



United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final-term Exam :: Fall 2020

Course Code: CSE 1115 Course Title: Object Oriented Programming

Total Marks: 25 Time: 1hr 15 min

READ THIS CAREFULLY: Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules

Question 1 (3+2)

A) Consider the following code:

[3]

<pre>public class Demo{ public static void moveAround(Moveable m){ m.run(Moveable.FORWARD); } public static void communicate(Speaker s){ s.speak(); } public static void main(String[] args){ Robot r = new Robot(); moveAround(r); communicate(r); } }</pre>	<pre>interface Moveable{ int FORWARD = 0, BACKWARD = 1; void run(int direction); void walk(); } interface Speaker{ void speak(); }</pre> <p>OUTPUT: Running in direction 0 speaking</p>
---	---

An object of the **Robot** class is passed to the **communicate** method and also to the **moveAround** method. Your task is to write the **Robot** class in such a way that it produces the given output.

B) Consider the following code:

[2]

```
interface Counter{
    int increment();
}
interface Clock extends Counter{
    void showTime();
}
public class NC{
    public static void main(String[] args){
        Clock c = new Clock// complete this statement
    }
}
```

Usually, we cannot create an object of an interface. But that restriction can be bypassed by following a procedure called...well you should know that by now. Complete the given code to make it work. You cannot change any of the given code. Write only the codes which are not given in the question.

Question 2 (3 + 2)

A) Consider the following code:

[3]

```
public static void assign(int id, int age){
    validate(age);
    System.out.println("assigned " + id + " to work");
}
```

- (i) Write a method **validate()** that takes as input an integer age and **throws** an exception if age is less than 18.
- (ii) Now, modify the following **assign()** method to handle the exception from your validate method. (you cannot delete any line)

B) Consider the following code:

[2]

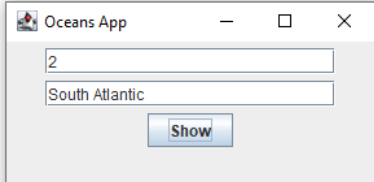
<pre>class Test{ private int a; int b; static int c; class Inner1{ public void m(){ System.out.println(a); // 1 System.out.println(b); // 2 System.out.println(c); // 3 } } }</pre>	<pre>static class Inner2{ public void m(){ System.out.println(a); // 4 System.out.println(b); // 5 System.out.println(c); // 6 } }</pre>
--	--

The given code contains 2 inner classes, **Inner1** and **Inner2**. Both inner class contains method **m()**. Now, not all statements inside the **m()** methods are correct. You need to write the **incorrect** statements inside the **m()** methods in the given code.

Question 3 (5 + 10)

A) Consider the following code:

[5]

<pre>class GUITest{ public static void main(String[] args) { JFrame f = new JFrame("Oceans App"); f.setSize(300, 150); f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); f.setLayout(new FlowLayout()); String [] oceans = {"Arctic", "North Atlantic", "South Atlantic", "Indian", "North Pacific", "South Pacific", "Antarctic"}; JTextField tf1 = new JTextField(20); JTextField tf2 = new JTextField(20); // Your code here f.setVisible(true); } }</pre>	
---	--

The code contains a String array named **oceans**. You should complete the code so that a GUI is generated as shown in the image. The GUI contains 2 JTextFields, **tf1** and **tf2** and a JButton **Show**. The user inputs an **index** in the range 0-6 in **tf1** and clicks the **Show** button. Then the ocean name in the given **index** of the **oceans** array is shown in **tf2**. Implement the mechanism of the **Show** button also. **Note:** Only write the codes that are not given in the question.

B) Two managers of the CC Bank approached you for some programming tasks. They have a **file**, containing the data (name, account id, and balance) of about 50,000 accounts. The first line of their file looks like this: [10]

Abhishek Ac10298 1000000

The rest of the lines of the file contains a similar structure with different values for name, account id, and balance. Using this file, the managers have the following requirements:

- The first manager's requirement is this: He needs a **file** containing the account data of only the accounts which's **balance** is **greater than 10000**. Your first task is to prepare a file which meets the requirement of the first manager.
- The second manager's requirement is: He needs a **file** containing the account data of all the accounts **sorted in descending** order according to **balance**. Your second task is to prepare a file which meets the requirement of the second manager.

Note: You can use a single code for the given 2 tasks.