Name:

Enrolment No:

UPES
UNIVERSITY WITH A PURPOSE

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

End Semester Examination, December 2019

Course: JAVA Programming	**Semester: VII**
Program: B.Tech. EE/EE spz in BCT	**Time: 03 hrs.**
Course Code: CSEG 313	**Max. Marks: 100**

Instructions:

SECTION A [20 Marks]

		Marks	CO
Q 1	Enlist the four different ways to make an object eligible for garbage collection.	4	CO2
Q2	What feature of JAVA language is demonstrated by the undermentioned code? Suggest the output for the same.	[1+3]	CO2, CO3

```
import java.io.*;
interface intfA
{
        void geekName();
}

interface intfB extends intfA
{
        void geekInstitute();
}

class sample implements intfB
{
        @Override
        public void geekName()
        {
                System.out.println("Rohit");
        }

        @Override
        public void geekInstitute()
        {
                System.out.println("JIIT");
        }
```

	public static void main (String[] args)
	{
	sample ob1 = new sample();
	
	// calling the method implemented
	// within the class.
	ob1.geekName();
	ob1.geekInstitute();
	}
	}		

Q3	Identify the six basic steps required to the Execution of Servlets.	4	CO3
Q4	Develop a Java code to illustrate maximum of three numbers using ternary operator.	4	CO1
Q5	Briefly explain the significance of "static" keyword in "public static void main()" w.r.t. Java programming.	4	CO1, CO2
SECTION B [40 Marks]			
Q 6	Suggest the syntax for the following commands: a) Cascading "member access operator" b) Declaring "Abstract class" c) Default constructor d) Implement Code reusability e) Import Scanner class	[5x2]	CO1, CO2

Q7	a) Write a Java program to demonstrate that static block and static variables are executed in order they are present in a program
	
	b) Predict the desired output of the following JAVA program:
	// Java program to demonstrate requesting
	// JVM to run Garbage Collector
	
	public class Test
	{
	public static void main(String[] args) throws InterruptedException
	{
	Test t1 = new Test();
	Test t2 = new Test();
	
	t1 = null;
	System.gc();
	t2 = null;
		[5+5]	CO2

	`Runtime.getRuntime().gc();`
	`}`
	`@Override`
	`protected void finalize() throws Throwable`
	`{`
	`System.out.println("Garbage collector called");`
	`System.out.println("Object garbage collected : " + this);`
	`}`
	`}`		

Q8	**Rewrite the same code using “Buffer class” and explain the reason of different output obtained w.r.t. both the codes.**
	
	// Code using Scanner Class
	
	`import java.util.Scanner;`
	`class Differ`
	`{`
	`public static void main(String args[])`
	`{`
	`Scanner scn = new Scanner(System.in);`
	`System.out.println("Enter an integer");`
	`int a = scn.nextInt();`
	`System.out.println("Enter a String");`
	`String b = scn.nextLine();`
	`System.out.printf("You have entered:- "`
	`+ a + " " + "and name as " + b);`
	`}`
	`}`
		10	**CO1, CO3**

Q9	**Why Java is not a purely Object-Oriented Language? What are the essential features that are necessary to qualify a programming language to be entitled as "OOP"?** **Enlist at least 1 programming language that is a purely object oriented.** **OR** **Answer the following :** **a) Is main method compulsory in JAVA?** **b) How are parameters passed in JAVA?** **c) Why Java doesn't support Multiple Inheritance? Justify your answer with the adequate code.** **d) Advantages of a Java Servlet**	**10** **[2+2+3 +3]**	**CO1, CO2, CO3**
SECTION-C [40 Marks]			
Q 10	**I. Elucidate the following built-in Exceptions in Java:** **a) ArithmeticException** **b) ArrayIndexOutOfBoundsException** **c) ClassNotFoundException** **d) FileNotFoundException** **e) IOException** **f) InterruptedException** **g) NoSuchFieldException** **h) NoSuchMethodException** **i) NullPointerException** **j) NumberFormatException** **k) RuntimeException** **l) StringIndexOutOfBoundsException** **II. Fill in the blanks with appropriate Comment(s) and suggest the output of the following program:** **// Java program to demonstrate user defined exception** **// This program throws an exception whenever balance** **// amount is below Rs 1000** **class MyException extends Exception** **{** **//store account information**	**[12+8]**	**CO1, CO2, CO3**

```
        private static int accno[] = {1001, 1002, 1003, 1004};

        private static String name[] = {"Nish", "Shubh", "Sush",
    "Abhi", "Akash"};

        private static double bal[] =
                {10000.00, 12000.00, 5600.0, 999.00, 1100.55};


        MyException() { } //_____________


        MyException(String str) { super(str); } //________________


        public static void main(String[] args)
        {
                try {
                        // __________________
                        System.out.println("ACCNO" + "\t" +
    "CUSTOMER" +

        "\t" + "BALANCE");

                        // ____________________________
        for (int i = 0; i < 5 ; i++)
        {
           System.out.println(accno[i] + "\t" + name[i] +

                                                        "\t" + bal[i]);

                       // display own exception if balance < 1000
                if (bal[i] < 1000)
                     {
                         MyException me = new MyException("Balance is
    less than 1000");
                        throw me;
                      }
                        }
                } //________________

                catch (MyException e)
                {
                        e.printStackTrace();
                 }
        }
```

	} **OR** i. Detail about the access modifier(s) in JAVA. ii. How could a user access any class inside a package? Justify your answer via providing a suitable syntax/code. iii. Suggest the output of the following program:	[8+4+2 +6]	

```
import java.io.*;

        public class Test
        {

            public static void main(String[] args)
            {
                    System.out.println("Hi Geek (from main)");
                    Test.main("Geek");
            }
            public static void main(String arg1)
            {
                    System.out.println("Hi, " + arg1);
                    Test.main("Dear Geek","My Geek");
            }
            public static void main(String arg1, String arg2)
            {
                    System.out.println("Hi, " + arg1 + ", " + arg2);
            }
        }
```

iv. Guess the output of the following JAVA Program:

```
abstract class Shape
{
        String color;

        // these are abstract methods
        abstract double area();
        public abstract String toString();

        // abstract class can have constructor
        public Shape(String color) {
                System.out.println("Shape constructor called");
                this.color = color;
        }
        // this is a concrete method
        public String getColor() {
                return color;
        }
}
class Circle extends Shape
{
        double radius;

        public Circle(String color,double radius) {

                // calling Shape constructor
                super(color);
                System.out.println("Circle constructor called");
                this.radius = radius;
        }

        @Override
        double area() {
                return Math.PI * Math.pow(radius, 2);
        }

        @Override
        public String toString() {
                return "Circle color is " + super.color +
                                        "and area is : " + area();
        }

}
class Rectangle extends Shape{

        double length;
        double width;

        public Rectangle(String color,double length,double width) {
                // calling Shape constructor
                super(color);
                System.out.println("Rectangle constructor called");
                this.length = length;
                this.width = width;
        }

        @Override
        double area() {
                return length*width;
        }
```

```
    @Override
    public String toString() {
        return "Rectangle color is " + super.color +
                                        "and area is : " + area();
    }

}
public class Test
{
    public static void main(String[] args)
    {
        Shape s1 = new Circle("Red", 2.2);
        Shape s2 = new Rectangle("Yellow", 2, 4);

        System.out.println(s1.toString());
        System.out.println(s2.toString());
    }
}
```

Q 11	**a) Suggest the output of the undermentioned code w.r.t. the concept of Thread Priority under Multi-Threading in JAVA:**	**[10+10]**	**CO1, CO2, CO3**

```
// Java program to demonstrate getPriority() and setPriority()

import java.lang.*;

class ThreadDemo extends Thread
{
        public void run()
        {
                System.out.println("Inside run method");
        }

        public static void main(String[]args)
        {
                ThreadDemo t1 = new ThreadDemo();
                ThreadDemo t2 = new ThreadDemo();
                ThreadDemo t3 = new ThreadDemo();

                System.out.println("t1 thread priority : " +
                                                t1.getPriority());
                System.out.println("t2 thread priority : " +
                                                t2.getPriority());
                System.out.println("t3 thread priority : " +
                                                t3.getPriority());

                t1.setPriority(2);
                t2.setPriority(5);
                t3.setPriority(8);

                // t3.setPriority(21); will throw IllegalArgumentException

                System.out.println("t1 thread priority : " +
                                                t1.getPriority());
                System.out.println("t2 thread priority : " +
                                                t2.getPriority());
                System.out.println("t3 thread priority : " +
                                                t3.getPriority());

                // Main thread

                System.out.print(Thread.currentThread().getName());
                System.out.println("Main thread priority : "
                                        +
Thread.currentThread().getPriority());
```

```
                    // Main thread priority is set to 10

                    Thread.currentThread().setPriority(10);
                    System.out.println("Main thread priority : "
                                        +Thread.currentThread().getPriority());
            }
    }
```

b) Brief about:

i. **‘implement’ keyword**
ii. **‘extends’ keyword**
iii. **Non-static method(s)**
iv. **‘this’ keyword**
v. **JRE**
vi. **Inheritance**
vii. **Class PATH**
viii. **Encapsulation**
ix. **Generic Servlets**
x. **Exception Handling**