KOWSIK NANDAGOPAND
CSE S4 ROLL NO31

12 June 2020

ROLL NO 31

OO Design & Programming

PART A

(1.) Java is a compiler and interpreter.
When we first execute the .java file it is made to a .class file using javac compiler. It makes the object code for the java. This code is platform independent.
While ~~to ex~~ during execution .class is verified using byte code verifier and this ~~verifier~~ is done by the JVM which is an interpreter and platform dependent.

.java → .class → byte code verifier → ...
(.java → .class: Compiler; byte code verifier → ...: Interpreter.)

(2.) • char this Must be Too Long;
This does not seems to be problem. Java does not restrict the programmer to use variable name length less than ~~a~~ specific value.

- int bubble=0, toil=9, trouble=8
  These variables are of int with correct name convention. None of the variables comes in keyword.

- String 8ball;
  This is invalid variable name since variable should not start with integer even though integer is allowed.

- int double;
  double & Double are key words should not be used as variable name. Hence invalid.

P.T.O

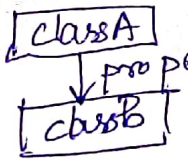
(3.) Polymorphism:

It is a special property of the OOPs that we can use the methods used in the OOPS class having same name for multiple functionality. Means that we can do overloading of methods that takes in different parameters and can be used to do various tasks. In other words its ability of OOPs to take many form of same thing. Single symbol to represent multiple.

eg:- Question 4 function overloading.

Inheritance

Inheritance is a property of OOPs. The class defined in OOPs Language can borrow or inherit the properties from the classes by making the class parent class.



(4) Function overloading.

functions/methods defined in a class can be defined multiple times but taking in different parameters. So the same function name can provide different functionality based on input data.

```
eg:- class A {
        public void fun1(int i){

        }
        public void fun1 (double i){

        }
     }
```

Here fun 1 is overloaded with int & double

Function overriding

A function/method defined in subclass with same name as the function defined in the parent class. The function in subclass thus hides or overrides the class defined in the parent class. This is said to be function overriding.

eg:-

```
class A {
    public void fun1(int){
    }
}
class B extend A {
    public void fun1(int){
    }
}
```

Here the fun1 in class A is overriden in class B after inheriting the class A in class B. It is not mandatory to keep @override decorator in java befor overriding function. (3/6)

PART B

P.T.O

6

(a)
```
public class Calculation functions {
    double area = -1; // Not defined
    public A Circle (double radius) {
        area = 3.14159 * radius * radius;
        return(area);
    }
```

PART B

(6)

(a)

```
import java.util.*;
public class AreaCalculator {
double area = -1; // undefined;
    public double area (double radius){
        area = 3.14159 * radius * radius;
        return (area);
    }
    public double area (double length, double breadth){
        area = length * breadth;
        return (area);
    }
    public double area (double a, double b, double c){
        double s = (a+b+c)/2; // Heron's formula
        area = Math.sqrt(s*(s-a)*(s-b)*(s-c));
        return (area);
    }
}

public void static main (String args[]){
    AreaCalculator a = new AreaCalculator();
    Scanner s = new Scanner (System.in);
    String menu = "1.Circle 2.Rectangle 3.Triangle";
    double a, b, c, radius, length, breadth, area;
    System.out.println (menu);
    switch (s.nextInt()){
        case 1:
            System.out.println ("Enter radius")
            radius = s.nextDouble();
        area = a.area (radius);
            System.out.println ("Area =" + area);
            break;
```

(4½)

```
case 2 :
    System.out.println(" Enter length & breadth ");
    length = s.nextDouble();
    breadth = s.nextDouble();
    area = a.area(length, breadth);
    System.out.println(" Area = " + area);
    break;

case 3:
    System.out.println(" Enter sides of triangle a, b, c ");
    a = s.nextDouble();
    b = s.nextDouble();
    c = s.nextDouble();
    area = a.area(a, b, c);
    System.out.println(" Area using herons = " + area);
    break;
default:
    System.out.println(" No such option ");
    break;
} //close switch

} //close main class
```

(b)

```
public class Box {
    ~~height~~
    ~~double height, breadt~~
    double Height, Width, Depth;
    Box(double h, double w, double d){
        this.Height = h;
        this.Width = w;
        this.Depth = d;
```

(5/6)

```
    }
    public double volume() {
        double volume = Height * width * Depth;
        return volume;
    }
}

public static void main (String args[]){
    Box box1 = New Box(1, 2, 3);
    Box box2 = new Box(1, 1, 1);

    String.out.println ("Volume of box1" +box1.volume()
                        + "unit cube");
    String.out.println ("Volume of box2" + box2.volume()
                        + "unit cube");
}

}
```