



PROGRAMMING IN JAVA

Assignment 3

TYPE OF QUESTION: Online Programming

Number of questions: 5

Total mark: $5 \times 2 = 10$

QUESTION 11:

Problem statement:

This program is related to the generation of Fibonacci numbers.

For example: 0,1, 1,2, 3,5, 8, 13,... is a Fibonacci sequence where 13 is the 8th Fibonacci number.

A partial code is given and you have to complete the code as per the instruction given below.

Public test case:

Input: 8

Output:

13

Private test case:

Input: 1

Output:

0

Private test case:

Input: 2

Output:

1

Input: 3

Output:

1



Prefixed Fixed Code:

```
import java.util.Scanner; //This package for reading input
public class Fibonacci {

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt(); //Read an integer
        System.out.println(fib(n)); //Generate and print the n-th Fibonacci
                                   //number
    }
    static int fib(int n) {
```

Template code:

```
//complete the code segment to find the nth Fibonacci number in the
Fibonacci sequence and return the value. Write the function recursively.
```

Suffixed Fixed Code:

```
}
}
```

Invisible code: Not applicable

Sample Solution:

```
import java.util.Scanner;
public class Fibonacci {

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        System.out.println(fib(n));
    }
    //Template code:
    static int fib(int n) {
        if (n==1) //Terminal condition
            return 0;
        else if (n==2)
            return 1;
        return fib(n - 1) + fib(n - 2); //Recursive call of function
    }
}
```



QUESTION 12:

Problem statement:

Define a class **Point** with two fields **x** and **y** each of type **double**. Also, define a method **distance(Point p1, Point p2)** to calculate the distance between points **p1** and **p2** and return the value in **double**.

Complete the code segment given below. Use **Math.sqrt()** to calculate the square root.

Public test case:

Input: 2.0 3.0
1.0 2.0

Output:

1.4142135623730951

Private test case:

Input: 2.0 1.0
1.0 1.0

Output:

1.0

Prefixed Fixed Code:

```
import java.util.Scanner;

public class Circle extends Point{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        Point c=new Point(); //Create a point center
        c.x=sc.nextDouble();
        c.y=sc.nextDouble();
        Point p=new Point(); //Create a point on circumference
        p.x=sc.nextDouble();
        p.y=sc.nextDouble();
        Circle cl=new Circle(); //Create an object of class Circle
        cl.distance(p1,p2); //Calcualte radius of the circle

    }

}
```

Template code:

```
//Complete the code segment to define a class Point with parameter x,y and
method distance()for calculating distance between two points.
Note: Pass objectsof type class Point as argument in distance() method.
```



Suffixed Fixed Code:

Not applicable

Invisible code:Not applicable

Sample Solution:

```
import java.util.Scanner;

public class Circle extends Point{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        Point p1=new Point();
        p1.x=sc.nextDouble();
        p1.y=sc.nextDouble();
        Point p2=new Point();
        p2.x=sc.nextDouble();
        p2.y=sc.nextDouble();
        Circle c1=new Circle();
        c1.distance(p1,p2);

    }

}

//Template code:
class Point{
    double x;
    double y;

    public static void distance(Point p1,Point p2){
        double d;
        d=Math.sqrt((p2.x-p1.x)*(p2.x-p1.x) + (p2.y-p1.y)*(p2.y-p1.y));
        System.out.println(d);
    }

}
```

QUESTION 13:

A class Shape is defined with two overloading constructors in it. Another class Test1 is partially defined which inherits the class Shape. **The class Test1 should include two overloading constructors as appropriate for some object instantiation shown in main() method.** You should define the constructors using the super class constructors. Also, override the method calculate() in Test1 to calculate the volume of a Shape.



Test case:

Input: 2.0 3.0 4.0

Output:

16.0
24.0

Private test case:

Input: 2.0 1.0 1.0

Output:

4.0
2.0

Prefixed Fixed Code:

```
import java.util.Scanner;
class Shape{
    double length, breadth;
    Shape(double l, double b){ //Constructor to initialize a Shape object
        length = l;
        breadth= b;
    }
    Shape(double len){ //Constructor to initialize another Shape object
        length = breadth = len;
    }
    double calculate(){// To calculate the area of a shape object
        return length * breadth ;
    }
}
public class Test1 extends Shape{
```

Template code:

```
//Create a derived class constructor which can call the one parametrized
constructor of the base class
//Create a derived class constructor which can call the two parametrized
constructor of the base class
//Override the method calculate() in the derived class to find the volume
of a shape instead of finding the area of a shape
```



Suffixed Fixed Code:

```
public static void main(String args[]){
    Scanner sc = new Scanner(System.in); //Create an object to read
                                           //input
    double l=sc.nextDouble(); //Read length
    double b=sc.nextDouble(); //Read breadth
    double h=sc.nextDouble(); //Read height
    Test1 myshape1 = new Test1(l,h);
    Test1 myshape2 = new Test1(l,b,h);
    double volume1;
    double volume2;
    volume1 = myshape1.calculate();
    volume2=myshape2.calculate();
    System.out.println(volume1);
    System.out.println(volume2);
}
```

Invisible code:Not applicable

Sample Solution:



```
import java.util.Scanner;

class Shape {
double length, breadth;
Shape(double l, double b) {
    length = l;
    breadth = b;
}
Shape(double len) {
    length = breadth = len;
}
double calculate() { // area of shape
    return length * breadth;
}
}

public class Test1 extends Shape {
//Template code:
    double height;
    Test1(double length, double h) {
//base class constructor with one parameter is called
        super(length);
        height = h;
    }

    Test1(double length, double breadth, double h) {
//base class constructor having two argument is called
        super(length, breadth);
        height = h;
    }

    double calculate() {
        return length * breadth * height;
    }

//Suffixed fixed code
public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    double l = sc.nextDouble();
    double b = sc.nextDouble();
    double h = sc.nextDouble();
    Test1 myshape1 = new Test1(l, h);
    Test1 myshape2 = new Test1(l, b, h);
    double volume1;
    double volume2;
    volume1 = myshape1.calculate();
    volume2 = myshape2.calculate();
    System.out.println(volume1);
    System.out.println(volume2);
}
}
```



QUESTION 14:

This program to exercise the call of static and non-static methods. A partial code is given defining two methods, namely `sum()` and `multiply ()`. You have to call these methods to find the sum and product of two numbers. Complete the code segment as instructed.

Public test case:

Input: 3 5

Output:

8
15

Private test case:

Input: 1 2

Output:

3
2

Prefixed Fixed Code:

```
import java.util.Scanner;
class QuestionScope {
    int sum(int a, int b){ //non-static method
        return a + b;
    }
    static int multiply(int a, int b){ //static method
        return a * b;
    }
}
public class Test3{
    public static void main( String[] args ) {
        Scanner sc = new Scanner(System.in);
        int n1=sc.nextInt();
        int n2=sc.nextInt();
```

Template code:

```
//Called the method sum() to find the sum of two numbers.
//Called the method multiply() to find the product of two numbers.
```

Suffixed Fixed Code:

```
    }
}
```




Invisible code:Not applicable

Sample Solution:

```
import java.util.Scanner;
class QuestionScope {
    int sum(int a, int b){ //non-static method
        return a + b;
    }
    static int multiply(int a, int b){ //static method
        return a * b;
    }
}
public class Test3{
    public static void main( String[] args ) {
        Scanner sc = new Scanner(System.in);
        int n1=sc.nextInt();
        int n2=sc.nextInt();

        QuestionScope st = new QuestionScope(); // Create an object to call non-
                                                //static method
        int result1=st.sum(n1,n2); // Call the method
        int result2=QuestionScope.multiply(n1,n2); // Create an object to call
                                                //static method

        System.out.println(result1);
        System.out.println(result2);
    }
}
```

QUESTION 15:

Problem Statement

Complete the code segment to swap two numbers using call by object reference.

Public test case:

Input: 10 20

Output:

20 10

Private test case:

Input: 11 29



Output:

29 11

Prefixed Fixed Code:

```
import java.util.Scanner;
class Question { //Define a class Question with two elements e1 and e2.
    Scanner sc = new Scanner(System.in);
    int e1 = sc.nextInt(); //Read e1
    int e2 = sc.nextInt(); //Read e2
}
public class Question3 {
```

Template code:

```
// Define static method swap() to swap the values of e1 and e2 of class
Question.
```

Suffixed Fixed Code:

```
public static void main(String[] args) {
    //Create an object of class Question
    Question t = new Question ();
    //Call the method swap()
    swap(t);

    System.out.println(Question.e1+" "+Question.e2);
}
}
```

Invisible code: Not applicable



Sample Solution:

```
//Prefixed Fixed code
import java.util.Scanner;
class Question { //Define a class Question with two elements e1 and e2.
    Scanner sc = new Scanner(System.in);
    int e1 = sc.nextInt(); //Read e1
    int e2 = sc.nextInt(); //Read e2
}
public class Question3 {

//Template code
//Define static method swap() to swap the values of e1 and e2 of class T.
public static void swap(Question t) {
    int temp = t.e1;
    t.e1 = t.e2;
    t.e2 = temp;
}

public static void main(String[] args) {
//Create an object of class Question
    Question t = new Question ();
    //Call the method swap()
    swap(t);

    System.out.println(t.e1+" "+t.e2);
}
}
```

*****END*****