

Paper Solution

Class: FE
Sub: CP-II

Q.1 a)

Working of JVM: 5 Marks
Feature description: 5Marks

JVM:

Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the *Java Virtual Machine (JVM)*. The JVM is an *interpreter for bytecode*. Translating a Java program into bytecode helps makes it much easier to run a program in a wide variety of environments. As only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all interpret the same Java bytecode. If a Java program were compiled to native code, then different versions of the same program would have to exist for each type of CPU connected to the Internet. This is, of course, not a feasible solution.

Thus, the interpretation of bytecode is the easiest way to create truly portable programs.

The fact that a Java program is interpreted also helps to make it secure. Because the execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside of the system.

Architectural Neural:

A central issue for the Java designers was that of code longevity and portability. Operating system pgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever."

b)

Data Types in Java: 5 Marks
Wrapper Class: 5 Marks

Data Types in Java:

Java defines eight simple (or elemental) types of data: **byte** , **short** , **int** , **long** , **char** , **float**, **double** , and **boolean** .

These can be put in four groups:

- Integers This group includes **byte** , **short** , **int** , and **long** , which are for whole-valued signed numbers.
- Floating-point numbers This group includes **float** and **double** , which represent numbers with fractional precision.
- Characters This group includes **char** , which represents symbols in a character

set, like letters and numbers.

- Boolean This group includes **boolean** , representing true/false values.

Name	Width (in bits)	Range
long	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	-2,147,483,648 to 2,147,483,647
short	16	-32,768 to 32,767
byte	8	-128 to 127
double	64	1.7e-308 to 1.7e+308
float	32	3.4e-038 to 3.4e+038
char	2	
boolean	1	

Wrapper Class:

Wrapper Classes are used to convert primitive type elements into objects.

Primitive data type	Wrapper class
byte	Byte
short	Short
int	Int
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

The methods defined within Wrapper Classes are used to:

- 1) Convert Primitive nos to Object invoking constructor of Wrapper Class:

```
int i=5;
Integer intOb = new Integer(i);

float f=5.8f;
Float floatOb = new Float (f);

char ch='b';
Character chOb = new Character(ch);
```

```
boolean b = true;  
Boolean Ob = new Boolean(b);
```

2) Convert object to primitive nos using intValue()

```
int i = intOb.intValue();  
float f= floatOb.float Value();  
double d = doubleOb.doubleValue()
```

3) Convert primitive data typed values to String objects using static toString().

```
String s1=Integer.toString(i);  
String s2=Float.toString(f);  
String s3=Double.toString(d);
```

4) Convert string objects to numeric objects using valueOf()

```
Integer Ob =Integer.valueOf(str)  
Float Ob=Float.valueOf()  
Double ob= Double.valueOf()
```

5) Convert String objects to primitive data type values using parse method:

```
int i = Integer.parseInt(str)  
float f= Float.parseFloat(str)  
double d = Double.parseDouble(str)
```

Q.2 a)

Program to find Factorial of number:

Logic calculating factorial:

5Marks

Accepting number at run time:

5Marks

```
import java.io.*;
class Factorial
{
    public static void main(String[] args)
    {
        DataInputStream dis=new DataInputStream(System.in);

        try
        {
            char ch;
            do
            {
                System.out.println("\nEnter number of which factorial is
                to be calculated");
                int num=Integer.parseInt(dis.readLine());
                int fact=1;

                for(int i=num; i>0; i--)
                {
                    fact=fact*i;
                }
                System.out.println("Factorial of "+num+" is:\n"+fact);

                System.out.println("Enter the choice y=yes & n=no");
                ch=(char)(dis.read());
            }
            while(ch=='y' || ch=='Y');

        }
        catch (Exception e)
        {
            System.out.println("Caught"+e);
        }
    }
}
```

OUTPUT

path>java Factorial

Enter number of which factorial is to be calculated

6

Factorial of 6 is:

720

b) **Employee details showing Program:**

Class Employee definition with data members empid, empname, designation, salary and methods get_employee, show_grade(), show_employee(): **7Marks**

Call to methods: **3Marks**

```
import java.io.*;
public class employee
{
    String empname;
    int empid;
    String designation;
    double salary;
    char grade;

    public void get_employee()
    {
        try{
            DataInputStream in=new DataInputStream(System.in);

            System.out.println("Enter name\n");
            empname=in.readLine();
            System.out.println("Enter code");
            empid=Integer.parseInt(in.readLine());
            System.out.println("Enter designation\n");
            designation=in.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Exception is caught");
        }
    }
    public void putInfo()
    {
        System.out.println"\nName:"+
        empname+"\nCode:"+empid+"\n";
        System.out.println"\nDesignation:"+ designation );
    }

    public void showGrade()
    {
        try
        {
            DataInputStream dis=new DataInputStream(System.in);
            System.out.println("enter grade of employee");
            grade=(char)(dis.read());
        }
        catch(IOException e)
        {
            System.out.println("Exception is caught"+e);
        }
    }
}
```

```

    }
    public void getSalary()
    {
        if(grade=='A'||grade=='a')
            salary=78000;
        else if(grade=='B'||grade=='b')
            salary=60000;
        else if(grade=='C'||grade=='c')
            salary=50000;
        else if(grade=='D'||grade=='d')
            salary=35000;
        else    salary=15000;
    }

    public void putSalary()
    {
        System.out.println("Grade:\t"+grade);
        System.out.println("SALARY:\t"+salary);
    }
    void show_employee()
    {
        putInfo();
        putSalary();
    }
}
class employee_details
{
    public static void main(String[] args) throws Exception
    {
        DataInputStream dis=new DataInputStream(System.in);
        System.out.println("How many employees");
        int n=Integer.parseInt(dis.readLine());

        employee e[]=new employee [n];

        for(int i=0;i<n;i++)
        {
            e[i].get_employee();
            e[i].getSalary();
        }

        for(int i=0;i<n;i++)
        {
            System.out.println("\n\nDetails of Employee");
            e[i].display();
        }
    }
}

```

Q. 3 a) To count upper case, lower case letters, blank spaces, digits from the string:

Each carry :

2 ½ Marks

b) To arrange accepted nos in ascending order:

Accepting number s and storing into an array:

4 Marks

Logic to have ascending order of it:

6 Marks_____

```
import java.io.*;
class ArrayDemo
{
    public static void main(String[] args)
    {
        DataInputStream dis=new DataInputStream(System.in);
        int n,i;
        int a[];
        int temp;
        try
        {
            System.out.println("How many elements");
            n=Integer.parseInt(dis.readLine());
            a=new int[n];

            for(i=0;i<n;i++)
                a[i]=Integer.parseInt(dis.readLine());

            for(int pass=1;pass<n;pass++)
            {
                for(i=0;i<=pass-1;i++)
                {
                    if (a[i]>a[i+1])
                    {
                        temp=a[i];
                        a[i]=a[i+1];
                        a[i+1]=temp;
                    }
                }
            }
            System.out.println("\n sorted array elements:\n");
            for(i=0;i<n;i++)
                System.out.println(a[i]);
        }
        catch (Exception e)
        {
            System.out.println("Exception is caught"+e);
        }
    }
}
```

OUTPUT

```
path>java ArrayDemo
How many elements
5
```

21

81

41

31

51

sorted array elements:

21

31

41

51

81

Q.4 a)

Inheritance:

Types: 4Marks

Inability of Multiple Inheritance: 2Marks

Alternative to Multiple Inheritance with exp. : 4Marks

Types:

Single Inheritance, Multilevel Inheritance, Multiple Inheritance, Hybrid Inheritance

Inability of Multiple Inheritance:

Java doesn't support multiple inheritance.

Alternative to Multiple Inheritance :

Interface

b)

difference between abstract, interface, final class: 5Marks

Program to calculate area: 5Marks

Abstract class:

If the class itself fails to define method, just declare it within it by preceding declaration with abstract keyword. and declare the class enclosing abstract method as abstract class. It becomes the responsibility of subclass to define the method.

Interface:

An alternative to abstract class is an interface, no requiring abstract keyword.

Difference between class and interface.

class is defined with class keyword, whereas interface definition requires interface keyword.

class may contain either method definition or declaration or both, interface supports only method declaration.

if method declared in class, should precede with abstract keyword, and class is defined with same, in case of interface no need to use abstract keyword as by default methods declared within it act as abstract.

No restriction on access specifier while defining method, whereas method defined within interface should have public access specifier.

Abstract class is always extended by some other class to define method, interface is requiring class implementing declared method.

Variables declared in interface treated as global(static), constant(final). not creating the copies per instance.

Final:

Final class can not be overided. It should start with keyword final.

Program:

```
import java.io.*;
abstract class Figure
{
    double length;
```

```

        double breadth;
        double radius;
        double base;
        double height;

        abstract double area();
    }

class Rectangle extends Figure
{
    double area()
    {
        double result= length*breadth;
        return result;
    }
}
class Circle extends Figure
{
    double area()
    {
        double result= 3.14 * radius * radius;
        return result;
    }
}
class Triangle extends Figure
{
    double area()
    {
        double result= 0.5*base*height;
        return result;
    }
}

class abstractDemo
{
    public static void main(String[] args)throws IOException,
    NumberFormatException
    {
        DataInputStream dis=new DataInputStream(System.in);

        Figure ref;
        Rectangle r=new Rectangle();

        System.out.println("Enter length");
        r.length=Double.parseDouble(dis.readLine());

        System.out.println("Enter breadth");
        r.breadth=Double.parseDouble(dis.readLine());
    }
}

```

```
        ref=r;
        System.out.println("Area of rectangle: "+ ref.area());

        Circle c=new Circle();

        System.out.println("Enter radius");
        c.radius=Double.parseDouble(dis.readLine());

        ref=c;
        System.out.println("Area of rectangle: "+ref.area());

        Triangle t=new Triangle();
        System.out.println("Enter base");
        t.base=Double.parseDouble(dis.readLine());

        System.out.println("Enter height");
        t.height=Double.parseDouble(dis.readLine());

        ref=r;
        System.out.println("Area of triangle: "+ref.area());
    }
}
```

```

import java.io.*;
abstract class Figure
{
    double length;
    double breadth;
    double radius;
    double base;
    double height;

    abstract double area();
}

class Rectangle extends Figure
{
    double area()
    {
        double result= length*breadth;
        return result;
    }
}

class Circle extends Figure
{
    double area()
    {
        double result= 3.14 * radius * radius;
        return result;
    }
}

class Triangle extends Figure
{
    double area()
    {
        double result= 0.5*base*height;
        return result;
    }
}

class abstractDemo
{
    public static void main(String[] args)throws IOException, NumberFormatException
    {
        DataInputStream dis=new DataInputStream(System.in);

        Figure ref;
        Rectangle r=new Rectangle();

        System.out.println("Enter length");
        r.length=Double.parseDouble(dis.readLine());
    }
}

```

```
System.out.println("Enter breadth");
r.breadth=Double.parseDouble(dis.readLine());

ref=r;
System.out.println("Area of rectangle: "+ ref.area());

Circle c=new Circle();

System.out.println("Enter radius");
c.radius=Double.parseDouble(dis.readLine());

ref=c;
System.out.println("Area of rectangle: "+ref.area());

Triangle t=new Triangle();
System.out.println("Enter base");
t.base=Double.parseDouble(dis.readLine());

System.out.println("Enter height");
t.height=Double.parseDouble(dis.readLine());

ref=r;
System.out.println("Area of triangle: "+ref.area());
    }
}
```

Q.5 a)

Difference between Single threaded & multithreaded application:

3 Marks

Benefits of Multithreading:

2 Marks

Multithreading explanation:

5 Marks

b)

Package definition:

3Marks

Demonstration:

7Marks

```
package pkg1;
import java.io.*;
public class personal_details
{
    static String company_name="Parle-G";
    String name;
    int code;

    public void getInfo()
    {
        try{
            DataInputStream in=new DataInputStream(System.in);

            System.out.println("Enter name\n");
            name=in.readLine();
            System.out.println("Enter code");
            code=Integer.parseInt(in.readLine());
        }
        catch(Exception e)
        {
            System.out.println("Exception is caught");
        }
    }
    public void putInfo()
    {
        System.out.println("\nCompany:"+company_name+"\nName:"+
            name+"\nCode:"+code+"\n");
    }
}

package pkg1;

public interface emp_grade
{
    public void empGrade();
}
```

```

package pkg1;
import java.io.*;

public class salary_details extends personal_details implements emp_grade
{
    double salary;
    char grade;

    public void empGrade()
    {
        try
        {
            DataInputStream dis=new DataInputStream(System.in);
            System.out.println("enter grade of employee");
            grade=(char)(dis.read());
        }
        catch(IOException e)
        {
            System.out.println("Exception is caught"+e);
        }
    }

    public void putGrade()
    {
        System.out.println("Grade:\t"+grade);
    }

    public void getSalary()
    {
        if(grade=='A'||grade=='a')
            salary=78000;
        else if(grade=='B'||grade=='b')
            salary=60000;
        else if(grade=='C'||grade=='c')
            salary=50000;
        else if(grade=='D'||grade=='d')
            salary=35000;
        else
            salary=15000;
    }

    public void putSalary()
    {
        putGrade();
        System.out.println("SALARY:\t"+salary);
    }
}

```

```

package pkg1;

public class employee_details extends salary_details
{
    /*public void getmaininfo()
    {
        getInfo();
        empGrade();
        getSalary();
    }*/

    public void display()
    {
        putInfo();
        putSalary();
    }
}

package pkg2;

import java.io.*;

public class Employee_package extends pkg1.salary_details
{
    public static void main(String[] args) throws Exception
    {

        DataInputStream dis=new DataInputStream(System.in);
        System.out.println("How many employees");
        int n=Integer.parseInt(dis.readLine());

        pkg1.employee_details e[]=new pkg1.employee_details[n];

        for(int i=0;i<n;i++)
        {
            e[i]=new pkg1.employee_details();
            e[i].getInfo();
            e[i].empGrade();
            e[i].getSalary();
        }

        for(int i=0;i<n;i++)
        {
            System.out.println("\n\nDetails of Employee");
            e[i].display();
        }
    }
}

```

Q.6 a)

Exception Handling keywords explanation:

5 Marks

Example:

5 Marks

```
import java.io.*;
class OddNoException extends Exception
{
    OddNoException(String s1)
    {
        super(s1);
    }
}
class OwnException
{
    public static void main(String[] args)
    {
        DataInputStream dis=new DataInputStream(System.in);
        int num;
        try
        {
            System.out.println("Enter the number");
            num=Integer.parseInt(dis.readLine());
            if(num%2!=0)
            {
                OddNoException e=new OddNoException("Number is odd");
                throw e;
            }
        }
        catch (OddNoException e)
        {
            System.out.println("Exception is caught " +e);
            System.out.println("Exception is caught " +e.getMessage());
        }
        catch(IOException e)
        {
            System.out.println("Exception is caught " +e);
        }
        finally
        {
            System.out.println("Outside of try and catch.");
        }
    }
}
path>javac OwnException.java
path>java OwnException
Enter the number
3
Exception is caught OddNoException: Number is odd
Exception is caught Number is odd
Outside of try and catch.
```

b)

Java applet : 3 Marks
Displaying "Hello Java": 4 Marks
Relation between applet and HTML: 3 Marks

Applet is a small java program that is required to embed into any supporting language like HTML.
An applet is not stand alone program like an application.

```
import java.awt.*;
import java.applet.*;
/*
<applet CODE="Simple.class" WIDTH=400 HEIGHT=100>
</applet>
*/

public class Simple extends Applet
{
    //Overridden methods
    public void init()
    {

    }

    public void start()
    {

    }

    public void stop()
    {

    }

    public void destroy()
    {

    }

    public void paint ( Graphics g )
    {
        g.drawString ( "Hello Java!",10,10);
    }
}
```

Q.7 a) Difference between C++ and Java

Each difference carry 1 Mark

b) System arrayCopy method

Package, class name where it is available: 2Marks

Example : 3Marks

c) Static members:

Defining static members: 1Mark

Accessing it inside same class: 1Mark

Accessing it outside the class: 2Marks

Use of it: 1Mark

d) Access Specifiers:

Types : 2Marks

Example demonstrating access to it 3Marks

e) Thread Synchronization:

Definition: 1Mark

Need: 2Marks

Example: 2Marks