ENSF 409
Winter Semester 2021
Assignment 07

Assignment Instructions

Complete and submit exercise 18.2 (Lesson 18). This assignment is due on March 08, 2021 by 5:59 PM Mountain Time.

General Instructions

Academic Integrity and Collaboration

This is an individual assignment. Your submission must be your own original work. You may not work with others to complete the assignment, although you may collaborate on exercises which are not submitted. It is not academic misconduct to ask for help on the discussion boards, but you may not copy code or complete answers from your peers.

Submission

You must submit your assignment in the specified format. Due to the number of students in the course, we are using automated grading. If your submission does not have the correct folder structure and file names, it will not be marked. This is a strict requirement.

File and directory names are case-sensitive.

You must submit your assignment to the appropriate D2L dropbox folder. You may submit multiple times before the assignment deadline, but only the last submission will be graded. Previous uploads are discarded by D2L. Therefore, each upload must be a complete submission. Do not submit multiple files across separate submissions.

The assignment must be submitted as a single zip folder named with your student ID. Furthermore, when the file is unzipped, the contents must be in a folder with the same name. You may wish to verify that your zip file has been correctly generated (includes the student ID directory, contains all files). You can do this by copying the zip file into another folder, unzip it, and examine the structure.

Within the folder with your ID number, you must create a subdirectory for each exercise within the assignment. Use lowercase and employ a _ to separate sub exercise numbers. For example, Exercise 1.3 should be in a folder `exercise1_3`.

Below is an example for submitting Assignment 1, assuming a student ID of *1234765*.

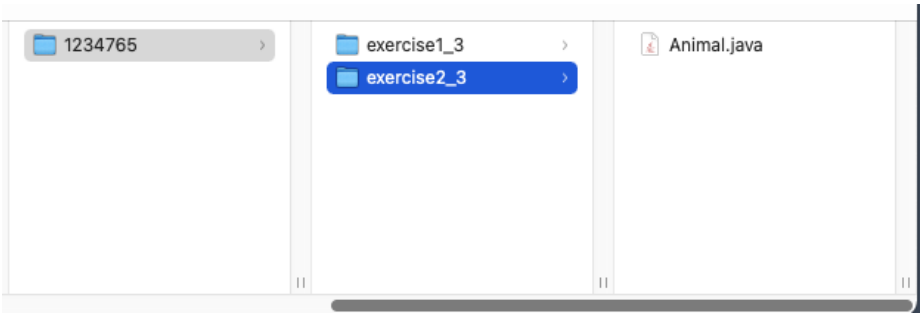

Figure 1: The folder structure for Assignment 1

```
ENSF409 > ls
1234765
ENSF409 > zip -r 1234765.zip 1234765
  adding: 1234765/ (stored 0%)
  adding: 1234765/exercise2_3/ (stored 0%)
  adding: 1234765/exercise2_3/Animal.java (stored 0%)
  adding: 1234765/exercise1_3/ (stored 0%)
  adding: 1234765/exercise1_3/Hello.java (stored 0%)
ENSF409 > ls
1234765         1234765.zip
ENSF409 >
```

Figure 2: Creating the zip file in Mac/Linux

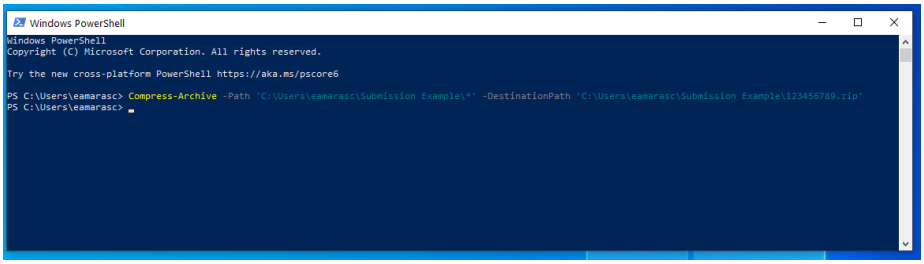

Figure 3: Creating the zip file in Windows

Evaluation

Code which does not compile will not receive any points. Your code must compile and execute correctly to receive full marks. Assignments are graded using OpenJDK 11. While we do not anticipate that any of the assignments in this course will execute differently in any version of Java from 8 on, you should use this version for optimal results.

When style conventions are introduced in lessons, all subsequent submissions should adhere to the conventions.

Deadline

All homework assignments are due at 17:59 (5:59 PM) Mountain Time. You are responsible for the conversion between Mountain Time and your local time zone, if applicable. Be aware that the switch from Mountain Standard Time to Mountain Daylight Time occurs during the term.

It is recommended that you do not leave your submission until the last minute in case of technical issues. Once the dropbox folder closes, you will not be able to submit your assignment. Late submissions will not be accepted.

If there is a technical problem and you are unable to submit via the dropbox, do not email your assignment. The University of Calgary email system will remove zip files. Instead, you may email a link to download your .zip folder from OneDrive. The email must be received by Dr. Barcomb and Dr. Marasco before the assignment deadline and the OneDrive folder should not show any changes to the .zip file after the deadline has passed.

ENSF 409
Exercises - Lesson 17

The following exercises are described in the video for Lesson 17.

Exercise 17.1

1. Return to the code you wrote for exercise 14.3
2. Do not assume that every log line will be well-formatted
3. Add exceptions for everything which could go wrong

> - Two types of situations to consider:
>   - Regular expression does not match, e.g., line contains [20/Jun/**800**:10:05:44] and DateTime does not match
>   - A value is not found in the enum, e.g., line contains [20/**Foo**/2020:10:05:44], which matches the DateTime regular expression but is not found in the enum Months.

Exercise 17.2

1. Return to the code you wrote for exercise 17.1
2. Create user-defined exceptions for the different error situations which can occur
3. Change the code to throw your user-defined exceptions

> - Two types of situations to consider:
>   - Regular expression does not match, e.g., line contains [20/Jun/**800**:10:05:44] and DateTime does not match
>   - A value is not found in the enum, e.g., line contains [20/**Foo**/2020:10:05:44], which matches the DateTime regular expression but is not found in the enum Months.

Exercise 17.3

1. Catch the exceptions that you threw in Exercise 17.2
2. Try a specific catch for one type of exception, and let a general catch capture the other
3. Try adding one catch at the point where the error occurs, and another in main

ENSF 409
Exercises - Lesson 18

The following exercises are described in the video for Lesson 18.

Exercise 18.1

1. Download the necessary .jar files for JUnit 4
2. Follow the tutorial to show one successful test run and one failed test run for `Calculator.java`

- Download `junit.jar` and `hamcrest-core.jar`
  - https://github.com/junit-team/junit4/wiki/Download-and-Install
- Follow the "Getting started" tutorial
  - https://github.com/junit-team/junit4/wiki/Getting-started
  - You can use the provided commands or add to your CLASSPATH settings

Tip: Remember to replace XX in the tutorial commands with the correct version of JUnit 4.

Important: Regardless of your IDE, you must be able to compile and run tests from the command line.

Exercise 18.2

1. You have been asked to test and correct a colleague's code. Download the provided code file `StringProcessor.java` from the course repository
2. Using the provided requirements, write unit tests to test the functionality of the given file
3. Correct the code so that it fulfills all of the requirements
4. You may create additional private data members or methods, but only the specified constructor, getter, and methods will be tested

- Class is called `StringProcessor`
- Has a constructor that takes in a single String to be stored
- Has a getter method for the stored string value
- Provides a public method `addTogetherMirror`
  - Accepts one String argument
  - Adds the input String with the stored data member String
  - The method should trim the leading and trailing whitespaces from each String before adding them together
  - Reverses the combined String to return the "mirror-image" String in all lowercase values (e.g. "cat" -> "tac")
- Provides a public static method `idProcessing`
  - A veterinary clinic needs to create unique identifiers for all its animal clients up to December 2021. The identifier will consist of the owner's first and last initials, the first initial of the animal's name, and the pet's year of birth (e.g. EMD2010)
  - Accepts three Strings and an integer as arguments (first name, last name, pet name, year)
  - All names must begin with a capital and may contain punctuation or spaces (e.g. O'Malley)
  - All names must be a minimum of two letters and a maximum of 26 letters
  - Must check that each argument is in the valid format (throws IllegalArgumentException for any invalid arguments)
  - An identifier cannot be provided if a pet has not yet been born (any time after 2021)
  - Returns the desired identifier as a String
- Provides a public method `secretCode`
  - Accepts an integer offset that is used to encode the stored data member String
  - Each character in the String should be shifted by the provided offset
  - If the end of the alphabet is reached, return to the start and keep counting (e.g. offset = 3, A –> D, Z –> C, a –> d, z –> c)
  - Case is preserved: A – Z is a separate set from a – z
  - Returns the encoded String

Tip: Write the tests before you look at the provided code in detail. Don't incorporate your colleague's mistakes into your tests!

Tip: You only need to submit the corrected `StringProcessor.java` file, not your tests.