Government Polytechnic, Gandhinagar

Information Technology Department

Sem-5

JAVA PROGRAMMING (3350703)

Laboratory Manual

Practical-1

Aim:

Install JDK, write a simple “Hello World” or similar java program, compilation, debugging, executing using java compiler and interpreter.

Code:

```
public class practical1
{
	public static void main(String args[])
	{
		System.out.print("Hello World");
	}
}
```

Practical-2

Aim:

Write a program in Java to generate first n prime numbers.

Code:

```
public class practical2
{
	public static void main(String args[])
	{
		int i=0;
		int n=0;
		String primenumbers=" ";

		for(i=1;i<=50;i++)
		{
			int count=0;
			for(n=i;n>=1;n--)
			{
				if(i%n==0)
				{
					count = count + 1;
				}
			}
			if(count==2)
			{
				primenumbers = primenumbers + i +" ";
			}
		}
```

```
                System.out.println("prime numbers from 1 to 50 are:");
                System.out.println(primenumbers);
        }
}
```

Practical-3

Aim:

Write a program in Java to find maximum of three numbers using conditional operator.

Code:

```
public class practical3
{
	public static void main(String[] args)
	{
		int a=20,b=30,c=10;
		int max = (a> b && a>c) ? a : ((b>a && b>c) ? b : c) ;
		System.out.println(max + " is maximum");
	}
}
```

Practical-4

Aim:

Write a program in Java to find second maximum of n numbers without using arrays.

Code:

```
class practical4
{
	public static void main(String args[])
	{
		int a=10,b=20,c=30;

		if((a>b)&&(a>c))
		{
			if(b>c)
			{
			System.out.println("b is second_largest");
			}
			else
			{
			System.out.println("c is second_largest");
			}
		}
		else if((b>c)&&(b>a))
		{
			if(a>c)
```

```
                {
                System.out.println("a is second_largest");
                }
                else
                {
                System.out.println("c is second_largest");
                }
            }
            else if((c>a)&&(c>b))
            {
                if(a>b)
                {
                System.out.println("a is second_largest");
                }
                else
                {
                System.out.println("b is second_largest");
                }
            }
            else
            {
            System.out.println("all are equal");
            }
        }
}
```

Practical-5

Aim:

Write a program in Java to reverse the digits of a number using while loop.

Code:

```
class ReverseNumber
{
	public static void main(String args[])
	{
		int num = 1234, reversed = 0;
		while(num!=0)
		{
			int digit = num%10;
			reversed= reversed * 10 + digit;
			num/=10;
		}
			System.out.println("Reversed Number:"
			+reversed);
	}
}
```

Practical-6

Aim:

Write a program in Java to convert number into words & print it.

Code:

```
public class practical6
{
	public void NintoW(int n, String ch)
	{
		String one[] = { " ", " One", " Two", " Three", " Four", " Five", " Six", " Seven", " Eight", " Nine", " Ten"," Eleven", " Twelve", " Thirteen", " Fourteen", "Fifteen", " Sixteen", " Seventeen", " Eighteen"," Nineteen" };


		String ten[] = { " ", " ", " Twenty", " Thirty", " Forty", " Fifty", " Sixty", "Seventy", " Eighty", " Ninety" };


		if (n > 19)
		{
			System.out.print(ten[n / 10] + " " + one
			[n % 10]);
		}
		else
		{
			System.out.print(one[n]);
		}
		if (n > 0)
			System.out.print(ch);
	}
```

```
        public static void main(String[] args)
        {

                int n=66;
                System.out.print(n);
                if (n <= 0)
                {
                        System.out.println("Enter numbers greater than 0");
                }
                else
                {
                        practical6 a = new practical6();
                        a.NintoW((n / 10000000) % 100, " crore");
                        a.NintoW(((n / 100000) % 100), " lakh");
                        a.NintoW(((n / 1000) % 100), " thousand");
                        a.NintoW(((n / 100) % 10), " hundred");
                        a.NintoW((n % 100), " ");
                }
        }
}
```

Practical-7

Aim:

Write programs in Java to use Wrapper class of each primitive data types.

Code:

```
public class practical7
{
	public static void main(String[] args)
	{
		int i=7;
		float f=3.5f;
		char c='k';
		boolean b=true;

	//Converting primitive data type into object.
		Integer i1 = new Integer(i);
		Float f1= new Float(f);
		Character c1 = new Character(c);
		Boolean b1 = new Boolean(b);

	//getting primitive data type from object.
		System.out.println("i = " + i1.intValue());
		System.out.println("f = " + f1.floatValue());
		System.out.println("c = " + c1.charValue());
		System.out.println("b = " + b1.booleanValue());
	}
}
```

Practical-8

Aim:

Write a program in Java to multiply two matrix.

Code:

```
class MatrixMultiplication
{
	public static void main(String args[])
	{

	//creating two matrices

		int a[][]={{1,1,1},{2,2,2},{3,3,3}};
		int b[][]={{1,1,1},{2,2,2},{3,3,3}};

	//creating another matrix to store the multiplication of two matrices

		int c[][]=new int[3][3];  //3 rows and 3 columns

	//multiplying and printing multiplication of 2 matrices
		for(int i=0;i<3;i++)
		{
			for(int j=0;j<3;j++)
			{
				c[i][j]=0;
				for(int k=0;k<3;k++)
				{
				c[i][j]+=a[i][k]*b[k][j];
```

```
                    }//end of k loop

                System.out.print(c[i][j]+" ");  //printing matrix element
                }//end of j loop

                    System.out.println();//new line
            }

        }
}
```

Practical-9

Aim:

Write a static block which will be executed before main( ) method in a class.

Code:

```
class practical9
{
        // static block will be executed before main method.
    static
    {
        System.out.println("Hello..");
    }

    // main method
    public static void main(String args[])
    {
        System.out.println("Bye");
    }
}
```

Practical-10

Aim:

Write a program in Java to demonstrate use of this keyword. Check whether this can access the private members of the class or not.

Code:

```
class practical10
{
	private int n;
	void setdata(int n)
	{
		this.n = n; //accessing private member (n) using this keyword.
	}
	void getdata()
	{
		System.out.println("n=" + n);
	}
}
public class practical10_main
{
	public static void main(String[] args)
	{
		practical10 obj1 = new practical10();
		obj1.setdata(7);
		obj1.getdata();
	}
}
```

Practical-11

Aim:

Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object.

Code:

```
class OverloadDemo
{
	void display()
	{
		System.out.println("No parameters");
	}
	void display(int a)
	{
		System.out.println("a: " + a);
	}
	void display(double a)
	{
		System.out.println("double a: " + a);
	}
	void display(int a, int b)
	{
		System.out.println("a and b: " + a + " " + b);
	}
}
class OverloadDemo_main
{
```

```
    public static void main(String[] args)
    {
        OverloadDemo ob = new OverloadDemo();
        ob.display();
        ob.display(10);
        ob.display(123.25);
        ob.display(10, 20);
    }
}
```

Practical-12

Aim:

Write a program in Java to demonstrate the use of private constructor and also write a method which will count the number of instances created using default constructor only.

Code:

```
class Box
{
	int length;
	int width;
	int height;
	static int count=0;

	Box()
	{
		length = 10;
		width = 10;
		height = 10;
		Box.count_instances();
	}

	Box(int l, int w,int h)
	{
		length = l;
		width = w;
		height = h;
	}
```

```
        //following method counts number of objects created using
default constructor.
    static void count_instances()
    {
        count++;
    }
}
public class practical12
{
    public static void main(String[] args)
    {
        Box b1 = new Box();
        Box b2 = new Box();
        System.out.println("No of Objects Created: " + Box.count );
    }
}
```

Practical-13

Aim:

Write a program in Java to demonstrate the use of 'final' keyword in the field declaration. How it is accessed using the objects.

Code:

```
class practical13
{
	final int a=15;
	final void funn1()
	{
		//a=30;final can not change
		System.out.println("say hello.."+a);
	}
}
class practical13_main
{
	public static void main(String[] args)
	{
		practical13 f1 = new practical13();
		f1.funn1();
	}
}
```

Practical-14

Aim:

Develop minimum 4 program based on variation in methods i.e. passing by value, passing by reference, returning values and returning objects from methods

Code:

Practical: 14.1 (Pass by Value)

```
class passbyvalue
{
	int data=41;
	void change(int data)
	{
		data=data+100;
	}

public static void main(String args[])
 {
	passbyvalue pa=new passbyvalue();
	System.out.println("before change = "+pa.data);
	pa.change(141);
	System.out.println("after change = "+pa.data);
 }
}
```

Practical: 14.2 (Pass by Reference)

```
class passbyreference
{
```

```
  int data=100;
  void change(passbyreference re)
  {
    re.data=re.data+200;
  }
  public static void main(String args[])
   {
          passbyreference re = new passbyreference();
          System.out.println("before change = "+re.data);
      re.change(re);
      System.out.println("after change = "+re.data);

   }
}
```

Practical: 14.3 (Return by Value)

```
class RBV
{
        int i;
        void set_data(int a)
        {
        i=a;
        }
        int incrByOne()
        {
        i++;
```

```
        return (i);
        }
}
public class Example
{
        public static void main(String[] args)
        {
        RBV obj1 = new RBV();
        obj1.set_data(5); //Pass by Value
        int value = obj1.incrByOne(); // Return by Value
        System.out.println("i : " + value);
        }
}
```

Practical: 14.4 (Return by Objects)

```
class RBR
{
        int i;
        RBR Copy()
        {
        RBR temp = new RBR();
        temp.i = i;
        return temp;
        }
        void display()
        {
        System.out.println("i : "+ i);
```

```
        }
}
public class Example
{
        public static void main(String[] args)
        {
        RBR obj1 = new RBR();
        obj1.i=7;
        RBR obj2 = obj1.Copy(); //Return by Reference
        obj1.display();
        obj2.display();
        }
}
```

Practical-15

Aim:

Aim :- Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance

Code:

Practical: 15.1 (Single Inheritance)

```
class a
{
	void disp()
	{
		System.out.println("Class A Call");
	}
}
class b extends a
{
	void disp1()
	{
		System.out.println("Class B Call");
	}
}

class inheritance_main
{
	public static void main(String args[])
	{
		b o1 = new b();
		o1.disp();
		o1.disp1();
	}
}
```

Practical: 15.2 (Multilevel Inheritance)

```
class a
{
	void disp()
	{
		System.out.println("A call");
	}
}
class b extends a
{
	void disp1()
	{
		System.out.println("B call");
	}
}
class c extends b
{
	void disp2()
	{
		System.out.println("C call");
	}
}
class multilevel_main
{
	public static void main(String args[])
	{
		c o1=new c();
		o1.disp();
		o1.disp1();
		o1.disp2();
```

```
	}
}
```

Practical: 15.3 (Hierarchical Inheritance)

```
class a
{
	void disp()
	{
		System.out.println("a");
	}
}
class b extends a
{
	void disp()
	{
		System.out.println("b");
	}
}
class c extends b
{
	void disp()
	{
		System.out.println("c");
	}

}
class d extends a
{
	void disp()
	{
```

```
                System.out.println("d");
        }
}
class override_main
{
        public static void main(String args[])
        {
                b o1 = new b();
                c o2 = new c();
                d o3 = new d();
                o1.disp();
                o2.disp();
                o3.disp();
        }
}
```

Practical-16

Aim:

Create a class to find out whether the given year is leap year or not. (Use inheritance for this program)

Code:

```
class a
{
	int year = 2004;
	void LeapYear()
	{
		System.out.println("year : " + year);
	}
}
class b extends a
{
	void LeapYear1()
	{
		if(year%4==0)
		{
			System.out.println("is leap year");
		}
		else
		{
			System.out.println("is not leap year");
		}
	}
}
```

```
class practical16
{
	public static void main(String args[])
	{
		b o1 = new b();
		o1.LeapYear();
		o1.LeapYear1();
	}
}
```

Practical-17

Aim:

Write an application that illustrates how to access a hidden variable. Class A declares a static variable x. The class B extends A and declares an instance variable x. display( ) method in B displays both of these variables.

Code:

```
class A
{
	static int x;
}
class B extends A
{
	int x;
	void display()
	{
		//super.x prints value of variable x of class A
		System.out.println("Value of x in A :" +super.x);
		//x prints value of variable x of class B
		System.out.println("Value of x in B :"+x);
	}
}
public class practical17
{
	public static void main(String[] args)
	{
		A obj = new A();
```

```
            obj.x=10;
            B obj1 = new B();
            obj1.x=20;
            obj1.display();
        }
}
```

Practical-18

Aim:

Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.

Code:

```
class A
{
	int x;
	A(int a)
	{
	x=a;
	}
}
class B extends A
{
	int y;
	B(int a,int b)
	{
		super(a); //super() calls the constructor of the class A
		y = b;
	}
	void display()
	{
		System.out.println("X :" + x);
		System.out.println("Y :" + y);
	}
```

```
}
public class practical18
{
	public static void main(String[] args)
	{
		B obj = new B(2,4);
		obj.display();

	}
}
```

Practical-19

Aim:

Write a program that illustrates interface inheritance. Interface P12 inherits from both P1 and P2. Each interface declares one constant and one method. The class Q implements P12. Instantiate Q and invoke each of its methods. Each method displays one of the constants.

Code:

```
interface P1
{
	int I = 1;
	void showI();
}
interface P2
{
	int J = 2;
	void showJ();
}
interface P12 extends P1, P2
{
	int K = 3;
	void showK();
}
class Q implements P12
{
	public void showI()
	{
		System.out.println("I : " + I);
	}
```

```
	public void showJ()
	{
		System.out.println("J : " + J);
	}
	public void showK()
	{
		System.out.println("K : " + K);
	}
}
public class practical19
{
	public static void main(String[] args)
	{
		Q obj = new Q();
		obj.showI();
		obj.showJ();
		obj.showK();
	}
}
```

Practical-20

Aim:

Write an application that illustrates method overriding in the same package and different packages. Also demonstrate accessibility rules in inside and outside packages.

Code:

1. Method Overriding:

```
class A
{
	void show()
	{
		System.out.println("Inside Class-A");
	}
}
class B extends A
{
	// Overriding method show of class A
	void show()
	{
		System.out.println("Inside Class-B");
	}
}
public class override_demo
{
	public static void main(String[] args)
	{
		B obj = new B();
```

```
            obj.show();
            //show() method of class-B will be called.
        }
}
```

2. Accessibility Rule inside and outside package

2.1 Accessibility Rule Inside package

```
package p1;
class Protection
{
        int n=1;
        public int n_pub=2;
}
public class practical20
{
        public static void main(String[] args)
        {
                Protection p1 = new Protection();
                System.out.println("n = " +p1.n);
                System.out.println("n_pub = " +p1.n_pub);
        }
}
```

2.2 Accessibility Rule outside package

```
package p1;
```

```
public class Protection
{
	int n = 1;
	public int n_pub = 2;
}
import p1.*;
public class Example
{
	public static void main(String[] args)
	{
		Protection p1 = new Protection();
		//System.out.println("n = " + p1.n); //error
		System.out.println("n_pub = " +p1.n_pub);
	}
}
```

Practical-21

Aim:

Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area()in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.

Code:

```
abstract class Shape
{
	abstract void area();
}
class Triangle extends Shape
{
	int base,height;
	Triangle(int b,int h)
	{
		base=b;
		height=h;
	}
	void area()
	{
		float ar = (0.5f)* (base*height);
		System.out.println("Area of Triangle: "+ar);
	}
}
class Rectangle extends Shape
{
```

```
	int length,width;
	Rectangle(int l,int w)
	{
		length=l;
		width=w;
	}
	void area()
	{
		float ar = length*width;
		System.out.println("Area of Rectangle: "+ar);
	}
}
class Circle extends Shape
{
	int radius;
	Circle(int r)
	{
		radius = r;
	}
	void area()
	{
		float ar = 3.14f * radius * radius;
		System.out.println("Area of Circle: "+ar);
	}
}
public class practical21
{
```

```
        public static void main(String[] args)
        {
                Triangle t1 = new Triangle(2,4);
                t1.area();
                Rectangle r1 = new Rectangle(3,5);
                r1.area();
                Circle c1= new Circle(2);
                c1.area();
        }
}
```

Practical-22

Aim:

Write a program in Java to demonstrate implementation of multiple inheritance using interfaces.

Code:

```
interface A
{
	void method1();
}
interface B
{
	void method2();
}
class InterfaceDemo implements A , B
{
	public void method1()
	{
		System.out.println("Inside Method - 1");
	}
	public void method2()
	{
		System.out.println("Inside Method - 2");
	}
}
public class practical22
{
	public static void main(String args[])
```

```
	{
		InterfaceDemo obj = new InterfaceDemo();
		obj.method1();
		obj.method2();
	}
}
```

Practical-23

Aim:

Write a program in Java to demonstrate use of final class.

Code:

```
final class A
{
	int i;
	void showI()
	{
		System.out.println("I : " + i);
	}
}


class B //extends A	// ERROR! Can't subclass A because class A is final
{
	int j;
	void showJ()
	{
		System.out.println("J : " + j);
	}
}
public class practical23
{
	public static void main(String[] args)
	{
		A obj = new A();
```

```
            B obj1 = new B();
            obj.i = 2;
            obj1.j = 3;
            obj.showI();
            obj1.showJ();
        }
}
```

Practical-24

Aim:

Write a program in Java to develop user defined exception for 'Divide by Zero' error.

Code:

```
class MyException extends Exception
{
	public String toString()
	{
		return "Attempt to divide by zero.";
	}
}
public class practical24
{
	public static void main(String[] args)
	{
		try
		{
			throw new MyException();
		}
		catch(MyException e )
		{
			System.out.println(e);
		}
	}
}
```

Practical-25

Aim:

Write a program in Java to demonstrate multiple try block and multiple catch exception

Code:

```
public class practical25
{
	public static void main(String[] args)
	{
		try
		{
			int a= 4/0;
			try
			{
				int b[] = {1,2};
				System.out.println(b[5]);
			}
			catch(ArrayIndexOutOfBoundsException e)
			{
				System.out.println("Array index is out of bound.");
			}
		}
		catch(ArithmeticException e)
		{
			System.out.println("Divide by zero is not defined");
		}
	}
}
```

Practical-26

Aim:

Write an small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing of Rs 400.00, Rs 300.00 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 500 thereafter..

Code:

```
class MyException extends Exception
{
	public String toString()
	{
		return "Not Sufficient Fund.";
	}
}
class Account
{
	double amount;
	Account(double a)
	{
		amount = a;
	}
	void deposite(double a)
	{
		amount = amount + a;
		System.out.println(a + " rupees deposited.");
		System.out.println("New balance is : " + amount);
	}
```

```
	void withdraw(double a)
	{
		if((amount - a)<0)
		{
			System.out.println("Attempt to withdraw " + a + "
rupees is failed.");


			try
			{
				throw new MyException();
			}
			catch(MyException e)
			{
				System.out.println(e);
			}
		}
		else
		{
		amount = amount -a;
		System.out.println(a + " rupees withdrawn.");
		System.out.println("New balance is : " + amount);
		}
	}
}
public class practical26
{
	public static void main(String[] args)
	{
```

```
        Account a1 = new Account(1000);
        a1.withdraw(400.00);
        a1.withdraw(300.00);
        a1.withdraw(500.00);
    }
}
```

Practical-27

Aim:

Write a program that executes two threads. One thread displays "Thread1" every 2,000 milliseconds, and the other displays "Thread2" every 4,000 milliseconds. Create the threads by extending the Thread class

Code:

```
class NewThread extends Thread
{
	NewThread(String s)
	{
		super(s);
	}
	public void run()
	{
		if( getName().equals("Thread1")==true)
		{
			for(int i=1;i<=5;i++)
			{
				System.out.println("Thread1");

				try
				{
					Thread.sleep(2000);
				}
				catch(InterruptedException e)
				{
					System.out.println("Exception Occurred.");
```

```
                        }
                    }
                }
                else
                {
                    for(int i=1;i<=5;i++)
                    {
                        System.out.println("Thread2");

                        try
                        {
                            Thread.sleep(4000);
                        }
                        catch(InterruptedException e)
                        {
                            System.out.println("Exception Occurred.");
                        }
                    }
                }
        }
}
class practical27
{
        public static void main(String[] args)
        {
                NewThread t1 = new NewThread("Thread1");
                NewThread t2 = new NewThread("Thread2");
```

```
        t1.start();
        t2.start();
    }
}
```

Practical-28

Aim:

Write a program that executes two threads. One thread will print the even numbers and the another thread will print odd numbers from 1 to 50.

Code:

```
class NewThread extends Thread
{
	NewThread(String threadname)
	{
		super(threadname);
	}
	public void run()
	{
		if( getName().equals("Odd")==true)
		{
			for(int i=1;i<=50;i=i+2)
			{
				System.out.print(" " + i);

				try
				{
					Thread.sleep(1000);
				}
				catch(InterruptedException e)
				{
					System.out.println("Exception Occurred.");
```

```
                        }
                    }
                }
                else
                {
                    for(int i=2;i<=50;i=i+2)
                    {
                        System.out.print(" " + i);

                        try
                        {
                            Thread.sleep(1000);
                        }
                        catch(InterruptedException e)
                        {
                            System.out.println("Exception Occurred.");
                        }
                    }
                }
        }
}
public class practical28
{
        public static void main(String[] args)
        {
                NewThread t1 = new NewThread("Odd");
                NewThread t2 = new NewThread("Even");
```

```
            t1.start();
            t2.start();
        }
}
```

Practical-29

Aim:

Write a program in Java to demonstrate use of synchronization of threads when multiple threads are trying to update common variable.

Code:

```
class Counter
{
	int c=0;
	synchronized void increment()
	{
		c++;
		try
		{
			Thread.sleep(1000);
		}
		catch(InterruptedException e)
		{
			System.out.println("caught: " + e);
		}
		System.out.println("C = " + c);
	}
}
class NewThread extends Thread
{
	Counter obj;
	NewThread(Counter o)
```

```
	{
		obj = o;
	}
	public void run()
	{
		obj.increment();
	}
}
public class practical29
{
	public static void main(String[] args)
	{
		Counter c1 = new Counter();
		NewThread t1 = new NewThread(c1);
		t1.start();
		NewThread t2 = new NewThread(c1);
		t2.start();
		NewThread t3 = new NewThread(c1);
		t3.start();
	}
}
```

Practical-30

Aim:

Write a program in Java to create, write, modify, read operations on a Text file

Code:

```
import java.io.*;
public class practical30
{
	public static void main(String[] args)
	{
		try
		{
			// Creating new Text file
			File file = new File("E:/newfile.txt");
			if (!file.exists())
			{
				file.createNewFile();
				System.out.println("File created Successfully.");
			}

			// Writing to new Text file
			FileOutputStream fout=new
FileOutputStream("E:\\newfile.txt");
			String s="India is Great.";
			byte b[]=s.getBytes();
			fout.write(b);
			fout.close();
```

```
                System.out.println("Writing Complete!");


                //Reading from Text file
                FileInputStream fin = new
FileInputStream("E:\\newfile.txt");
                int i;
                while((i=fin.read())!=-1)
                {
                        System.out.print((char)i);
                }
                fin.close();


                // Modifying Text file
                fout=new FileOutputStream("E:\\newfile.txt");
                s="I love India.";
                b=s.getBytes();
                fout.write(b);
                fout.close();
                System.out.println("\nModification Complete!");


                //Reading from Text file
                fin = new FileInputStream("E:\\newfile.txt");
                while((i=fin.read())!=-1)
                {
                        System.out.print((char)i);
                }
                fin.close();
        }
```

```
            catch (IOException e)
            {
                System.out.println("I/O Exception occurred.");
            }
        }
}
```