# CS 250 - Fall 2021 - HW#2

Due: Sep 15, 2021, 11:59 pm

## Objectives

The objectives of this assignment are:

- Familiarizing with the problem-solving steps: analysis, design, implementation, and testing.
- Practice editing, compiling, running, and revising a Java application.
- Practicing exception handling
- Practicing using interface
- Practicing I/O programming

---

## Part 1. Non-programming Questions ( 50 points)

### Q1. (18 points) Choose one correct answer for each sub-question below.

**1. Which type of exception that Java requires to be explicitly handled or declared in the method it may be thrown?**

| | |
|---|---|
| a. Checked exception | b. Unchecked exception |
| c. Both checked and unchecked | d. None of them |

**2. A subclass can only inherit __________ attributes and methods from its superclass.**

| | |
|---|---|
| a. public | b. protected |
| c. public and protected | d. all except protected |

**3. In a Java program, we can throw an exception inside the __________ .**

| | |
|---|---|
| a. try block | b. catch block |
| c. main method | d. finally block |

**4. The root of Java's exception class hierarchy is the __________ class.**

| | |
|---|---|
| a. Exception | b. Throwable |
| c. IOException | d. RuntimeException |

**5. In Java, ___________ files are more portable and platform independent.**

| | |
|---|---|
| a. binay | b. text |

c. both text and binary

d. raw

**6. In Java, a class can extend ________ superclass (parent class) and implement ______ interface.**

a. only one, only one

b. only one, more than one

c. more than one, only one

d. more than one, more than one

## Q2. (16 points)

Implement the following method (complete Java code below the //TODO comment:

```
/**
 * This method receives the height and weight of a person, calculate BMI,
 *         and classify the body mass
 * @param weight: weight in pounds (lb)
 * @param height: height in inches  (ft)
 * @return: classification string: Underweight, Normal, Overweight, and Obese
 */
public static String calcBMI(double weight, double height) {
        //TODO add code below
}
```

Below is the requirements and guidelines for this method:

- You need to check if the weight and height parameters are valid (should be positive). If they are invalid, throw a new instance of IllegalArgumentException with the error message “Invalid inputs!!!”
- If two parameters are valid, calculate the BMI using the equation: BMI = weight (lb) / [height (in)]$^2$ x 703
- Using if-else or switch on the value of BMI and return the classification as follows:

| | |
|---|---|
| - BMI < 18.5 | => Underweight |
| - 18.5 <= BMI < 25 | => Normal |
| - 25.0 <= BMI < 30 | => Oveweight |
| - BMI >= 30 | => Obese |

## Q3. (16 pts)

Analyze the following code

```
public static void hw2_q3(int num) {
        try {
                System.out.println( "Begin the try block code" ) ;
```

```
                        if (num > 100)
                                throw new Exception (num + " is too large" ) ;
                        System.out.println( "End the try block code" ) ;
                } catch ( Exception e ) {
                        System.out.println( "ERROR: " + e.getMessage ( ) ) ;
                }
        }
```

Determine the output of the Java program if you call such above method in the main() method with the following inputs and justify your answers (explain why such outputs happen):

a. hw2_q3(0)

b. hw2_q3(101)

c. hw2_q3(150.0)

---

## Part 2. Programming Question (50 points + bonus 10 points)

### 2.1 Problem Descriptions

Develop a Java application to read an input text file named grades.txt which stores a number of students' grades separated by spaces (they can be in multiple lines). The number of grade items in the input file is unknown. Your program needs to read such a file, parse the grades, calculate the total grades and the average, and save such outputs to a file name grades_stats.txt.

**Example input/output**: Given the following input grades.txt file

```
1.2 3.9 3.2 2.5
2.2 2.3 4.0 3.5 3.9
```

After running, your program should output the following text file named grades_stats.txt with the following content

```
Total:        26.7
Average:    2.967
```

### 2.2 Problem Design and Implementation Guideline

Below is the general guideline to help you implement this program:

- First, you need to create the grades.txt file as an input file for your program. You can do this task by right-clicking on the src item of your Eclipse project, choose File, New. Then enter grades.txt in the File Name text box and press Enter. A

new sub-window named grades.txt will be open on your Eclipse. Then, you can enter the grade items in multiple lines and separated them by spaces, similar to the above example file. Remember to save it.
- You can follow the lab code to write code to open the file grades.txt using the file path “src/grades.txt”, read each line, parse the line into multiple tokens using the split() method, and parse tokens to numbers. Then you add the parsed number to a total variable and count the number of grade items so far.
- You are required to use try-catch to catch any exception that can occur in your program (File not found, IO exception, parsing errors, etc.)
- After you read all the lines and parsing the inputs, you can calculate the average by dividing the total by the count of grade items.
- Finally, write code to output the total grade and average grade to an output file name grades_stats.txt. You can use the file path “src/grades_stats.txt” to output the file so this file will be in the same directory (src) of your project.

## 2.3 Coding Requirements

**A. Source code**: Write your **own** code

**B. Program requirements**: Your program can solve and meet all of the requirements described in 2.1 and 2.2. Do not write additional code that is not required.

**C. Coding style:** Your source code must be properly indented and contain adequate comments for class, methods, variables, etc.

- At the beginning of your Java class file, specify the following items:
    - Your name, the assignment number for this program, the date you create the file
    - The problem description that defines your application
    - Enumerate (list) the goals of your application
    - Enumerate (list) the inputs of your application
    - Enumerate (list) the outputs of your application
- Before each method in your class, specify the following items:
    - Specify the purpose of the corresponding method
    - Enumerate the possible inputs and outputs of the corresponding method
    - Write down the pseudocode to implement the corresponding method
- Others comments for variables, inside method implementation

### 2.4 Grading Rubrics

Your code in this part will be evaluated using the following grading scheme (50 points for completed functional source code + bonus 10 points for good coding style with code comments):

| Points | Item |
|---|---|
| 25 | Correct class, variable, and method definition |
| 10 | Consistent coding-style, such as indents, curly bracket positions, etc. |
| 15 | Have all required variables and methods |
| 10 | Good and helpful comments |

---

## Assignment Submission Instructions

1. Answer Part 1 questions in a WORD, PDF, or any other document application

2. Compress the whole Eclipse source code of your project in part 2 in the zip format using the Export function

Submit two items above to the HW#2 assignment folder on D2L.