**WIA1002 Tutorial 1**

Write the definition of a class *Telephone* that contains:

• An instance variable *areaCode*

• An instance variable *number*

• A static variable *numberOfTelephoneObject* that keeps track of the number of *Telephone* objects created

• A constructor that accepts two arguments used to initialize the two instance variables

• The accessor and mutator methods for *areaCode* and *number*

• A method *makeFullNumber* that does not accept any argument, concatenates areaCode and number with a dash in between, and returns the resultant *String*.

Write the statements to:

* • Instantiate 5 *Telephone* objects and store them in an array. Iterate through the array to print the full number of the 5 *Telephone* objects on the console. Your output should look as below:

03-79676300

03-79676301

03-79676302

03-79676303

03-79676304

public class Telephone {  
 private String areaCode**;** private String number**;** private static int *numberOfTelephoneObject*=**0;** Telephone(String areaCode**,**String number){  
 this.areaCode=areaCode**;** this.number=number**;** *numberOfTelephoneObject*++**;** }  
 public String getAreaCode(){  
 return this.areaCode**;** }  
 public void setAreaCode(String areaCode){  
 areaCode=this.areaCode**;** }  
 public String getNumber(){  
 return this.number**;** }  
 public void setNumber(String number){  
 number=this.number**;** }  
 public String makeFullNumber(){  
 return this.areaCode+"-"+this.number**;** }  
 public static int getNumberOfTelephoneObject() {  
 return *numberOfTelephoneObject***;** }  
}

public class Main {  
 public static void main(String[] args) {  
 Telephone[] telephones = new Telephone[**5**]**;** for (int i = **0;** i < **5;** i++) {  
  
 telephones[i] = new Telephone("03"**,** "7967630" + i)**;** System.*out*.println(telephones[i].makeFullNumber())**;** }  
 }  
}

What is the output for the following? Explain.

|  |
| --- |
| class Person {  public Person() {  System.out.println("(1) Performs Person's tasks"); }  }  class Employee extends Person {  public Employee() {  this("(2) Invoke Employee's overloaded constructor");  System.out.println("(3) Performs Employee's tasks ");  }  public Employee(String s) {  System.out.println(s);  } }  public class Faculty extends Employee {  public Faculty() {  System.out.println("(4) Performs Faculty's tasks");  }  public static void main(String[] args) {  new Faculty(); }  } |

OUTPUT:

(1) Performs Person's tasks

(2) Invoke Employee's overloaded constructor

(3) Performs Employee's tasks

(4) Performs Faculty's tasks

When Faculty is created, constructor of Employee is called.

Constructor of Employee which take parameter String is called with argument ("(2) Invoke Employee's overloaded constructor” , So (2) Invoke Employee’s overloaded constructor is printed.

In Employee class, the super() method is called implicitly, which invoke the constructor of superclass Person. So, “(1) Performs Person’s tasks” is printed.

After executing constructor of Person, remaining statement in constructor of Employee class are execute. So, “(3) Performs Employee’s tasks” is printed.

Once constructor of Employee class is completed, constructor of Faculty class resume execution. So “(4) Performs Faculty’s task” is printed.

What is the output for the following? Explain.

public class C {

public static void main(String[] args) {

Object[] o = {new A(), new B()};

System.out.print(o[0]);

System.out.print(o[1]);

}

}

class A extends B {

public String toString() {

return "A";

}

}

class B {

public String toString() {

return "B";

}

}

OUTPUT: AB

In main, array of Object is created and initialised with 2 object of classes A and B. when System.out.print(o[0]); is executed, it invoke toString() of first object(A). so class A will return A. (DO NOT Execute B bcuz this is not a constructor but method, method do not have super)

When System.out.print(o[1]); is executed, it invoke toString() of second object in array, class B. so return B. Output is the concatenation of 2 printed string.

4.

Write a class definition for an abstract class, *Vehicle*, that contains:

• a double instance variable, *maxSpeed*

• a protected double instance variable, *currentSpeed*

• a constructor accepting a double used to initialize the *maxSpeed* instance variable

• an abstract method, accelerate, that accepts no parameters and returns nothing.

• a method *getCurrentSpeed* that returns the value of *currentSpeed*

• a method *getMaxSpeed* that returns the value of *maxSpeed*

• a method *pedalToTheMetal*, that repeatedly calls accelerate until the speed of the vehicle is equal to *maxSpeed*. *pedalToTheMetal* returns nothing.

Can you create an instance of *Vehicle?*

public abstract class Vehicle {  
 private double maxSpeed**;** protected double currentSpeed**;** Vehicle(double maxSpeed){  
 this.maxSpeed=maxSpeed**;** }  
 public abstract void accelerate()**;** public double getCurrentSpeed(){  
 return this.currentSpeed**;** }  
 public double getMaxSpeed(){  
 return this.maxSpeed**;** }  
 public void pedalToTheMetal(){  
 while(currentSpeed<maxSpeed){  
 accelerate()**;** }  
 }  
}

instance of Vehicle cannot be created since it is an abstract class.

Question 5

Assume the existence of an interface, *Account*, with the following methods:

• *deposit*: accepts an integer parameter and returns an integer

• *withdraw*: accepts an integer parameter and return a Boolean

Define a class, *BankAccount*, that implements the above interface and has the following members:

* • an instance variable named *balance*
* • a constructor that accepts an integer that is used to initialize the instance variable
* • an implementation of the *deposit* method that adds its parameter to the *balance* variable. The new balance is returned as the value of the method.
* • an implementation of the *withdraw* method that checks whether its parameter is less than or equal to the *balance* and if so, decreases the *balance* by the value of the parameter and returns *true*; otherwise, it leaves the *balance* unchanged and returns *false*.

public class BankAccount implements Account {  
 private int balance**;** BankAccount(int balance){  
 this.balance=balance**;** }

@Overide  
 public int deposit(int deposit){  
 this.balance+=deposit**;** return this.balance**;** }

@Overide  
 public boolean withdraw(int money){  
 if(money<=balance){  
 balance-=money**;** return true**;** }  
 else{  
 return false**;** }  
 }  
}