United International University
Department of Computer Science and Engineering

CSI 211/1115: Object Oriented Programming

Final Examination, Summer 19 Time: 2 Hours

Part A. Answer all questions from Part A

1. (a) Write the output of the following program for the given inputs. Please bear in mind that missing import statements should not be regarded as errors for this question. [3]

```
class ErrorAnywhere {
    public static void main(String args[])
    {
        int a, b, c = 4;
        Scanner sc = new Scanner(System.in);
        a = sc.nextInt();
        b = sc.nextInt();

        try{
            a = a/a;
            c = c + 1;
            b = b / b;
            c = c + 1;
        }
        catch(ArithmeticException f){
            System.out.println(c);
        }finally{
            System.out.println("END");
        }
    }
}

Inputs
a = 0, b = 0
a = 1, b = 0
a = 1, b = 1
```

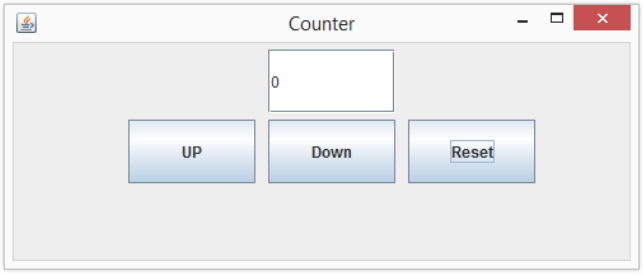
(b) Suppose, you are creating a java program for a showroom that displays four brands of bike :- [5]

i. FZS
ii. R15_V3
iii. Fazer
iv. R15_V2

Initially the shop will start with 100 pieces of each of the brands, and those numbers will reduce as the bikes are sold one by one. Create a class named Inventory containing variables representing number of pieces available for each of the brands, and individual methods for decreasing those numbers as sales are processed. Inventory class is going to require a custom exception that says "**Arsenal running dry**" whenever $FZS < 30$ or $R15\_V3 < 40$ or $Fazer < 34$ or $R15\_V2 < 50$ and individual supplies will be replenished (get back to 100) subsequently.

2. You are required to complete a Java GUI application like below which will implement a 3-bit Up and Down counter [minimum value: 0 and maximum value: 7]. Initially the value will be set as 0.The Up [8]

button will perform the operation of up counter and the Down button will perform the operation of down counter. The Reset button will set the value of the counter as 0.



The template code is given below:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UpDownCounter extends JFrame {
    private JTextField both;
    private JButton upCounter , downCounter , reset;
    private Container c;

    public UpDownCounter(){
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Counter");
        setBounds(0,0,500,250);
        c = getContentPane();
        c.setLayout(null);

        both = new JTextField("0");
        both.setBounds(200,5,100,50);
        c.add(both);

        upCounter = new JButton("UP");
        upCounter.setBounds(90,60,100,50);
        c.add(upCounter);

        downCounter = new JButton("Down");
        downCounter.setBounds(200,60,100,50);
        c.add(downCounter);

        reset = new JButton("Reset");
        reset.setBounds(310,60,100,50);
        c.add(reset);

        // add necessary code here.

    }
    class actionHandler implements ActionListener {
        public void actionPerformed(ActionEvent e){
```

```
            // add necessary code here.
        }
    }
    public static void main(String[] args) {
        // add necessary code here.
    }
}
```

3. Suppose, four friends Dustin, Mike, Lucas and Eleven are on a mission of rescuing their fifth friend Will from the monster named, Demogorgon and its fellow monster gang.

```
Class Friend implements Runnable{
    // Write your code here
}

Class RescueWill{
    public static void main(String[] args){
          Friend friend1= new Friend("Dustin");
          Friend friend2= new Friend("Mike");
          Friend friend3= new Friend("Lucas");
          System.out.println("Eleven saves them all");
    }
}
```

```
Expected output:
Dustin comes to rescue Will.
Mike comes to rescue Will.
Lucas comes to rescue Will.
Dustin killed 1 monster.
Mike killed 1 monster.
Lucas killed 1 monster.
Dustin killed 2 monster.
Mike killed 2 monster.
Lucas killed 2 monster.
Dustin is attacked by Demogorgon.
Mike is attacked by Demogorgon.
Lucas is attacked by Demogorgon.
Eleven saves them all.
```

(a) Now complete the "Friend" class above to get the expected output. You cannot modify the class "RescueWill". [5]

(b) Will the output of your code always end with the line "Eleven saves them all" ? If yes, explain the reason. If not, then provide necessary modifications in the code to ensure that the output always ends with the line "Eleven saves them all." (You can modify anywhere in the code including in the class "RescueWill".) [3]

4. Find errors (if any) in the following code snippet and then rewrite the correct code. If there isn't any error, write down the output of the code. [4]

```
public class Book{
     class Cover{
           public void foo(){
                System.out.println("Inside foo");
           }
    }
     void bar(){
            System.out.println("Inside bar");
     }
     public static void main(String args []){
           Cover a= new Cover();
           a.foo();
           a.bar();
    }
}
```

5. You can insert null values in ArrayList and also can set null as a key for HashMap. What will be the output of the following program? Write the output with proper reasoning. [4]

```
import java.util.*;
class Tricky {
   public static void main(String args[])
   {
       ArrayList<String> list = new ArrayList<String>();
       list.add("C");
       list.add(null);
       list.add("A");
       list.add(null);
       System.out.println("ArrayList: " + list);
       HashMap<Integer, String> hm = new HashMap<Integer, String>();
       hm.put(1, "A");
       hm.put(2, "B");
       hm.put(null, "C");
       hm.put(null, null);
       hm.put(3, null);
       System.out.println("HasMap: " + hm);
   }
}
```

Part B. Answer any two questions from Part B

6. Suppose in your system you are using two different Stack classes one as a Stack of Integers and another as a Stack of Strings. Due to the addition of some new system features, you need another Stack of a class named Student. Implementing a new stack every time you need a stack of different data types will increase the systems complexity. So you have decided to write a generic class instead of writing a new class every time. Now write a generic class named GenericStack<T>which will solve this problem. A sample implementation of the previous classes are shown below. Just convert these classes to a single equivalent Generic class. [4]

```java
class StackInt{
    private int max_size=5;
    private int cursize=0;
    Integer stack_arr[]=new Integer[max_size];

    void push(Integer val){
        if(cursize>=max_size){
            System.out.println("Status >>> Stack
                Full");
        }
        else{
            stack_arr[cursize]=val;
            cursize++;
        }
    }

    Integer pop(){
        Integer top_value=0;

        if(cursize==0){
            System.out.println("Status >>> Empty
                Stack");
        }
        else{
            top_value=stack_arr[cursize-1];
            cursize--;
        }

        return top_value;
    }
}
```

```java
class StackString{
    private int max_size=5;
    private int cursize=0;
    String stack_arr[]=new String[max_size];

    void push(String val){
        if(cursize>=max_size){
            System.out.println("Status >>> Stack
                Full");
        }
        else{
            stack_arr[cursize]=val;
            cursize++;
        }
    }

    String pop(){
        String top_value="0";

        if(cursize==0){
            System.out.println("Status >>> Empty
                Stack");
        }
        else{
            top_value=stack_arr[cursize-1];
            cursize--;
        }

        return top_value;
    }
}
```

7. Write a program in java that takes a string "one,two,three,four,five", splits the string by delimeter "," and writes each token in "tokens.txt" file. You can use any of the file writer method to accomplish this task. You are given a code of how to split a string. [4]

```java
String string = "004-034556";
String[] parts = string.split("-");
String part1 = parts[0]; // 004
String part2 = parts[1]; // 034556
```

8. Suppose, you have an arraylist of all the students IDs of your class. Sort the list according to the last digit of ID in descending order. Use a comparator to implement this. For example, the arrayList contains 123, 234, and 342 values, you will get 234, 123, and 342 values after the sort operation. You have to write only the comparator method. You do not need to write any main class or student class. [4]