2018/ 2019 EXAMINATION

OBJECT ORIENTED PROGRAMMING

SECTION A

Part 1

a) **Static** is a java keyword used to distinguish class variables from instance.

b) **Final** is a key word used in java to define constants in a class.

c) **A package** is a collection of related classes in java.

d) The output of java compiler is called a **byte code.**

e) **Polymorphism** is the ability of an object in java to appear in different forms based on the environment it is in.

f) **Inheritance** is the ability of a java class to take on properties and behaviors of another related class.

g) In Object Oriented Programing a **methods** are used to model the behaviors of objects of a class.

h) **A constructor** is a special method within a java class that has the same name as the class and such a method has no return type.

i) **Extends** is a java key word used to indicate that class A is a subset of class B.

j) **A variable** in Objected Oriented Programing is a combination of the return type, name and a parameter list of the method and is used to uniquely identify a method in a class.

Part II

```
public class Staff{
public static String staffNo;
        public static String staffName;
        public static int age;

        public void me() {
                System.out.println(staffNo+"s\n"+staffName+"\n"+age);
                        }
        }
import java.util.Scanner;
public class Fulltimestaff extends Staff {

        public static double basicsalary;
        public static double deductions;
        public static double monthly_salary;
        public void monthlysalary(double basicsalary,double deductions) {
```

```
                monthly_salary=basicsalary-deductions;
System.out.println(staffNo+"\n"+staffName+"\n"+age+"\n"+monthly_salary);

        }
        public static void main(String[] args) {
                Scanner x= new Scanner(System.in);
                System.out.println("Enter the staff Number:");
                staffNo=x.next();
                System.out.println("Enter the staff Name:");
                staffName=x.next();
                System.out.println("Enter the age:");
                age=x.nextInt();
                System.out.println("Enter the basic salary:");
                basicsalary=x.nextDouble();
                System.out.println("Enter the deductions:");
                deductions=x.nextDouble();

                Fulltimestaff obj1 =new Fulltimestaff();
                obj1.monthlysalary(basicsalary,deductions);

        }

}
Import java.util.Scanner;

public class Parttimestaff extends Staff {

public int no_of hours worked;

public double tax;

public double Pnetsalary;

public void monthlysalary (int no.of hours worked, double tax) {

 Pnetsalary=((no_of hours worked * 2500) – tax (no_of hours worked * 2500))

System.out.println(staffNo+ “\n” +  staffName + “\n” + staffAge + “\n” + Pnetsalary);

          }

Public static void main (String [ ] args ) {

Scanner y = new scanner (System.in);

StaffNo = y.next( );

staffName = Y.next( );
```

```
staffAge = Y.nextInt( );
tax = Y.nextDouble( );
Parttimestaff obj1 = new ParttimeStaff ( );
obj1.monthlySalary (no_ofhours worked, tax);
                }
            }
```

**Section B**

QUESTION 1

a) A constructor is a special method in OOP that initializes an instantiated objects.

b) It has no return type

c)

```
Public class Me{
public static int x;
public static int sum ( int x) {
        return ( x+ 10);
    }
}
    Public class You extends Me {
        public static int sum (int x) {
            return ( x+20);
        }
    }
```

d)

```
public class Student {
public static int x;
public static int y;
```

```
 public Student ( int x, int y){
        this.x = x;
        this.y=y;
}

public Student( int x, int y, int z){
        this.x=x;
        this.y=y;
        this .z=z;
        }
public static void main( String [ ] args){
Student obj1=new Student(1,6,8);
System.out.println(obj1.x);
System.out.println(obj1.y);
System.out.println(obj1.z);

     }
        }
```

QUESTION 2

a)
```
public Human ( string tname, int tage ){
name = tname;
age = tage;
}
```

b)
```
public Human ( string name, int age ){
this.name = name;
this.age = age;
}
```

c) Human obj1 = new Human ( name, age) ;

d) public class Human{

public String name;

public public int age;

public Human ( ) {

}

e) Human ob1 = new Human ( " ", 0);

**QUESTION 4**

**a(i)** A nested class is a class with in another class

**(ii)** inner class is a non static nested class

**(iii)** An anonymous class refers to an inner class that doesn't have a name

**b(i) y** cannot be resolved to a variable in output before being declared and initialized

**entry variable** is not given a data type

the main function where execution of a program is missing

**(ii)** Declare y before the output statement

Establish the main function in the program after the nested class

Put a data type to a variable Entry

(iii) class Hash{

int x;

Entry entry;

Hash(entry){

this.entry = entry;

}

```
        class Entry{
        int y;
        void display ( ) {
        System.out.println("value =" + x);
                }
        }
}
```

**QUESTION 5**

a) When it has a key word abstract

   When it has at least one of its methods abstract

b) Final for constant

   Abstract for method

c)
```
interface Me {
    public abstract int sum ( int  w);
     public final interview x = 10;
                }

public class  You{
public static int me;
public static String name;
        public You(int me, String name){
                this.me = me;
                this.name = name;
                        }
    public class Them extends you implements Me{
                public static int number;
        public Them(int number){
                super(me,name);
```

```
            this.number = number;
                    }
        public int sum(int w){
            return (w*interviewer);
}
public static void main(String [ ] args){
 Them obj1 = new Them (20, “Moses”, 15);
 obj1.sum(16);
System.out.println(obj1.sum(16));
        }
}
```

# End

## 2017/2018 examination

## Object oriented programming

**PART I**

Write down only the most appropriate word to fill the blank space in each of the following statements about Java as an object oriented programming language.

a) In Java, **Class** is a generic representation for a set of objects with similar features.
b) When a Java program is compiled, the output is called **java byte code** and it is the input to the Java interpreter.
c) **Static variable** is a type of variable whose value belongs to the entire class.

d) **Instance variable** is a type of variable whose value belongs to the object and each object can have its own different value.
e) When writing an object-oriented program, **methods** are used to model the behaviours of objects of that class.
f) In a Java program, **final** is a keyword used to indicate that a certain attribute is a constant.
g) When implementing inheritance, **extends** is the keyword used to show that a class is a subclass of another class.
h) If a class has a method that should be accessible only with in that class, **private** keyword is used as its visibility modifier.
i) When class defines a method with the keyword **final**, then any attempt by the subclass to override such a method will cause the sub class to hide the super class' definition of this method and this results into a compile time error.
j) If a class is defined with the keyword **final** after its visibility modifier, then this class can never be inherited.

(2 Marks @, Total= 20 Marks)

## PART II

A factory produces two types of products, namely; AnimalFeed and Plastic. Each product has a ProductName, LineManager, Quantity and Price. For AnimalFeed, each feed has AnimalType, FeedFormula and ExpiryDate. On the other hand, each Plastic has Capacity(amount of content the plastic product it can store). Using the concept of inheritance in Java in this case study, write down one super class and two subclasses and make sure that each of the subclasses is able to capture the details of its objects through the keyboard and print them on the screen. [Hint: make use of input classes like Scanner to capture the details of the objects of each class].

```
public class product{

        public static String ProductName;

        public static String LineManager;

        public static double quantity;

        public static double price;

public Product(String ProductName, String LineManager,double quantity, double price){

                this.ProductName= ProductName;

                this.LineManager=LineManager;

                this.quantity= quantity;

                this.price= price;


                }
```

```
        }
import java.util.Scanner;
public class AnimalFeed{
        public static String Animaltype;
        public static String FeedFormular;
        public static String expiryDate;
        public AnimalFeed(String Animaltype, String FeedFormular,String expiryDate){
                super(ProductName,LineManager,quantity,price);
            this.Animaltype=Animaltype;
            this.FeedFormular=FeedFormular;
            this.expiryDate=expiryDate;

            }
        public static void main(String[ ]  args) {
            Scanner m= new Scanner(System.in);
            System.out.println("Enter the product Name");
                productName=m.next();
            System.out.println("Enter the line manager");
                LineManager=m.next();
            System.out.println("Enter the quantity");
                quantity=m.nextDouble();
            System.out.println("Enter the price");
                price=m.nextDouble();
            System.out.println("Enter the Animaltype");
                Animaltype =m.next( );
            System.out.println("Enter the Feed Formular");
                FeedFormular = m.next( );
            System.out.println("Enter the expiry date");
                expiryDate = m.next( );
```

```
System.out.println (productName+ “\n”+ LineManager+ “\n”+ quantity+ “\n”+ price + “\n”+
Animaltype+ “\n” + Feedformular + “\n” + expiryDate);
}

        }


import java.util.Scanner;
public class Plastic{
        public static double capacity;
        public AnimalFeed(String Animaltype, String FeedFormular,String expiryDate){
                        super(ProductName,LineManager,quantity,price);
                this.capacity = capacity;

                }
public static void main (String [ ] args){
        Scanner t = new Scanner( System.in);
System.out.println(“Enter the product Name”);
        productName=t.next();
System.out.println(“Enter the line manager”);
        LineManager=t.next();
System.out.println(“Enter the quantity”);
        quantity=t.nextDouble();
System.out.println(“Enter the price”);
        price=t.nextDouble();
System.out.println(“Enter the capacity”);
        capacity= t.nextDouble();
System.out.println (productName+ “\n”+ LineManager+ “\n”+ quantity+ “\n”+ price + “\n”+
capacity);
```

```
}
        }
```

(4 Marks for super class, 8 Marks for each sub class, Total=20 Marks)

## SECTION B [60 Marks]- Choose only 3 Questions

### Question One

a) Assuming that a given class A has members with visibility modifiers provided in the table below, copy and complete the table by **clearly ticking** the cell where that class member is accessible. The cells where the class' members are not accessible should be left blank.

| Visibility Modifier | Accessible in that same class | Accessible in a class in same package | Accessible in the sub class | Accessible in a class from another package |
|---|---|---|---|---|
| public | ✓ | ✓ | ✓ | ✓ |
| protected | ✓ | ✓ | ✓ | |
| default | ✓ | ✓ | ✓ | |
| private | ✓ | | | |

(1 Mark @ correctly ticked cell, total=10 marks )

b) Using an example of your choice, write down a set of Java classes to demonstrate method overriding in Java. Your overriden method must be called in the class that overrides it.
(10 marks)

```
public class Me{

        public static int x;

        public static int y;
```

```
        public Me(int x, int y){
                this.x=x;
                this.y=y;
                        }
         public static int sum(int x, int y){
                return(x+y);
                                }
        public class You extends Me{
                public You( ){
                super(x,y);
                                }

                public static int sum(int x, int y){
                        return(x+y+10);
                                }
public static void main (String[ ] args){
        You.sum(2,4);
System.out.println(You.sum(2,4));
}
   }
```

Question Two

a) What do you understand by the term constructor in Java? (2 Marks)
**A constructor is a special method in object oriented programming that initialises instantiated objects .**

b) using an example of your choice, write down a Java program that demonstrates constructor overloading.Your program must make use of the overloaded costructors in the main().
(8 Marks)

```
public class Me{
```

```
        public static int x;
        public static int y;
    public Me(int x, int y){
        this.x=x;
        this.y=y;
            }
    public Me(int x, int y, int z){
        this.x=x;
        this.y=y;
        this.z=z;
            }
   public static void main (String [ ] args){
      Me obj1= new Me(2,3,4);
   System.out.println(obj1.x+ "\n" + obj1.y + "\n" + obj1.z);
}
      }
```

c) A farmer wishes to partition his piece of land into different shapes. Some shapes should be circular while others should be rectagular. He wishes to know the length of the birbed wire required for each shape. Write down a Java program that applies the concept of method overloading to help this farmer. The overloaded method should be called in the main(). (10 Marks)

```
import java.util.Scanner;
public  class Farmer{
        public static final pie= 3.14;
       public static double length;
       public static double width;
        public static double radius;
public static double perimeter(double radius){
            return (2*(pie*radius));
```

```
                                    }
public static double perimeter(double length, double width){
        return (2*(length + width));
                                    }
public static void main(String [ ]args){
        Scanner x =new Scanner(System.in);
        System.out.println("enter the radius of the circle");
        radius=x.nextDouble( );
        Farmer.perimeter(radius);

        System.out.println('enter the length of the rectangle");
        length=x.nextDouble();
        System.out.println('enter the width of the rectangle");
        width= x.nextDouble();

        Farmer.perimeter(length,width);

        System.out.println("The length of barbed wire for the rectangular shape is" +
Farmer.perimeter(length,width));
        System.out.println("The length of barbed wire for the circular shape is" +
Farmer.perimeter(width));
            }
        }
```

## Question Three

a) Given that A is an array of n elements, write down the index of the second last element in A. (2 Marks)

**A[n-2]**

b) In an experiment to determine the mean length of a seedling in a garden, 6 values were obtained. Write down a Java program that uses an array to store these values and uses

this very array to determine and print on the screen the mean length.
(6 Marks)

```
import java.util.Scanner;

public class Average{

        public static int i;

        public static double mean;

        public static int sum;

public static void main (String [ ] args){

        Scanner y= new Scanner(System.in);

        int[ ] x = new int[6];

        for(i=0;i<x.length;i++){

                System.out.println("Enter the value of array:" + i);

                X[i] = y.nextInt();

}

        for(i=0;i<x.length;i++){

                sum=sum+x[i];

}

        mean = sum/x.length;

System.out.println("The mean length=" + mean);

        }

}
```

c) Given that A is a 2x2 matrix whose first row contains values 4, 6 (in that order) and second row contains value 1, 5 (in that order). Using the concept of multi-dimensional arrays, write down a Java program that can be used to determine and print on the screen the determine of A and its inverse. Remember that a matrix with determinant as zero has no inverse. Recall from elementary math that

Inverse of A= (1/Determinant)* Transpose of A. (12 Marks)

```
public class Matrix {

 public static int i;

 public static int j;

 public static double determinant;

 public static double inverse;
```

```
    public static void main(String[] args) {
        double x[][]= {
                {4,6},
                {1,5}
        };
        determinant=((x[0][0]*x[1][1])-(x[0][1]*x[1][0]));
        System.out.println(determinant);
        double[][]y = {
            {1/determinant*5,1/determinant*(-6)},
            {1/determinant*(-1),1/determinant*4}
        };
        for(i=0;i<x.length;i++) {
            for(j=0;j<x.length;j++) {
                System.out.println(y[i][j]);
            }
        }
    }
}
```

**Question Four**

a) For each of the following statements about Java programming language, indicate TRUE or FALSE and give a reason for your answer. (1 Mark @)

i. Every Java class has a super class except one class.
**False because super class is only required when different classes contain same properties.**

ii. A method declared as static in the superclass can be overriden in the subclass.
**True because its not declared constant.**

iii. Every Java class has a default constructor.
**True because by default a compiler creates a constructor to initialise objects even if it's not created**

iv. The main() method of a Java class can be non-static.
**False**

v. During method overriding, a method in the superclass declared as public can be overriden as private by the subclass.
**False because we cannot reduce the visibility of the inherited method**

vi. An abstract class can have a subclass.
**True because a class can provide implementation for an abstract class**

vii. An abstract class can be instantiated.
**False because it does not provide method bodies**

viii. During inheritance, the constructor of the superclass can be overridden by the subclass.
**False because a constructor can never be overridden since it has no implementation body**

ix. During inheritance, a Java class can have more than one superclass.
**False because a class can inherit properties of one class**

x. During method overriding, a method in the superclass declared as protected can be overriden as pubic by the subclass.
**True because we can increase visibility of the inherited method in object oriented programming**

b) Using any example of your choice, write down a set of Java programs to demonstrate polymorphism as an object-oriented programming principle. Your answer should have at least two Java programs in order to accomplish this task. (10 Marks)

```
Interface Me{

public abstract int x(int m, int y);

public abstract int M(int w, int V);

}


public class You{

public static String me;

public Static int Q;


public You(String tme, int tQ){

        tme=me;
```

```
        tQ= Q;
        }
}

public class We extends You implements Me{
public static int age;

public int x(int m, int y){
        return (m*y);
}
Public int M(int w, int V){
        Return (w-V);
}
public We(String tage){
        super(me,Q);
        tage=age;

        }

public static void main(String [ ] args){
We.x(6,2);
We.M(9,3);
System.out.println(We.x(6,2));
        }
}
```

**Question Five**

a) Write down two circumstances under which a class can be abstract. (2 Marks)

**A class can be abstract if it is defined with a keyword abstract.**

**A class can be abstract if it contains at least one of its methods abstract.**

b) Define Booking as an abstract class with an abstract method printBooking() that takes a string and returns nothing. (4 Marks)

```
abstract  class Booking{

        public abstract void printBooking( String Message);

}
```

c) Define Business as a Java interface with a constant called Tax with a value of 25(the total amount levied on the business' sales). This interface should also contain a method netIncome() that takes two integers and returns nothing.
(5 marks)

```
Interface Business{

public static final tax=25;

public abstract void netIncome(int x, int y);

}
```

d) Define a class Shop that implements interface Business. The class Shop should have two instance variables Quantity (integer) and Price (integer) whose values are captured from the user at runtime and passed to netIncome(). The method netIncome() should calculate and print on the screen the net amount obtained from the sales after taxation.
(9 Marks)

```
import java. Util. Scanner;

public class Shop implements Business{

        public int quantity;

        public  int price;

        public static void netIncome(int quantity, int price){

         int amount= ((quantity*price)- tax);
```

```
        System.out.println(" The net amount obtained from sales is" + amount);
                                                                    }
public static void main (String [ ] args){

Scanner m =new Scanner(System.in);
System.out.println("enter the quantity of products sold");
quantity=m.nextInt();
System.out.println("enter the price of the products sold");
price= m.nextInt();
 shop.netIncome(quantity, price);

  }
        }
```

~~~~ "Success" only comes before "Work" in English Dictionary ~~~~
~~~~