United International University (UIU)
Dept. of Computer Science & Engineering (CSE)

Final Exam :: Trimester: Summer - 2017

Course Code: **CSI 211**, Course Title: **Object-Oriented Programming**

Total Marks: **40** Duration: **2 Hours**

There are seven questions. **Answer any five questions**. Figures in the right-hand margin indicate full marks

Question 1 [3+5]

The following class has been created to help your course teacher to keep track of Students with low attendance and low test score. **Observe** the code below, the sample run/output and **create** the Exception Classes' mentioned in the code; **InSufficientAttendanceException**, **LowScoreException**, so that the program produces the following outputs.

```
import java.util.Scanner;
public class ScoreTracker {
 public static void main(String[] args) {
  Scanner scan = new Scanner(System.in);
  System.out.print("Enter number of Student:");
  int studentCount = scan.nextInt();
  for(int i =0; i<studentCount; i++){
   try{
       System.out.print("\nEnter attendance of "+i+"th 
Student:");
       int att = scan.nextInt();
       if (att<=4)
         throw new InSufficientAttendanceException(4);

       System.out.print("Enter total test score:");
       double totalScore = scan.nextDouble();
       if(totalScore <55 )
         throw new LowScoreException(55);

       System.out.print("Enter class test score:");
       double ctScore = scan.nextDouble();
       if(ctScore <8 )
         throw new LowScoreException(8, totalScore);
     }
     catch(Exception e){
       System.out.println(e.getMessage());
     }
   } // end of for loop
 }// end of main
}// end of class
```

Sample run/output

```
Enter number of Student:3

Enter attendance of 0th Student:3
Minimum attendance should be 4 days.

Enter attendance of 1th Student:20
Enter total test score:54
Total test score should be 55.0 or
above.

Enter attendance of 2th Student:12
Enter total test score:70
Enter class test score:7
Minimum class test score should be 8.0.
```

Question 2 [8]

Suppose you and your friend "Bondhu" are playing football. Since there are no other players, you are casually passing the ball to Bondhu and he's passing you back. As there is only one football, a player can only pass it when he has it on his foot. Now, write a Java program that simulates the scenario by writing two Java thread classes **Bondhu** and **Player1** where **Football** is a shared item**.** Also write a **Test** class that contains the main method which outputs the following:

```
Player1 passed
Bondhu passed
Player1 passed
Bondhu passed
.....
```

Note that, a player cannot pass if he does not have the football. So a sequence "Bondhu passed, Bondhu passed" is invalid. The game should **end** after each of the players has made **five** passes.

Question 3 [4+4]

a) **Update** the Java program below **to match the expected output**. You **cannot remove** any line of code and **you must not simply print the expected output**.

Code

```
class Work implements Runnable {
  String name;
  Thread t;
  Work(String name) {
    this.name = name;
    t = new Thread(this, name);
    System.out.println("New work: "+ t.getName());
    t.start();
  }
  public void run() {
      Thread.sleep(1000);
  }
}
public class Test {
  public static void main(String[] args) {
      Work w1 = new Work("Eat");
      Work w2 = new Work("Pray");
      Work w3 = new Work("Love");
      System.out.println("Main thread exiting");
} }
```

Expected Output

```
New work: Eat
New work: Pray
Eat: 3
New work: Love
Pray: 3
Love: 3
Love: 2
Pray: 2
Eat: 2
Pray: 1
Love: 1
Love: exiting
Eat: 1
Eat: exiting
Pray: exiting
Main thread exiting
```

b) The following Java program is incomplete and/or contains errors. List the errors (if any) and then rewrite the program to match the expected output. You are **not allowed** to (i) modify the main method and (ii) use method overloading.

```
class Dragon {
      String name;
      double firePower;
      public Dragon(String name, double firePower) {
            this.name = name;
            this.firePower = firePower;
      }}

public class FaultyMaybe {
   public static int minimum(int x, int y, int z) {
      int min = x;
      if (y < min) min = y;
      if (z > min) min = z;
      return min;
   }

   public static void main(String[] args) {
      System.out.println("Min of 3, 4, 5: " + minimum(3,4,5));
      System.out.println("Min of 3.3, 4.3, 1.5: " + minimum(3.3,4.3,1.5));
      System.out.println("Min of Robb, Jon, Bran: " + minimum("Robb","Jon","Bran"));

      Dragon [] drgns = new Dragon[3];
      drgns[0] = new Dragon("Drogon", 2.3);
      drgns[1] = new Dragon("Rhaegal", 1.2);
      drgns[2] = new Dragon("Viserion", 0.3);
     System.out.println("Least powerful dragon: "+minimum(drgns[0],drgns[1],drgns[2]));
   }
}
```

Expected output:

```
Min of 3, 4, 5: 3
Min of 3.3, 4.3, 1.5: 1.5
Min of Robb, Jon, Bran: Bran
Least powerful dragon: Viserion
```

Question 4 [1.5+1.5+5=8]

Suppose there are two Java classes, namely **Circle** and **Quadrilateral.** The **Circle** class contains an attribute **radius** and the **Quadrilateral** class contains attributes **dim1** and **dim2.** Both classes implement an interface **Drawable** that contains a method **void draw()**. They also have common methods **double getArea() and double getPerimeter().** You have decided to introduce a super/parent class for these two classes called **Shape2D** that will have **getArea() and getPerimeter()** methods as abstract methods since they only have meaning in the subclasses.

Suppose there are two more Java classes named **Rectangle** and **Square** that extend **Quadrilateral.** Now answer the following:

i. Decide whether the class **Shape2D** should be declared as an abstract class or an interface. Why so?
ii. Should the class **Quadrilateral** be abstract? Why so?
iii. Write complete Java code implementations of the aforementioned classes and interface(s). You should add parameterized constructors to the classes.

Main Method	**Your code should output the following when this main method is run:**

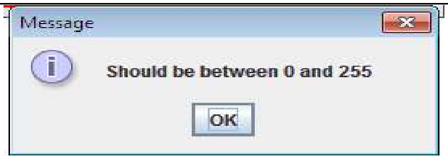
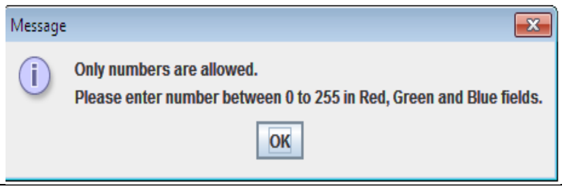
```
public static void main(String[] args) {
  Quadrilateral q1 = new Rectangle(10, 5);
  Quadrilateral q2 = new Square(8);
  Shape2D shape = new Circle(5);

  System.out.println("Area: " + shape.getArea());
  shape = q1;
  System.out.println("Area: " + shape.getArea());
  shape = q2;
  System.out.println("Area: " + shape.getArea());

  System.out.println("Perimeter: "+q2.perimiter());
  Drawable d = q2;
  d.draw();
}
```
	```
	Area: 78.54
	Area: 50.0
	Area: 64.0
	Perimeter: 32.0
	Drawing Square
	```

Question 5 [3+2+3]

Create a simple GUI application as shown below. There are 3 TextFields to take input for the RGB values of a color. **Clicking** the “Change Color” will **change the background color** to the specific color generated by the RGB values entered by the user in those 3 TextFields. If user enters **any value less than 0 or greater than 255**, a pop-up message will display as shown below. Also entering **any input other than integer** value will display another pop-up as shown below.

Any value<0 or >255 will display following pop-up.	Any non-number value will display the following popup.
 Message Should be between 0 and 255 OK	 Message Only numbers are allowed. Please enter number between 0 to 255 in Red, Green and Blue fields. OK

Question 6 [8]

The records of Chikungunya victims are stored in a txt file name ***victims.txt***. Each record contains name followed by age followed by district; the record are stored in ascending order of district name. Write a program that will read the records from the file, find the district with **highest** Chikungunya victims, and display the district name with victim count in console.

victims.txt	**Expected Output**
Bari-45-Barishal Hasan-50-Barishal Kayes-25-Dhaka Rawnak-35-Dhaka Jabed-43-Dhaka Sumi-48-Rajshahi	Dhaka-3

Question 7

Find out the output of the following Java program. [8]

```
import java.util.*;

class Device implements Comparable<Device> {
	String name, type;
	int price;
	public Device(String name, String type, int price) {
		this.name = name;
		this.type = type;
		this.price = price;
	}
	@Override
	public String toString() {
		return new String(name + " " +price);
	}
	@Override
	public int compareTo(Device o) {
		return price - o.price;
	}
}

public class CollectionTest {
	public static void main(String[] args) {
		ArrayList<Device> devices = new ArrayList<Device>();
		devices.add(new Device("Samsung S7", "Mobile", 1000));
		devices.add(new Device("iPhone 6", "Mobile", 1150));
		devices.add(new Device("Huawei P7", "Mobile", 500));

		List<Device> laptops = new LinkedList<Device>();
		laptops.add(new Device("Dell Alienware", "Laptop", 3500));
		laptops.add(new Device("MacBook", "Laptop", 2500));
		laptops.add(new Device("HP", "Laptop", 1400));

		for (Device d : devices)
			System.out.println(d);

		devices.addAll(laptops);

		Collections.sort(devices, new Comparator<Device>() {
			@Override
			public int compare(Device o1, Device o2) {
				return o1.name.compareTo(o2.name);
			}
		});
		System.out.println("\nAfter sorting devices: ");
		for (Device d : devices)
			System.out.println(d);

		Collections.sort(laptops, Collections.reverseOrder());
		System.out.println("\nAfter sorting laptops: ");
		for (Device d : laptops)
			System.out.println(d);

		devices.add(1, new Device("Sony", "Laptop", 2000));

		HashMap<String, Integer> map = new HashMap<>();

		map.put(devices.get(3).name, devices.get(3).price);
		try {
			System.out.println(map.get("MacBook"));
			System.out.println(map.get("Huawei P7"));
		}
		catch (Exception e) { e.printStackTrace   (); }
	}}
```