



United International University
CSI 211: Object Oriented Programming, Spring 2018
Final Exam
Total Marks: 40, Time: 2 hours

Answer all questions

1. (a) Edit the **Player** class in the following code so that the code gives the expected output. Do not just print the expected output. Do not edit the **Competition** class. You are not allowed to add any variables in the **Player** class. Only add necessary methods. Will your code give exactly the same output as the expected output? Explain the reason. [3+1]

Code:	Expected Output:
<pre>class Player implements Runnable{ Thread t; } class Competition{ public static void main(String[] args) { Player player1 = new Player("Cersei"); Player player2 = new Player("Jamie"); Player player3 = new Player("Tyrion"); } }</pre>	<pre>Cersei starts the race. Jamie starts the race. Tyrion starts the race. Cersei crossed 0 obstacle. Jamie crossed 0 obstacle. Tyrion crossed 0 obstacle. Cersei crossed 1 obstacle. Jamie crossed 1 obstacle. Tyrion crossed 1 obstacle. Cersei crossed 2 obstacle. Tyrion crossed 2 obstacle. Jamie crossed 2 obstacle. Tyrion finishes the race. Cersei finishes the race. Jamie finishes the race.</pre>

- (b) What will be the output of the following code? [4]

```
1 class NestedException {
2     public static void main(String args[]) {
3         try {
4             try {
5                 System.out.println("Inside B");
6                 int m = 115 / 0;
7                 System.out.println(m);
8             } catch (ArithmeticException e1) {
9                 System.out.println("Exception: E2");
10            }
11            try {
12                System.out.println("Inside A");
13                int m = 110 / 0;
14                System.out.println(m);
15            } catch (ArrayIndexOutOfBoundsException e2) {
16                System.out.println("Exception: E1");
17            }
18            System.out.println("Just other statement");
19            } catch (ArithmeticException e3) {
20                System.out.println("Arithmetic Exception");
21                System.out.println("Inside parent try catch block");
22            } catch (ArrayIndexOutOfBoundsException e4) {
23                System.out.println("ArrayIndexOutOfBoundsException");
24                System.out.println("Inside parent try catch block");
25            } catch (Exception e5) {
26                System.out.println("Exception");
27                System.out.println("Inside parent try catch block");
28            }
29            System.out.println("Lets give another input");
30        }
31    }
```

- (c) You are given a text file named “numbers.txt” which contains some numbers in each line that are separated with commas. Write a Java program to read the file and for each line print the sum of the numbers in console. A sample input and output is provided below: [4]

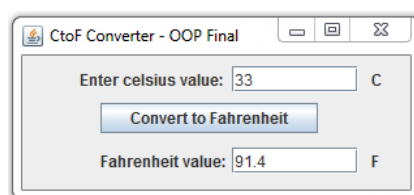
input.txt:	console:
10,11,12	33
1,18	19
2,13	15
33,22,1,1	57
1	1

```
class A {
    void meth() {
        \\Use class here
    }
}
class C {
}
```

2. (a) You need to use an object of a class **B** *extends* **C** inside the method **meth()** in the code above. You will not have to use class **B** anywhere else in your code. However, you need to make several objects of class **B** inside the method **meth()**. Will you make **B** a local class or an anonymous class? Explain the reason behind your decision and complete the code accordingly. [2]
- (b) Create Java classes **Triangle**, **Rectangle** and **Shape2D** where **Shape2D** is extended by the others. **Shape2D** also implements **Area** interface, which only has one method **getArea()**, that returns area of corresponding Shape. Your classes should be implemented such that running the following main method outputs the corresponding comments. You also need to add a method **compare** in Main class. Method **compare** may take any of the two shapes and prints which shape has larger area or prints “Same area” if both have same areas. However, you are **not** allowed to change the **main()** method. [6]

```
1 class Main
2 {
3     public static void main (String[] args)
4     {
5         Triangle t = new Triangle(10, 5);
6         System.out.println("Triangle area: " + t.getArea());
7         // Triangle area: 25
8         Rectangle r = new Rectangle(10, 3);
9         System.out.println("Rectangle area: " + r.getArea());
10        // Rectangle area: 30
11        System.out.println(compare(t, r)); // Rectangle
12        t = new Triangle(10, 10);
13        System.out.println(compare(r, t)); // Triangle
14        System.out.println(compare(r, r)); // Same area
15        System.out.println(compare(t, t)); // Same area
16    }
17 }
```

- (c) Write a generic Java method **maximum(arg[])** that returns the maximum element of the argument array. [4]
3. (a) You are required to complete a Java GUI application that has the functionality of converting temperature from Celsius value to Fahrenheit value. The application takes Celsius value from one JTextField (namely textFieldCelsius), converts this value into Fahrenheit value on click of a JButton (namely btnConvert) and displays the Fahrenheit value into another JTextField (namely textFieldFahrenheit). Suppose necessary code for GUI creation is already provided. You **only** need to add an ActionListener to the button. Formula for converting C to F: $F = (C \times (9/5)) + 32$. [4]



- (b) One married couple, *Jack* and *Rose*, share a Bank account. They should first check the balance in the account before making a withdrawal request. But very often their *cheques* bounce and need to pay fine for an overdrawn request. *Rose* identifies that *Jack* checks the balance (say \$100) and he needs to withdraw (\$50), therefore no problem. Problem arises when *Jack* plans for a sleep before withdrawing the money. In the meantime, *Rose* checks the balance (currently \$100) and withdraws the money (\$70), *Jack* still sleeps. When *Jack* wakes up, he requests for withdrawal of \$50 knowing that the account still has \$100. Hence, he requests for withdrawal of \$60, and receives an overdrawn (\$30) notification. Now, you have to write a java code with available *concurrency* libraries to solve the above problem. You have two classes: *BankAccount*, and *JackAndRoseWithdraw*. Both *Jack* and *Rose* check the balance and withdraw money. Your code restricts to withdraw money by *Rose* from the account until after *Jack* wakes up. [6]
- (c) Consider the following code. Make necessary changes in the class *Car* so that the main method gives the expected output. Do not change anything in the *Test* class. [3]

<u>Code:</u>	<u>Expected Output:</u>
<pre>import java.util.ArrayList; import java.util.Collections; class Test{ public static void main(String[] args) { ArrayList<Car> a = new ArrayList<Car>(); a.add(new Car(150.0, 5)); a.add(new Car(120.0, 9)); a.add(new Car(125.0, 8)); a.add(new Car(120.0, 7)); Collections.sort(a); for(Car c : a){ System.out.println(c.speed + " , " + c.durability); } } } class Car{ double speed; int durability; public Car(double speed, int durability) { this.speed = speed; this.durability = durability; } }</pre>	<pre>120.0 , 7 120.0 , 9 125.0 , 8 150.0 , 5</pre>

Now, make necessary changes in the *Test* class so that, duplicate values are not present in the ArrayList. [Hint: You could use other Collections objects, or ensure duplicate elements are not added to the ArrayList, or get rid of the duplicate elements after adding.] [1]

- (d) Following table contains data of books in the form of ids (*keys*) and titles (*values*). By using which data structure from *Java Collections* framework, adding and searching books by ID will be most efficient? Why so? [2]

ID (key)	Title (value)
101	Structured Programming
102	Algorithms
103	Operating Systems