.start();
}
}
```

2. Runnable Interface

```
class Multi ~~extends~~ implements Runnable {
    public void run(){}
}

p.s.v.m(String args[])
{   Multi m = new Multi();
    Thread t1 = new Thread(m);
    t1.start();
}
}
```

(I) Sleep

Source Code

```
public class ThreadA {
	public static void main(String args[])
	{
		t1 obj=new t1();
		obj.start();
	}
}
class t1 extends Thread{
	public void run()
	{
		int i;
		for(i=1;i<11;i++)
		{
			try
			{
				if(i==6)
				{
					System.out.println("Sleeping");
					sleep(5000);
				}
			}
			catch(Exception e) {}
			System.out.println(i);
		}
	}
}
```

Output

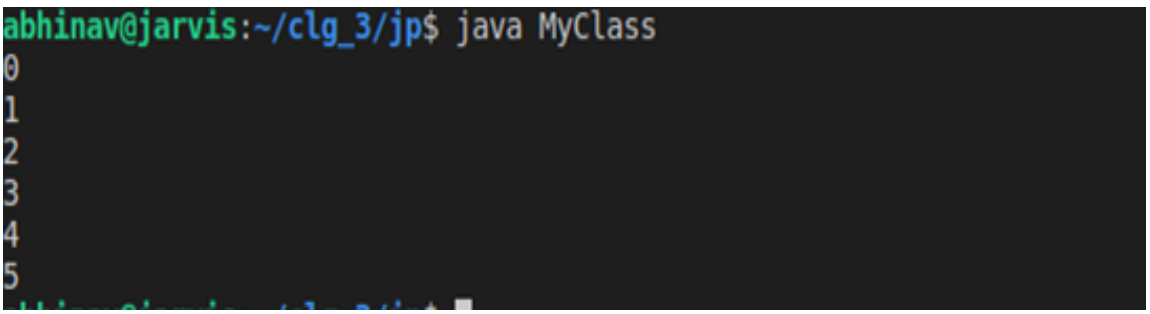
```
abhinav@jarvis:~/clg_3/jp$ java ThreadA
1
2
3
4
5
Sleeping
6
7
8
9
10
```

(II)Stop

Source Code

```
class Stop extends Thread {
	public void run()
	{
		for(int i = 0; i< 11; i++)
		{
			try
			{
				if(i == 6)
					stop();
				System.out.println(i);
			}
			catch(final Exception e)
			{
				System.out.println(e);
			}
		}
	}
}
public class MyClass {
	public static void main(String args[]) {
		final Stop ob2 = new Stop();
		ob2.start();
	}
}
```

Output

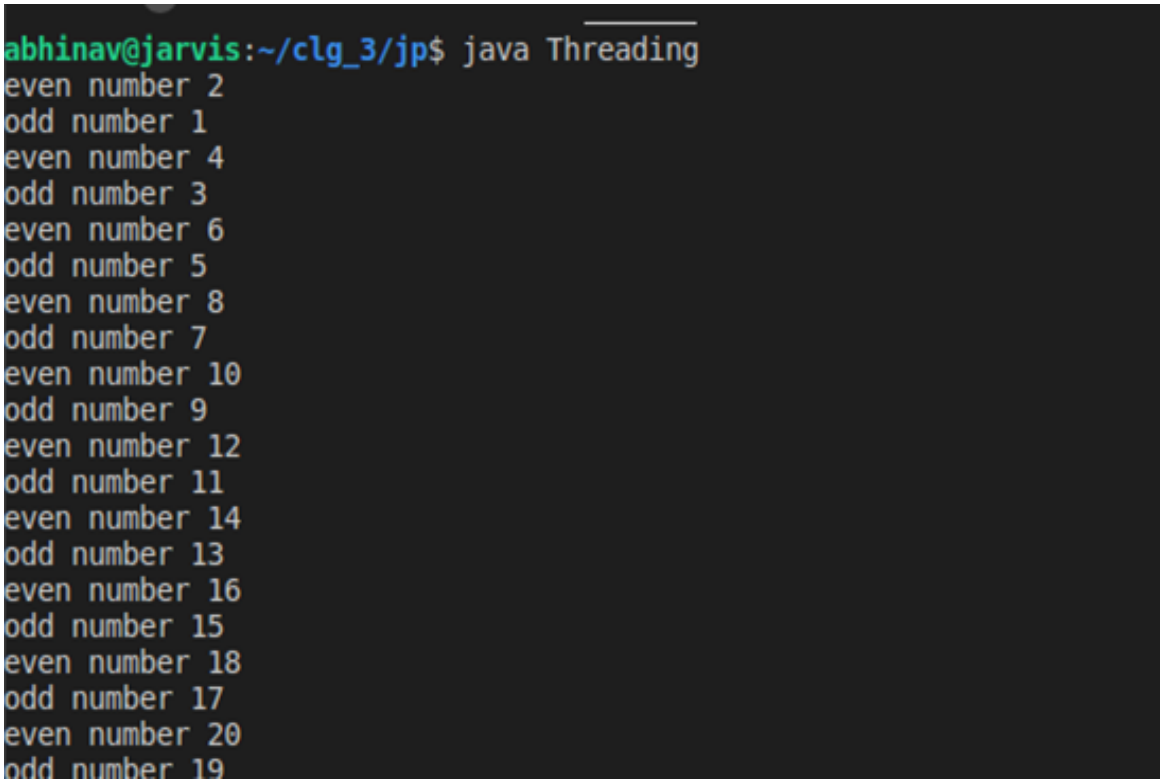


(III) Odd Even

Source Code

```
class even extends Thread{
	public void run()
	{
		for(int i=2;i<=20;i=i+2)
		{
			System.out.println("even number"+" "+i);
		}
	}
}
class odd extends Thread{
	public void run()
	{
		for(int i=1;i<=20;i=i+2)
			System.out.println("odd number"+" "+i);
	}
}
public class Threading
{
	public static void main(String[] args)
	{
		even e=new even();
		odd o=new odd();
		e.start();
		o.start();
	}
}
```

Output


abhinav@jarvis:~/clg_3/jp$ java Threading
even number 2
odd number 1
even number 4
odd number 3
even number 6
odd number 5
even number 8
odd number 7
even number 10
odd number 9
even number 12
odd number 11
even number 14
odd number 13
even number 16
odd number 15
even number 18
odd number 17
even number 20
odd number 19

Date :
Page No. :

EXPERIMENT 5

Aim

5.1 WAP to check whether a given string is palindrome or not

5.2 Given a string, return a new string made of n copies of the first 2 characters of original string

5.3 WAP that returns first half of a string, if length of string is even, it should return null for odd string

5.4 Given 2 strings, print a new string made up of first character of a, first of b, second of a, second of b & so on.

Software Used

VS Code

Theory

Strings in Java are objects that are backed internally by a character array. String object can be constructed a number of ways. Once a string has been created, its characters cannot be changed. Strings are immutable in nature.

They can be made in 2 ways:

1) By string literal

eg. String s = "Hello";

2) By new keyword

eg. String s = new String ("Hello");

Date :
Page No. :

In first way, JVM checks the string pool first. If string already exists in pool, a reference to the pool instance is returned. If not, then new string instance is created & placed in pool.
In string cost, JVM will create a new string object in normal heap memory

String, StringBuffer & StringBuilder classes are defined in java language. All are declared final, which means that none of these classes may be subclassed. All these implement CharSequence, Comparable & Serializable interfaces

The String class has 11 constructors that allow you to provide the initial value of string using different resources such as an array of characters

Some commonly used methods are:
1. char charAt (int index): Returns character at specified index
2. String concat (String str): Concatenate the specified string to end of string
3. int indexOf (char ch): Returns index within string of first ocurrence of specified character

Result
Given programs were implemented successfully & knowledge of string was gained

Codes:

1. Sequence is Palindrome or not

```
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: ");
      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("Palindrome.");
      }
      else
      {
         System.out.println("Not Palindrome.");
      }
   }
}
```

2. String made of first two character

```
import java.util.Scanner;
public class loop{
   public static void main(String[] args) {
      Scanner sc= new Scanner(System.in);
      String next="";
      System.out.println("Enter the string: ");
      String abc= sc.nextLine();
      int lenght = abc.length();
      if(lenght<=1){
         next=abc;

      }
      else{
         for (int i=0;i<lenght;i++){
            next=next+abc.charAt(0)+abc.charAt(1);
         }      }
      System.out.print(next);
```

```
        sc.close();

    }
}
```

3. Concatenation of two half of Strings

```
import java.util.Scanner;
public class half {
    public static void main(String[] args)
        Scanner sc= new Scanner(System.in);
        System.out.print("Enter the string: ");
        String abc= sc.nextLine();
        String bcd=null;
        int lenght=abc.length();
        int n=lenght/2;
        if(lenght%2==0)
        { bcd="";
            for(int i=0;i<n;i++){
                bcd=bcd+abc.charAt(i);

            }
            System.out.print(bcd);
        }
        else
            System.out.println(bcd);
        sc.close();
    }
}
```

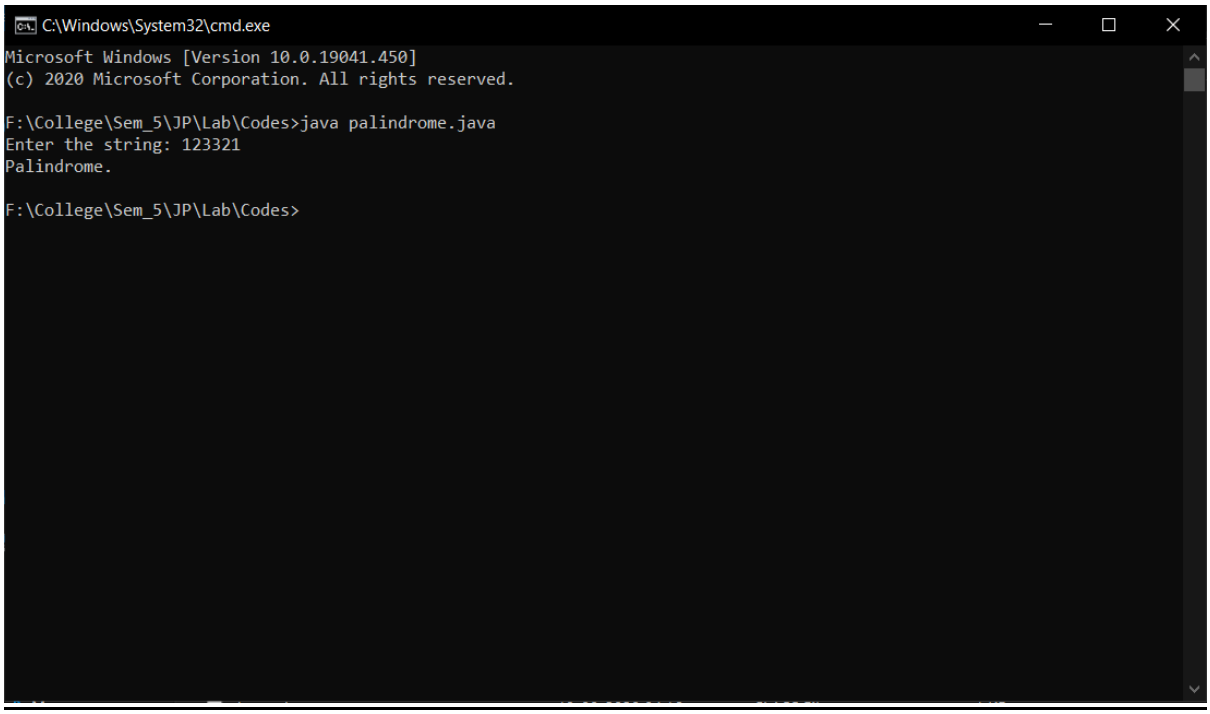
4. Char-Wise String Concatenation

```
import java.util.Scanner;
class Main {
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter a String: ");
    String str1=sc.nextLine();
    System.out.print("Enter another String: ");
    String str2=sc.nextLine();
    char[]temp1=str1.toCharArray();
    char[]temp2=str2.toCharArray();
    int n1=str1.length();
    int n2=str2.length();
    String ans="";
    int i=0,j=0;
    while(i<n1&&j<n2)
    {
      ans=ans+temp1[i]+temp2[j];
```

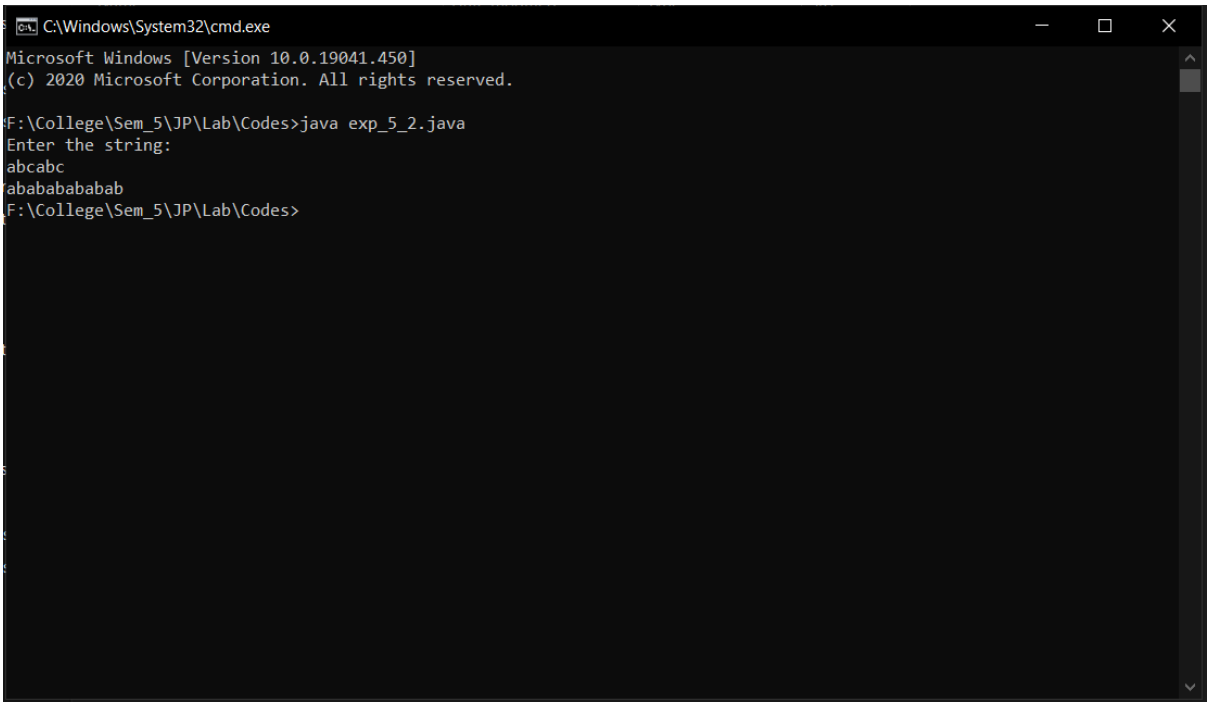
```
        i++;
        j++;
      }
      while(i<n1)
      {
        ans=ans+temp1[i];
        i++;
      }
      while(j<n2)
      {
        ans=ans+temp2[j];
        j++;
      }
      System.out.println(ans);
    }
  }
```

Output:

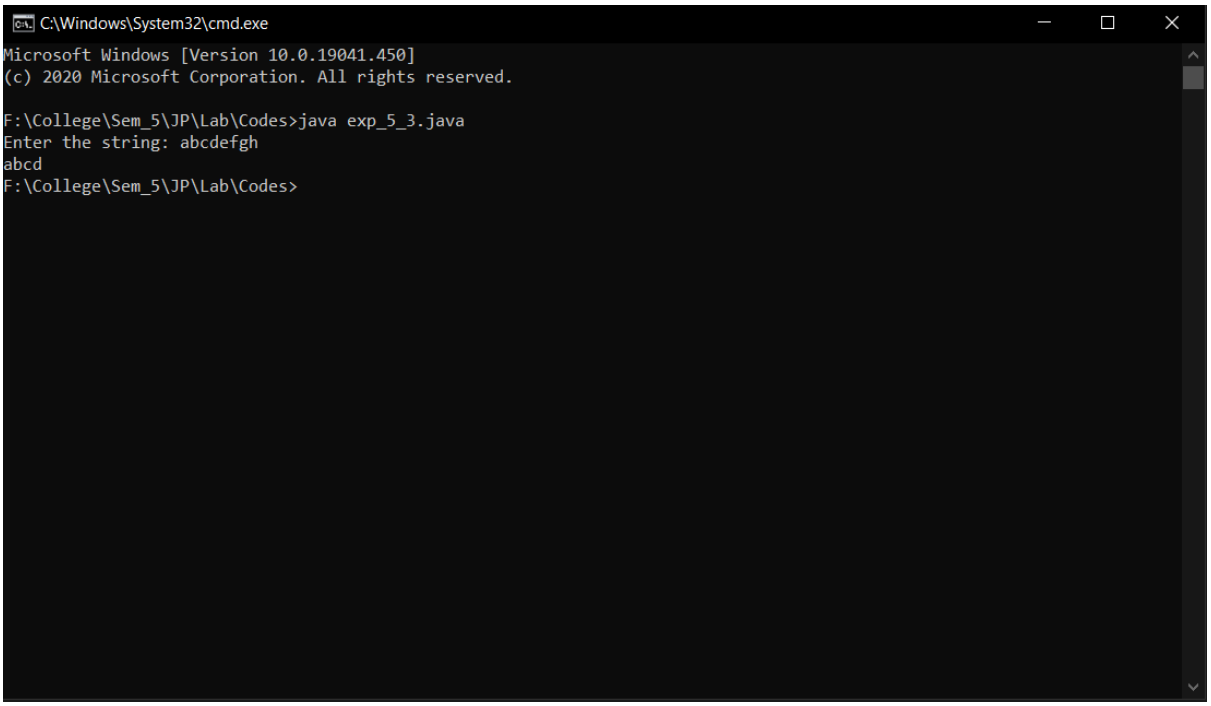
1. Sequence is Palindrome or not



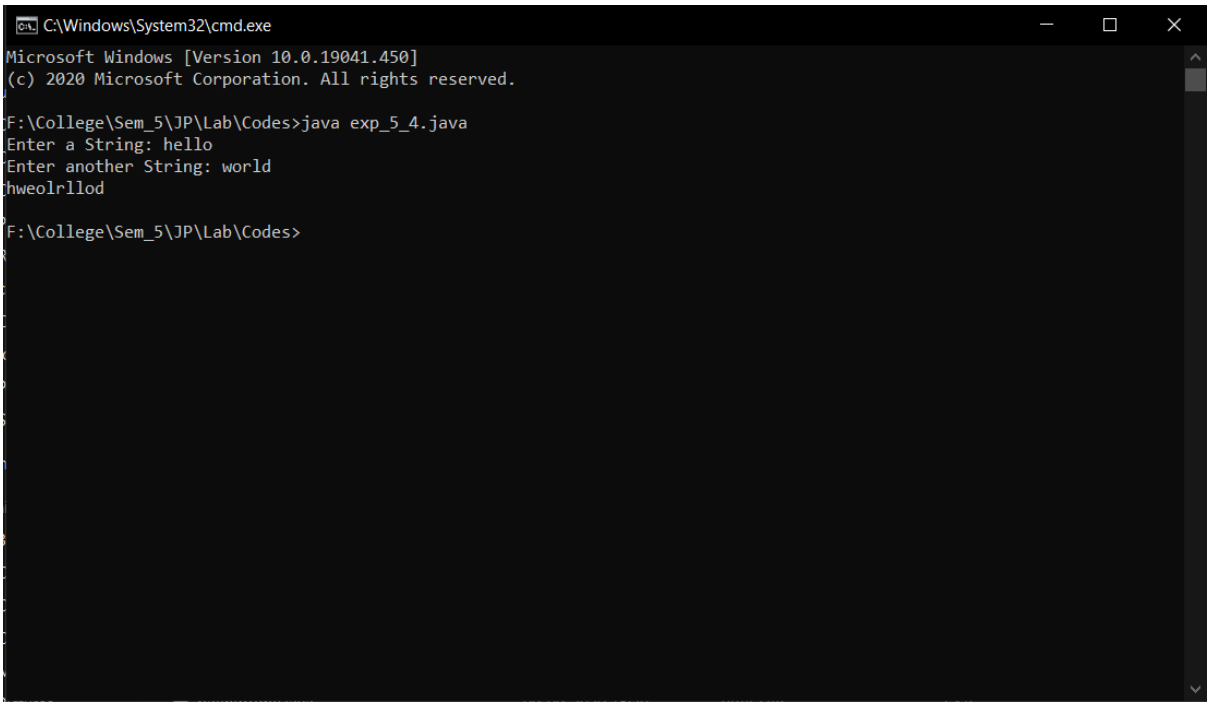
2. String made of first two character



3. Concatenation of two half of Strings



4. Char-Wise String Concatenation



Date :
Page No. :

EXPERIMENT 6

Aim

1. Draw a smiley face using applet programming.
2. Create an applet & pass the parameter name from HTML file. Fetch the parameter value in applet & check whether the string is palindrome or not. If palindrome, set value of textbox as "Name is palindrome" else set value as "Not palindrome"
3. Create an applet code with two text fields. In one field, user will enter a number. Check for prime digits & concatenate them. Set the concatenated value in second field.

Theory

→ Applet : It is a Java program that runs in Web browsers. An applet can be a fully functional Java application because it has entire Java API at its disposal

→ An applet is a Java class that extends the java.applet.Applet class
→ A main() method is not involved on an applet, & applet class will not define main().
→ Applets are designed to be embedded within a HTML page
→ When a user opens a HTML page that contains an

Date :
Page No. :

applet, the code for applet is downloaded to the user machine.

→ A JVM is required to view an applet.

→

Life cycle of an Applet.

→ init : Intended for initialization of any thing required by applet

→ start : Automatically called after the browser calls the init method.

→ stop : Automatically called when the users move off the page on which the applet sits.

→ destroy : Only called when browser shuts down normally

→ paint : Invoked immediately after the start method, & also any time the applet needs to repaint itself in the browser

The imp. syntax that brings the classes into the scope of our applet class

- java.applet.Applet
- java.awt.Graphics

The applet class

Every applet is an extension of the java.applet.Applet. The base Applet class provides methods that a derived class may call to obtain information & services.

Date :
Page No. :

→ Get applet parameters
→ Get network location of HTML file that contains the applet.
→ Get network location of applet class directory
→ Print a status message, fetch an image, fetch an audio clip, resize the applet.

Applet class provides an interface by which the viewer or browser obtains information about the applet & controls the execution.

Ques. 1

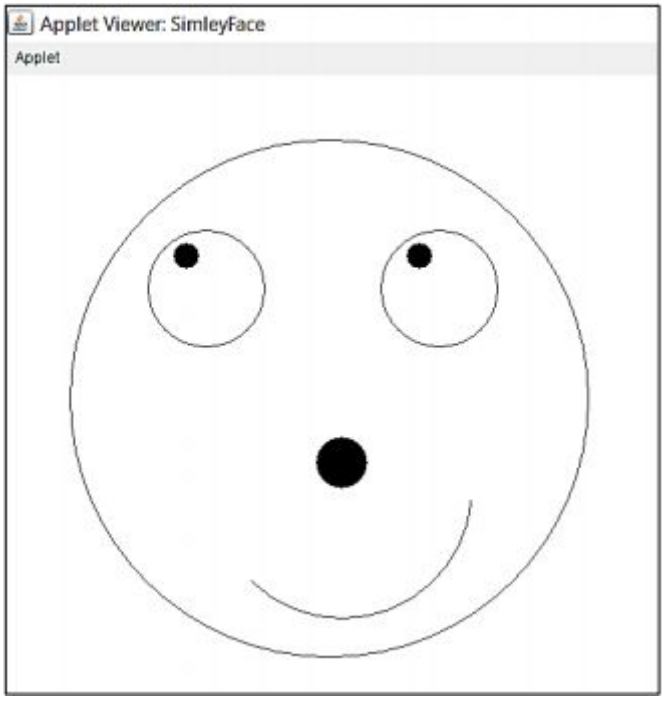
Java Code:

```
import java.applet.Applet;
import java.awt.Graphics;
	public class SimleyFace extends Applet {
		public void paint(Graphics g) {
			g.drawOval(50, 50, 400, 400);
			g.drawOval(110, 120, 90, 90);
			g.fillOval(130, 130, 20, 20);
			g.drawOval(290, 120, 90, 90);
			g.fillOval(310, 130, 20, 20);
			g.fillOval(240, 280, 40, 40);
			g.drawArc(160, 220, 200, 200, 225, 130);
		}
	}
```

HTML Code:

```
<html>
<head>
<body>
<applet code="SimleyFace.class" width=100 height=100></applet>
</body>
</head>
</html>
```

Output:



Ques. 2

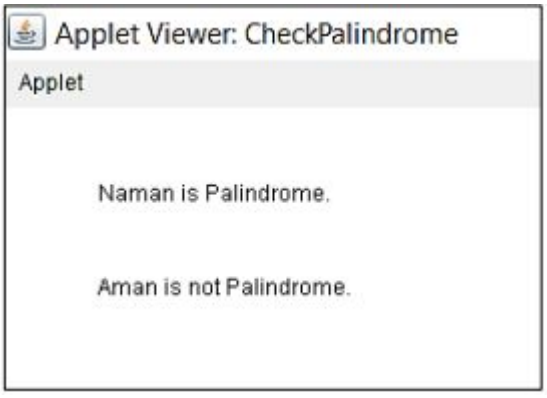
Java Code:

```
import java.applet.Applet;
import java.awt.Graphics;
	public class CheckPalindrome extends Applet {
		public int isPalindrome(String s) {
			String newstr = "", str = s.toLowerCase();
			for (int i = 0; i < str.length(); ++i) {
				newstr = str.charAt(i) + newstr;
			}
			if(str.equals(newstr)) return 1;
			else return 0;
		}
		public void paint(Graphics g) {
			String s1 = getParameter("P1"), s2 = getParameter("P2");
			if (isPalindrome(s1) == 1)
				g.drawString(s1 + " is Palindrome.", 50, 50);
			else
				g.drawString(s1 + " is not Palindrome.", 50, 100);
			if (isPalindrome(s2) == 1)
				g.drawString(s2 + " is Palindrome.", 50, 50);
			else
				g.drawString(s2 + " is not Palindrome.", 50, 100);
		}
	}
```

HTML Code:

```
<html>
<head>
<applet code = "CheckPalindrome.class" width = 100 height = 100>
<param name = "P1" value = "Naman">
<param name = "P2" value = "Aman">
</applet>
</head>
</html>
```

Output:



Ques. 3: Java Code:

```
import java.applet.Applet;
import java.awt.Graphics;
	public class testing extends Applet {
		TextField t1, t2;
		Label l1, l2;
		public void init() {
			t1 = new TextField(10);
			t2 = new TextField(10);
			l1 = new Label("Enter a number: ");
			l2 = new Label("Prime digits in the number are: ");
			add(l1);
			add(t1);
			add(l2);
			add(t2);
		}
	public void paint(Graphics g) {
		int num = Integer.parseInt(t1.getText());
		int prime = PrimeDigit(num);
		t2.setText(String.valueOf(prime));
	}
	public int PrimeDigit(int num) {
		int rem, flag;
		String str = "";
		while(num > 0) {
			rem = num % 10;
			flag = 1;
			for(int i = 2; i <= rem/i; ++i) {
				if(rem % i == 0) {
					flag = 0;
					break;
				}
			}
			if(flag == 1){
				str = String.valueOf(rem) + str;
				num /= 10;
			}
			return Integer.parseInt(str);
	}
}
```

HTML Code:

```
<html>
<head>
<applet code = "testing.class" width = 10 height = 10>
</applet>
</head>
</html>
```

Date :
Page No. :

EXPERIMENT 7

Aim

Programs based on File Handling

1) WAP to read a file & count the no. of words & characters in the file.
2) WAP to search for a specified word in the file
3) WAP to read a file & count no. of vowels in the file
4) WAP to read a file & count no. of lines
5) WAP to copy the contents of file into another file.

Software Used

VS Code

Theory

File Handling implies reading from & writing to a file. The file class from java.io package allows us to work with different format of files. In order to use the File class, you need to create an object of the class & specify the file name or directory name.

Java uses the concept of a stream to make i/o operations fast

A stream is a sequence of data. In Java, a stream is composed of bytes. It is called a stream because it flows. It is of two types:

Date :
Page No. :

① Byte stream :
This deals with byte data. When an i/p is provided & executed with byte data, it is called handling process with a byte stream.

② Character stream :
Incorporates character data.

⇒ There are various operations used in Java file :

↳ canRead() : To check if file is readable

↳ canWrite() : To check writeability of code

Mainly four operations can be performed :
↳ Create new file :
↳ Get file information
↳ Write to a file
↳ Read from a file

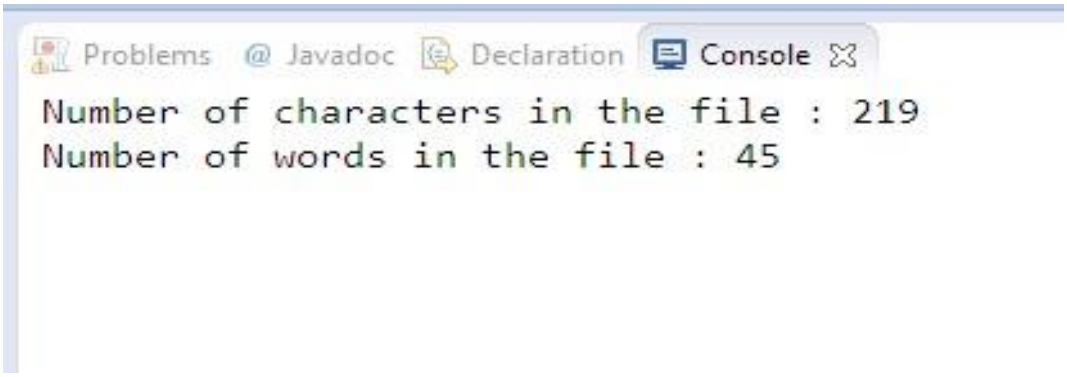
Result
File handling was studied & given programs were implemented successfully.

7.1 CODE:

```
package lab _7;
import java.io.*;

public class Exp7_1
{
  public static void main(String[] args) throws IOException
  {
   BufferedReader reader = null;            int
charCount = 0;
       int wordCount = 0;
       try
      {
  reader = new BufferedReader(new
FileReader("C:\\Users\\PRIYANKA\\Desktop\\SEM5\\eclipse\\java_lab\\src\\lab_7\\pri"));
         String currentLine = reader.readLine();
          while (currentLine != null)
         {
String[] words = currentLine.split(" ");

            wordCount = wordCount + words.length;

            for (String word : words)
            {

charCount = charCount + word.length();
            }

            currentLine = reader.readLine();
         }

         System.out.println("Number of characters in the file : "+charCount);

  System.out.println("Number of words in the file : "+wordCount);
         }

      catch (IOException e)
      {
         e.printStackTrace();
```

```
    }
finally
{          try
        {
           reader.close();
        }
        catch (IOException e)
        {
           e.printStackTrace();
        }
    }
  }
}
```

OUTPUT:

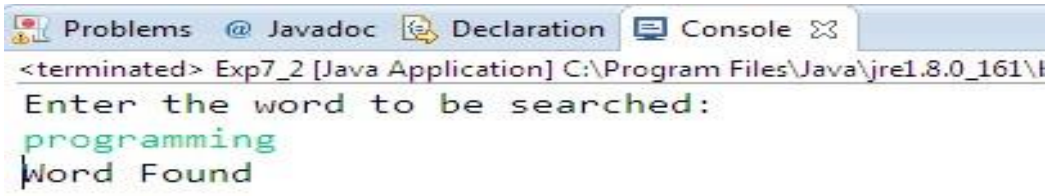


7.2 CODE:

```
package lab _7; import
java.io.*; import
java.util.Scanner; public
class Exp7_2 {
public static void main(String[] args) throws Exception{
          Scanner sc= new Scanner(System.in);
         File file=new File("C:\\Users\\PRIYANKA\\Desktop\\SEM
5\\eclipse\\java_lab\\src\\lab_7\\pri");
          String[] words=null;
   FileReader fr = new FileReader(file);
   BufferedReader br = new BufferedReader(fr);
```

```
    String s;
    System.out.println("Enter the word to be searched:");
    String input = sc.nextLine();
int count=0;
    while((s=br.readLine())!=null)
    {
      words=s.split(" ");
       for (String word : words)
       {
             if (word.equals(input))
             {
               count++;
             }
 }
    }
    if(count!=0)
    {
      System.out.println("Word Found");
    }
    else
    {
      System.out.println("Word Not Found");
    }

      fr.close();
      sc.close();
 }
```

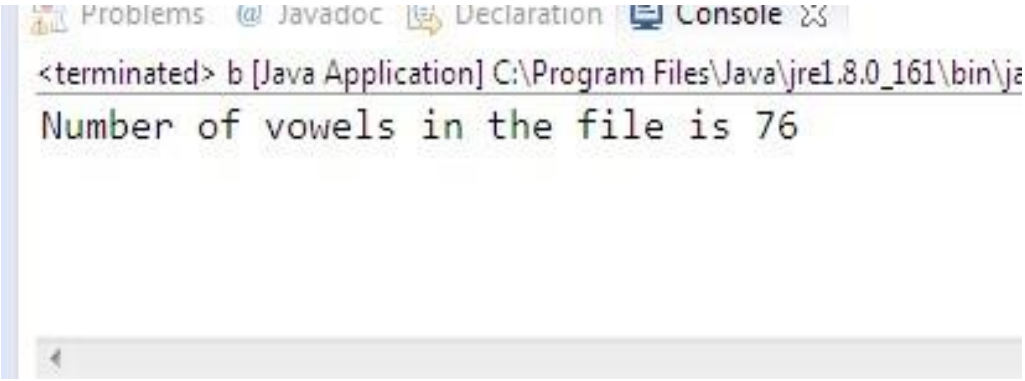
OUTPUT:



7.3 CODE:

```
package lab _7; import
java.util.*; public class
Exp7_3 { public static
void main(String[]
args) throws
Exception{
java.io.File file = new java.io.File("C:\\Users\\PRIYANKA\\Desktop\\SEM
5\\eclipse\\java_lab\\src\\lab_7\\pri");
Scanner input = new Scanner(file);
String fileContent = "";
while (input.hasNext())
        {
        fileContent += input.next() + " ";
                }
        input.close();

        char[] charArr = fileContent.toCharArray();
        int counter = 0;
        for (char c : charArr)
        {
                c=Character.toLowerCase(c);
                        if(c == 'a' || c == 'e' ||c == 'i' ||c == 'o' ||c == 'u')
                       ++counter;
                   }
                   System.out.println( "Number of vowels in the file is "+counter);
                }
                }
```

OUTPUT:



7.4 CODE:

```
package lab _7; import
java.io.*; public class
Exp7_4 {
        public static void main(String[] args) throws Exception{
                BufferedReader reader = null;
                int lineCount = 0;
                try
                {
reader = new BufferedReader(new FileReader("C:\\Users\\PRIYANKA \\Desktop\\SEM
5\\eclipse\\java_lab\\src\\lab_7\\pri"));

        String currentLine = reader.readLine();

        while (currentLine != null)
        {
       lineCount++;
           currentLine = reader.readLine();
        }

      System.out.println("Number of lines in the file : "+lineCount);
    }
    catch (IOException e)
    {
       e.printStackTrace();
 }
finally
```

```
    {
try
        {
           reader.close();
        }
        catch (IOException e)
        {
           e.printStackTrace();
        }
    }
 }
}
```

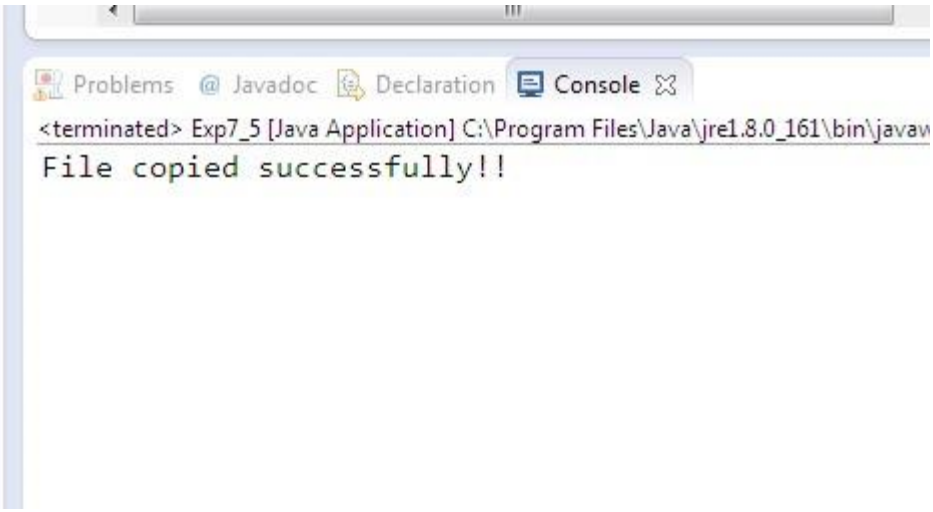
OUTPUT:

7.5 CODE:

```
package lab _7; imort
java.io.File; public class
Exp7_5
{
   public static void main(String[] args)
   {
    FileInputStream instream = null;   FileOutputStream outstream = null;
    try{
       File infile =new File("C:\\Users\\PRIYANKA \\Desktop\\SEM
5\\eclipse\\java_lab\\src\\lab_7\\input.txt");
       File outfile =new File("C:\\Users\\PRIYANKA\\Desktop\\SEM
5\\eclipse\\java_lab\\src\\lab_7\\output.txt");          instream = new
FileInputStream(infile);          outstream = new
FileOutputStream(outfile);          byte[] buffer = new byte[1024];
int length;
```

```
   while ((length = instream.read(buffer)) > 0){          outstream.write(buffer, 0, length);
       }          instream.close();
outstream.close();
       System.out.println("File copied successfully!!");

    }catch(IOException ioe){              ioe.printStackTrace();
     }
   }
}
```

OUTPUT:



Date :
Page No. :

EXPERIMENT 8

Aim

Create a user registration form in AWT & store the details of the user in database table. Also create a login form to check for credentials of the user

Theory

Applet : It is a Java program that runs in Web browsers. An applet can be a fully functional Java application because it has entire Java API at its disposal.

→ An applet is a Java class that extends the java.applet.Applet class

→ A main() method is not involved on an applet, & applet class will not define main()

→ Applets are designed to be embedded within a HTML page

→ When a user opens a HTML page that contains an applet, the code for applet is downloaded to the user machine

→ A JVM is required to view an applet.

Life Cycle of an Applet

→ init : Intended for initialization of anything required by applet

Date :
Page No. :

→ start : Automatically called after the browser calls the init method
→ stop : Automatically called when the users move off the page on which applet sits.
→ destroy : Only called when browser shuts down normally.
→ paint : Invoked immediately after the start method, & also any time the applet needs to repaint itself in the browser

The syntax that brings the classes into the scope of our applet class
→ java.applet.Applet
→ java.awt.Graphics

The applet class
Every applet is the extension of the java.applet.Applet. The base Applet class provides methods that a derived class may call to obtain information & services.
→ Get applet parameters
→ Get network location of HTML file that contains the applet
→ Get network location of applet class directory.
→ Print a status message, fetch an image, fetch an audio clip, resize the applet.

EXPERIMENT -8

AIM: Create a User Registration Form in AWT and store thedetails of the user in Database Table. Also create a Login Formto check for the credentials of the user.[Note: The registrationform should include all the checking of the values before thedata is stored in the database].

Code-

```
package JavaString;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class MyFrame extends JFrame implements ActionListener {

 private Container c;
 private JLabel title;
 private JLabel name;
 private JTextField tname;
 private JLabel mno;
 private JTextField tmno;
 private JLabel gender;
 private JRadioButton male;
 private JRadioButton female;
 private ButtonGroup gengp;
 private JLabel dob;
 private JComboBox date;
 private JComboBox month;
 private JComboBox year;
 private JLabel add;
 private JTextArea tadd;
 private JCheckBox term;
 private JButton sub;
 private JButton reset;
 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" };

// constructor, to initialize the components
// with default values.
public MyFrame()
{
   setTitle("Registration Form");
   setBounds(300, 90, 900, 600);
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   setResizable(false);

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

   title = new JLabel("Registration Form");
   title.setFont(new Font("Arial", Font.PLAIN, 30));
   title.setSize(300, 30);
   title.setLocation(300, 30);
   c.add(title);

   name = new JLabel("Name");
   name.setFont(new Font("Arial", Font.PLAIN, 20));
   name.setSize(100, 20);
   name.setLocation(100, 100);
   c.add(name);

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

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

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

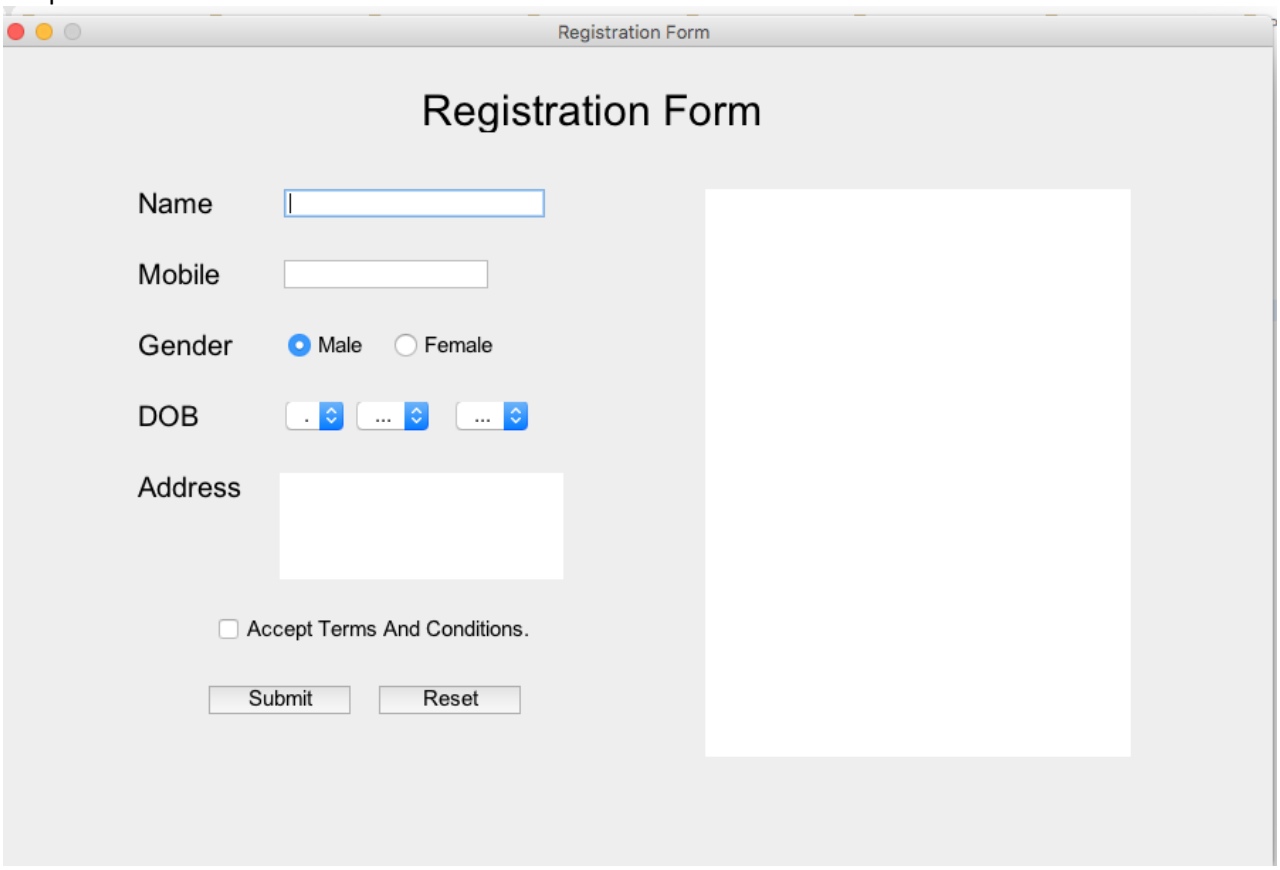
```
gender = new JLabel("Gender");
gender.setFont(new Font("Arial", Font.PLAIN, 20));
gender.setSize(100, 20);
gender.setLocation(100, 200);
c.add(gender);

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

female = new JRadioButton("Female");
female.setFont(new Font("Arial", Font.PLAIN, 15));
female.setSelected(false);
female.setSize(80, 20);
female.setLocation(275, 200);
c.add(female);

gengp = new ButtonGroup();
gengp.add(male);
gengp.add(female);

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

date = new JComboBox(dates);
date.setFont(new Font("Arial", Font.PLAIN, 15));
date.setSize(50, 20);
date.setLocation(200, 250);
c.add(date);

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

year = new JComboBox(years);
year.setFont(new Font("Arial", Font.PLAIN, 15));
year.setSize(60, 20);
year.setLocation(320, 250);
c.add(year);

add = new JLabel("Address");
add.setFont(new Font("Arial", Font.PLAIN, 20));
add.setSize(100, 20);
```

```
add.setLocation(100, 300);
c.add(add);

tadd = new JTextArea();
tadd.setFont(new Font("Arial", Font.PLAIN, 15));
tadd.setSize(200, 75);
tadd.setLocation(200, 300);
tadd.setLineWrap(true);
c.add(tadd);

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.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.setLocation(270, 450);
reset.addActionListener(this);
c.add(reset);

tout = new JTextArea();
tout.setFont(new Font("Arial", Font.PLAIN, 15));
tout.setSize(300, 400);
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);
}

public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == sub) {
        if (term.isSelected()) {
            String data1;
            String data
                = "Name : "
                  + tname.getText() + "\n"
                  + "Mobile : "
                  + tmno.getText() + "\n";
            if (male.isSelected())
                data1 = "Gender : Male"
                        + "\n";
            else
                data1 = "Gender : Female"
                        + "\n";
            String data2
                = "DOB : "
                  + (String)date.getSelectedItem()
                  + "/" + (String)month.getSelectedItem()
                  + "/" + (String)year.getSelectedItem()
                  + "\n";

            String data3 = "Address : " + tadd.getText();
            tout.setText(data + data1 + data2 + data3);
            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 = "";
        tname.setText(def);
        tadd.setText(def);
        tmno.setText(def);
        res.setText(def);
        tout.setText(def);
        term.setSelected(false);
        date.setSelectedIndex(0);
        month.setSelectedIndex(0);
        year.setSelectedIndex(0);
        resadd.setText(def);
```

```
    }
 }
}

class Registration {

 public static void main(String[] args) throws Exception
 {
    MyFrame f = new MyFrame();
 }
}
```

Output-



Date :
Page No. :

CBS-1

Aim

1) WAP using Vector, Arraylist & Hashmap
2) WAP on string tokenizer, string buffer & string builder

Software
VS Code

Theory

→ Arraylist: Part of collection framework & is present in Java.util package. It provides us with dynamic arrays in Java. Though it may be slower than standard arrays but can be helpful in programs where there is a lot of manipulation.

→ Hashmap: It is another data structure available in Java.util package which maps keys & value. It helps us retrieve the value when key is known. It is called a hashmap because it uses a technique called hashing. It is a technique of converting a large string to a smaller string which represents the same thing

→ Vector: We can treat a group of elements as a single object & manipulated with the help of various methods that are available in these classes

Date :
Page No. :

→ String Buffer: → It is a class that has same features as a String class except it is immutable.
→ By using String Buffer also we can handle bunch i.e. set or sequence of characters.
→ At a time only one thread can access String Buffer
→ If it allows multiple threads at a time, we will get data corruption
→ Heap memory by managing String class

→ String Tokenizer: → Class for manipulating string objects in advanced.
→ Available under java.util package.
→ Mainly used for splitting purpose only
→ To overcome the drawback of split method, ~~sout~~ string tokenizer was used.

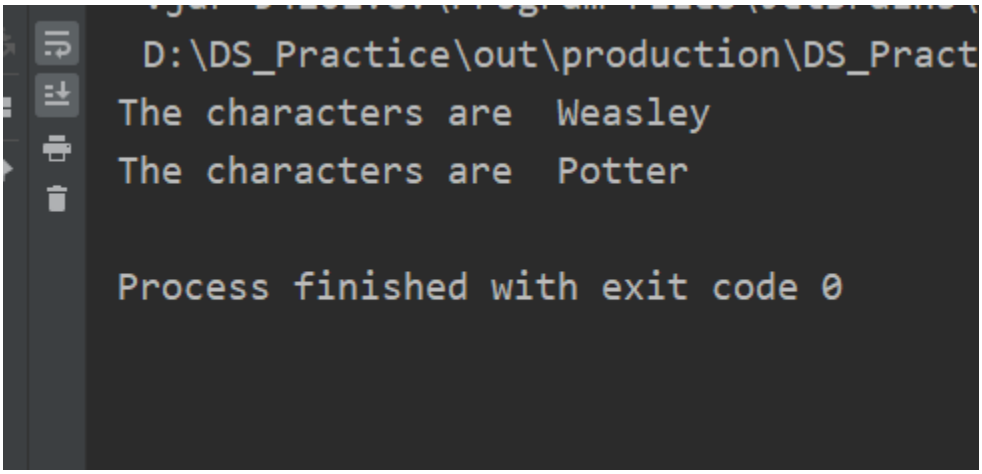
→ String Builder: These are like String objects, except that they can be modified. Internally, these objects are treated like variable-length arrays that contain a sequence of characters. These are mutable classes

CBS-1

1) Using Vector class:-

```
2)import java.util.Vector;
  public class VectorExample {
      Vector vector=new Vector();
  public void addCharacterandPrint()
  { vector.add("Weasley"); vector.add("Potter");
  for(int i=0;i<vector.size();i++)
  { System.out.println("The characters
  are\t"+vector.get(i)); }
  } public static void main(String args[])
      { VectorExample example=new VectorExample();
      example.addCharacterandPrint(); }
  }
```

Output



2.Using Hashmap :-

```
import java.util.HashMap;
import java.util.Set;
public class HashMapExample {
    HashMap hashMap=new HashMap();
    String Books[]={"Famous Five","Goosebumps","Robinson
Crusueo","Nancy Drew","The Cell","The Davinci
```

```
Code","Harry Potter"};
    public void mapAuthors(){
        hashMap.put("Famous Five","Enid Blyton");
        hashMap.put("Goosebumps","R.L.Stine");
        hashMap.put("Nancy Drew","Carolyn Keene");
        hashMap.put("The Cell","Christopher Pike");
        hashMap.put("The Davinci Code","Dan Brown");
        hashMap.put("Harry Potter","J.K. Rowling"); }
        public void getBookList(){
        for(int i=0;i<Books.length;i++)
        { if(hashMap.containsKey(Books[i]))
        {
System.out.println("Author"+(i+1)+":\t"+hashMap.get(Book
s[i])+"\t"+Books[i]); }
        else{ System.out.println("\nThe Imformation
about the author of the book\t"+Books[i]+"\tis not
available\n"); }
        } }
        public static void main(String args[]){
        HashMapExample hashMapExample=new
HashMapExample();
        hashMapExample.mapAuthors();
        hashMapExample.getBookList(); } }
```

Output:-

```
  D:\DS_Practice\out\production\DS_Practice com.company.HashMapExample
Author1:    Enid Blyton Famous Five
Author2:    R.L.Stine   Goosebumps

The Imformation about the author of the book    Robinson Crusueo    is not available

Author4:    Carolyn Keene   Nancy Drew
Author5:    Christopher Pike    The Cell
Author6:    Dan Brown   The Davinci Code
Author7:    J.K. Rowling    Harry Potter

Process finished with exit code 0
```

3. ArrayList example

```java
import java.util.*;
class ArrayListExample1{
    public static void main(String args[]){
        ArrayList<String> list=new
ArrayList<String>();//Creating arraylist
        list.add("Mango");//Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Printing the arraylist object
        System.out.println(list);
    }
}
```

output:-

```
D:\DS_Practice\out\production\DS_Practice com.company.ArrayListExample1
[Mango, Apple, Banana, Grapes]

Process finished with exit code 0
```

4. Using StringTokenizer

```java
import java.util.*;
class StirngTokenizer1
{
    public static void main(String args[])
    {
        System.out.println("Using Constructor 1 - ");
        StringTokenizer st1 =
                new StringTokenizer("Hello Guys How are
you", " ");
```

```
        while (st1.hasMoreTokens())
            System.out.println(st1.nextToken());

        System.out.println("Using Constructor 2 - ");
        StringTokenizer st2 =
                new StringTokenizer("JAVA : Code :
String", " :");
        while (st2.hasMoreTokens())
            System.out.println(st2.nextToken());

        System.out.println("Using Constructor 3 - ");
        StringTokenizer st3 =
                new StringTokenizer("JAVA : Code :
String", " :",  true);
        while (st3.hasMoreTokens())
            System.out.println(st3.nextToken());
    }
}
```

Output

```
 D:\DS_Practice\out\production\DS_Practice com.
Using Constructor 1 -
Hello
Guys
How
are
you
Using Constructor 2 -
JAVA
Code
String
Using Constructor 3 -
JAVA

:

Code

:

String

Process finished with exit code 0
```

5. using StringBuilder

```
public class StringBuilderExample {
    public static void main(String[] argv)
            throws Exception {
        StringBuilder str
                = new StringBuilder();
        str.append("GFG");

        System.out.println("String = "
                + str.toString());
        StringBuilder str1
                = new StringBuilder("AAAABBBCCCC");
        System.out.println("String1 = "
                + str1.toString());
        StringBuilder str2
                = new StringBuilder(10);
        System.out.println("String2 capacity = "
                + str2.capacity());
        StringBuilder str3
                = new StringBuilder(str1);
        System.out.println("String3 = "
                + str3.toString());
    }
}
```

output

```
 D:\DS_Practice\out\production\DS_Practice com.co
String = PRI
String1 = AAAABBBCCCC
String2 capacity = 10
String3 = AAAABBBCCCC

Process finished with exit code 0
```

Date :
Page No. :

CBS-2

Aim

WAP on object serialisation & deserialisation

Theory

Serialisation in Java is mechanism of writing the state of an object into a byte-system. It is mainly used in Hibernate, RMI, JPA, EJB & JMS technology.

Deserialisation is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist in the object.

The byte stream created is platform independent. So, the object serialised on one platform can be deserialised on a different platform.

To make a Java object serializable, we implement java.io.serializable interface.
The object Output stream class contains writeObject() method for serializing an object.
The object Input stream class contains readObject() method for deserializing an object

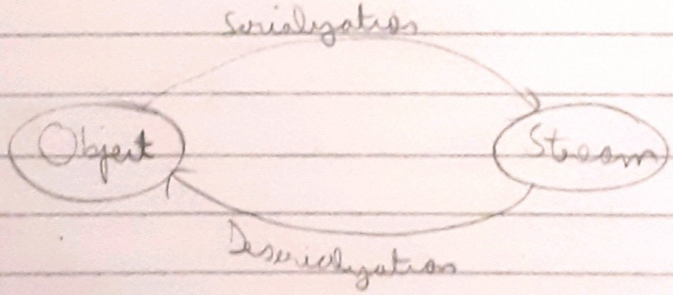
If a parent class has implemented serializable interface

Date :
Page No. :

then child class doesn't need to implement it but vice versa is not true.

Only non static members are saved via serialization process. Static data members & transient data members are not saved via serialization process.

constructor of object is never called when an object is deserialized.
Associated objects must be implementing serializable interface.



Result
Program implemented successfully

Code:

```
import java.io.*;

class Test{
	public static void main(String[] args){
		cbs_2 object = new cbs_2(1, "Hello World");
		String filename = "file.ser";
		try{
			FileOutputStream file = new FileOutputStream(filename);
			ObjectOutputStream out = new ObjectOutputStream(file);

			out.writeObject(object);

			out.close();
			file.close();

			System.out.println("Object has be Serialized");

		}
		catch(IOException ex){
			System.out.println("IOException is Caught");
		}

		cbs_2 object1 = null;
		try{
			FileInputStream file = new FileInputStream(filename);
			ObjectInputStream in = new ObjectInputStream(file);

			object1 = (cbs_2)in.readObject();

			in.close();
			file.close();

			System.out.println("object is Deserialized");
			System.out.println("a: " + object1.a);
			System.out.println("b: " + object1.b);
		}

		catch(IOException ex){
			System.out.println("IOException is Caught");
		}

		catch(ClassNotFoundException ex){
```

```
                    System.out.println("ClassNotFoundException is Caught");
                }


        }
}

class cbs_2 implements java.io.Serializable{
        public int a;
        public String b;

        public cbs_2(int a, String b)
        {
                this.a = a;
                this.b = b;
        }
}
```

Output: