**Question 1**

|  |  |
| --- | --- |
| **1.** | **class msgClass{** |
| **2.** | **public int content;** |
| **3.** | **}** |
| **4.** |  |
| **5.** | **public class Quiz3{** |
| **6.** | **private int sum;** |
| **7.** | **private int y;** |
| **8.** | **public static int x;** |
| **9.** | **public Quiz3(){** |
| **10.** | **sum = 5;** |
| **11.** | **x = 2;** |
| **12.** | **y = 2;** |
| **13.** | **}** |
| **14.** | **public Quiz3(int k){** |
| **15.** | **sum = sum + k;** |
| **16.** | **y = 3;** |
| **17.** | **x += 2;** |
| **18.** | **}** |
| **19.** | **public void methodA(){** |
| **20.** | **int x=1, y=1;** |
| **21.** | **msgClass [] msg = new msgClass[1];** |
| **22.** | **msgClass myMsg = new msgClass();** |
| **23.** | **myMsg.content = Quiz3.x;** |
| **24.** | **msg[0] = myMsg;** |
| **25.** | **msg[0].content = this.y + myMsg.content;** |
| **26.** | **this.y = this.y + methodB(msg[0]);** |
| **27.** | **y = methodB(msg[0]) + this.y;** |
| **28.** | **x = y + methodB(msg, msg[0]);** |
| **29.** | **sum = x + y + msg[0].content;** |
| **30.** | **System.out.println(x + " " + y+ " " + sum);** |
| **31.** | **}** |
| **32.** | **private int methodB(msgClass [] mg2, msgClass mg1){** |
| **33.** | **int x = 2;** |
| **34.** | **y = y + mg2[0].content;** |
| **35.** | **mg2[0].content = y + mg1.content;** |
| **36.** | **x = x + 2 + mg1.content;** |
| **37.** | **sum = sum + x + y;** |
| **38.** | **mg1.content = sum - mg2[0].content ;** |
| **39.** | **System.out.println(Quiz3.x + " " + this.y+ " " + sum);** |
| **40.** | **return sum;** |
| **41.** | **}** |
| **42.** | **public int methodB(msgClass mg1){** |
| **43.** | **int x = 1, y = 2;** |
| **44.** | **y = sum + mg1.content;** |
| **45.** | **this.y = y + mg1.content;** |
| **46.** | **x = Quiz3.x + 5 + mg1.content;** |
| **47.** | **sum = sum + x + y;** |
| **48.** | **Quiz3.x = mg1.content + x + 3;** |
| **49.** | **System.out.println(x + " " + y+ " " + sum);** |
| **50.** | **return y;** |
| **51.** | **}** |
| **52.** | **}** |

**Consider the following code:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Quiz3 a1 = new Quiz3();**  **Quiz3 a2 = new Quiz3(5);**  **msgClass msg = new msgClass();**  **a1.methodA();**  **a2.methodB(msg);** | **x** | **y** | **sum** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Question 2**

|  |
| --- |
| **class A{** |
| **public static int temp = 5;** |
| **public int sum;** |
| **public int y;** |
| **public A(){** |
| **y = temp - 3;** |
| **sum = temp + 2;** |
| **temp-=2;** |
| **}** |
| **public void methodA(int m, int n){** |
| **int x = 1;** |
| **y = y + m + (temp++);** |
| **x = x + 1 + n;** |
| **sum = sum + x + y;** |
| **System.out.println(x + " " + y+ " " + sum);** |
| **}** |
| **}** |
| **class B{** |
| **public static int x;** |
| **public int y = 4;** |
| **public int temp = -5;** |
| **public int sum = 2;** |
| **public B(){** |
| **y = temp + 3 ;** |
| **sum = 3 + temp + 3;** |
| **temp-=2;** |
| **}** |
| **public B(B b){** |
| **sum = b.sum;** |
| **x = b.x;** |
| **b.methodB(1,3);** |
| **}** |
| **public void methodA(int m, int n){** |
| **int x = 2;** |
| **y = y + m + (temp++);** |
| **x = x + 7 + n;** |
| **sum = sum + x + y;** |
| **System.out.println(x + " " + y+ " " + sum);** |
| **}** |
| **public void methodB(int m, int n){** |
| **int y = 0;** |
| **y = y + this.y;** |
| **x = this.y + 3 + temp;** |
| **methodA(x, y);** |
| **sum = x + y + sum;** |
| **System.out.println(x + " " + y+ " " + sum);** |
| **}** |
| **}** |

**Consider the following code:**

|  |  |  |  |
| --- | --- | --- | --- |
| **A a1 = new A();**  **B b1 = new B();**  **B b2 = new B(b1);**  **b1.methodA(3, 2);**  **b2.methodB(1, 2);** | **x** | **y** | **sum** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Question 3**

Given the following classes, write the code for the **BBAStudent**class so that the following output is printed when we run the **TestStudent**class.

**Name : Default BBA Student**

**Name : Barack Obama**

**Name : Joe Biden**

public abstract class Student{

private String name;

public final String getName(){

return name;

}

public final void setName(String name){

this.name = name;

}

}

public class TestStudent{

public static void printName(Student s){

System.out.println("Name : "+s.getName());

}

public static void main(String [] args){

printName(new BBAStudent());

printName(new BBAStudent("Barack Obama"));

printName(new BBAStudent("Joe Biden"));

}

}

**Question 4**

Write the **FootBallTeam and the CricketTeam** classes so that the following code generates the output below:

**public class Sports{**

**public static void printDetail(Team t){**

**System.out.println("----Printing Detail-----");**

**System.out.println(t);**

**t.getMascot();**

**}**

**public static void main(String [] args){**

**FootBallTeam f = new FootBallTeam("Brazil","Ace Purple");**

**CricketTeam c = new CricketTeam("Australia","Golden Griffin");**

**printDetail(f);**

**printDetail(c);**

**}**

**}**

**class Team{**

**protected String name = null;**

**protected String mascot = "Default mascot";**

**Team(String name, String mascot){**

**this.name = name;**

**this.mascot = mascot;**

**}**

**public void getMascot(){**

**System.out.println(mascot);**

**}**

**}**

**----Printing Detail-----**

**Our name is Brazil**

**We play Football**

**Our football team mascot name is Ace Purple**

**----Printing Detail-----**

**Our name is Australia**

**We play Cricket**

**Our cricket team mascot name is Golden Griffin**

**Question 5**

|  |
| --- |
| **public class Frodo extends Bilbo** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Frodo 1");** |
| **super.method1();** |
| **}** |
|  |
| **public void method3()** |
| **{** |
| **System.out.println("Frodo 3");** |
| **}** |
| **}** |
| **public class Gandalf** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Gandalf 1");** |
| **}** |
|  |
| **public void method2()** |
| **{** |
| **System.out.println("Gandalf 2");** |
| **method1();** |
| **}** |
| **}** |
| **public class Bilbo extends Gandalf** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Bilbo 1");** |
| **}** |
| **}** |
| **public class Gollum extends Gandalf** |
| **{** |
| **public void method3()** |
| **{** |
| **System.out.println("Gollum 3");** |
| **}** |
| **}** |

**And assuming the following variables have been defined:**

**Gandalf var1 = new Frodo();**

**Gandalf var2 = new Bilbo();**

**Gandalf var3 = new Gandalf();**

**Object var4 = new Bilbo();**

**Bilbo var5 = new Frodo();**

**Object var6 = new Gollum();**

In the table below, indicate in the right-hand column the output produced by

the statement in the left-hand column. If the statement produces more than one

line of output, indicate the line breaks with slashes as in "a/b/c" to indicate

three lines of output with "a" followed by "b" followed by "c". If the

statement causes an error, fill in the right-hand column with either the phrase

"compiler error" or "runtime error" to indicate when the error would be

detected.

**Statement Output**

|  |  |
| --- | --- |
| **var1.method1();** |  |
| **var2.method1();** |  |
| **var4.method1();** |  |
| **var6.method1();** |  |
| **var1.method2();** |  |
| **var3.method2();** |  |
| **var4.method2();** |  |
| **var5.method2();** |  |
| **var6.method2();** |  |
| **((Frodo)var4).method3();** |  |
| **((Frodo)var6).method2();** |  |
| **((Gollum)var1).method3();** |  |
| **((Gollum)var4).method1();** |  |
| **((Gandalf)var1).method2();** |  |
| **((Frodo)var4).method1();** |  |
| **((Gollum)var6).method2();** |  |
| **((Gandalf)var2).method1();** |  |
| **((Bilbo)var6).method2();** |  |
| **((Frodo)var1).method3();** |  |
| **((Gandalf)var5).method3();** |  |

**Task 6**

Write the **CheckingAccount**class so that the following code generates the output below:

**public abstract class Account{**

**private static long accountNumber = 100;**

**private String name;**

**private long accNumber;**

**private double balance;**

**public Account(String name){**

**this.name = name;**

**accNumber = accountNumber++;**

**balance = 0.0;**

**}**

**protected String getName(){**

**return name;**

**}**

**protected double getBalance(){**

**return balance;**

**}**

**protected void setBalance(double bal){**

**if (bal > 0){**

**balance = bal;**

**}**

**}**

**protected long getAccountNumber(){**

**return accNumber;**

**}**

**public abstract void printAccountInfo();**

**}**

**public class AccountTester{**

**public static void printInfo(Account a){**

**a.printAccountInfo();**

**}**

**public static void main(String [] args){**

**CheckingAccount c1 = new CheckingAccount("Nadeem", 400);**

**CheckingAccount c2 = new CheckingAccount("Susan", 600);**

**CheckingAccount c3 = new CheckingAccount("Raqib", 1000);**

**printInfo(c1);**

**printInfo(c2);**

**printInfo(c3);**

**System.out.println(c1.deposit(-100.00));**

**System.out.println(c1.deposit(0.00));**

**System.out.println(c1.deposit(200.00));**

**System.out.println(c1.withdraw(1000.00));**

**System.out.println(c1.withdraw(0.00));**

**System.out.println(c1.withdraw(300.00));**

**printInfo(c1);**

**}**

**}**

**Output of AccountTester:**

**=======================================**

**Name : Nadeem**

**Account Number : 100**

**Balance : 400.0**

**Your balance is less than the minimum amount**

**=======================================**

**=======================================**

**Name : Susan**

**Account Number : 101**

**Balance : 600.0**

**=======================================**

**=======================================**

**Name : Raqib**

**Account Number : 102**

**Balance : 1000.0**

**=======================================**

**Please enter amount greater than Zero**

**Please enter amount greater than Zero**

**Deposit Successful**

**You do not have enough funds**

**Please enter amount greater than Zero**

**Withdraw Successful**

**=======================================**

**Name : Nadeem**

**Account Number : 100**

**Balance : 300.0**

**Your balance is less than the minimum amount**

**=======================================**

**Task 7**

Write the **Cat** and **Dog** class so that the following code generates the output below

|  |
| --- |
| public abstract class Animal { |
| protected String sound = "Animal Sound"; |
| public Animal(){ |
| } |
| Animal(String \_sound){ |
| this.sound = \_sound; |
| } |
| public abstract String makeSound(); |
| } |
|  |
| public class AnimaDriver{ |
| public static void printSound(Animal a){ |
| System.out.println(a.makeSound()); |
| } |
| public static void main(String [] args){ |
| Dog d1 = new Dog("bark"); |
| Cat c1 = new Cat("meow"); |
| printSound(c1); |
| printSound(d1); |
| } |
| } |

**meow**

**bark**

**Task 8**

Given the following classes, write the code for the **Dog** class so that the following output is created [Dog extends animal].

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rover is crying.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**abstract public class Animal {**

**//Name of the Animal**

**private String name;**

**//Constructor**

**Animal(String \_name){**

**this.name = \_name;**

**}**

**//Return name**

**protected String getName(){**

**return name;**

**}**

**}**

**//Driver class**

**public class AnimalTorture {**

**private void hugAnimal(SoundSource a){**

**a.makeSound();**

**}**

**public static void main(String[] args) {**

**Dog d = new Dog("Rover");**

**AnimalTorture at = new AnimalTorture();**

**at.hugAnimal(d);**

**}**

**}**

**//Interface for making sound.**

**public interface SoundSource {**

**public void makeSound();**

**}**

**Task 9**

**Refactoring:**

Given the three existing classes: Point, Circle,and Rectangle, establish a properly designed inheritance hierarchy with an abstract class named **Shape**. On the following page, not inherited version of the classes are already given. You do not have to rewrite the Point class.

*Implement the hierarchy here: Shape, Circle, and Rectangle (no need to modify the Point class)*

|  |
| --- |
| Required Output |
| 0) 3.141592653589793 at 5, 10 |
| 1) 18.6875 at 40, 10 |
| 2) 12.566370614359172 at 15, 30 |

publicclass ShapesTwo{

publicstaticvoid main(String[] args){

int n = 3;

Shape[] shapes = new Shape[n];

shapes[0] = new Circle(5, 10, 2.0);

shapes[1] = new Rectangle(40, 10, 3.25, 5.75);

shapes[2] = new Circle(15, 30, 4.0);

for (int i = 0; i < n; i++){

System.out.println(i + ") " + shapes[i].getArea() + " at " +shapes[i].getX() + ", " +

shapes[i].getY());

}

}

}

|  |
| --- |
| **public class Point {**  **private int my\_xPos;**  **private int my\_yPos;**  **public Point(int x, int y) {**  **my\_xPos = x;**  **my\_yPos = y;**  **}**  **public int getX() {**  **return my\_xPos;**  **}**  **public int getY() {**  **return my\_yPos;**  **}**  **}**  **public class Rectangle{**  **private Point my\_upperLeft;**  **private double my\_width;**  **private double my\_height;**  **public Rectangle(int x, int y, double height, double width){**  **my\_upperLeft = new Point(x, y);**  **my\_width = width;**  **my\_height = height;**  **}**  **public int getX() {**  **return my\_upperLeft.getX();**  **}**  **public int getY() {**  **return my\_upperLeft.getY();**  **}**  **public double getArea() {**  **return my\_width \* my\_height;**  **}**  **}**  **public class Circle{**  **private Point my\_upperLeftCorner;**  **private double my\_radius;**  **public Circle(int x, int y, double diameter) {**  **my\_upperLeftCorner= new Point(x,y);**  **my\_radius = diameter / 2;**  **}**  **public int getX() {**  **return my\_upperLeftCorner.getX();**  **}**  **public int getY() {**  **return my\_upperLeftCorner.getY();**  **}**  **public double getArea(){**  **return Math.PI\*Math.pow(my\_radius,2);**  **}**  **}** |