ENSF 409
Winter Semester 2021
Assignment 01

Assignment Instructions

Complete and submit exercise 1.3 (Lesson 01) and exercise 2.3 (Lesson 02). This assignment is due on January 18, 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*.

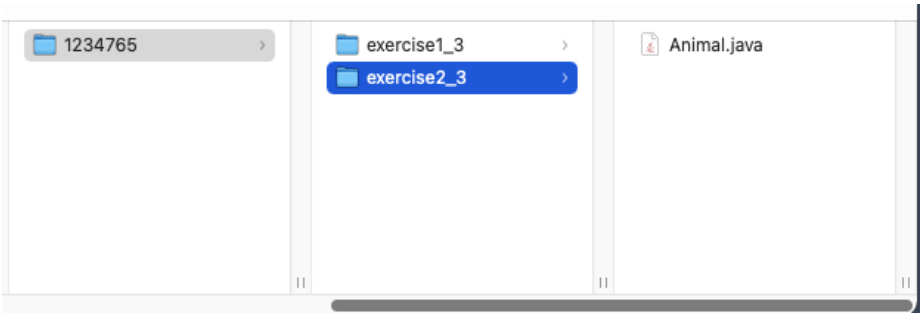

Figure 1: The folder structure for Assignment 1

```
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

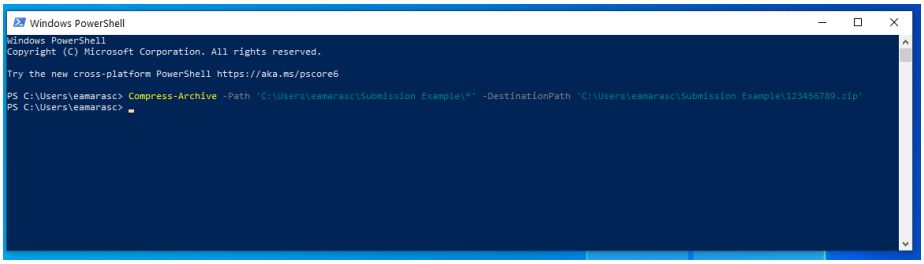

Figure 3: Creating the zip file in Windows

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 1

The following exercises are described in the video for Lesson 1.

Exercise 1.1

1. Create a directory/folder for this exercise
2. Type in the program as shown
3. Save the file as `Hello.java`
4. Compile the program using `javac Hello.java`
5. Run the program using `java Hello`

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

Tip: The Java Development Kit (JDK), which includes the compiler, can be downloaded from Oracle: `https://www.oracle.com`

Exercise 1.2

1. Modify the original program to include accurate documentation
2. Generate documentation using `javadoc Hello.java`
3. Look in the directory to see the files created
4. View the files

Add accurate author information, instead of the default text shown in the example. The remaining comments and documentation do not need to be the same as in the example, but must provide relevant information and generate documentation without errors.

```
/**
@author    Firstname Lastname <a href="mailto:firstname.lastname@ucalgary.ca">
    firstname.lastname@ucalgary.ca</a>
@version    1.1
@since      1.0
*/

/*
Hello is a simple example program which prints the classic message:
Hello, world
*/

public class Hello {
    /**
     This prints "Hello, world" to the terminal window.
     @param args Optional command-line argument
    */
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
} // End of class declaration
```

Note: Remember to update the version number as you change the program.

Exercise 1.3

1. Modify `Hello.java` to use all of the primitive data types: boolean, char, byte, short, int, long, float, double
2. Implement both an automatic and explicit cast
3. Compile and run the program

The snippet below illustrates a modification which includes `boolean`.

```
public static void main(String[] args) {
    boolean ExampleVariable = true;
    System.out.println("Hello? " +
        ExampleVariable);
}
```

ENSF 409

Exercises - Lesson 2

The following exercises are described in the video for Lesson 2.

Exercise 2.1

1. Type in the program as shown and add appropriate documentation and comments; save as `Creature.java`
2. Compile and run the program, giving one command-line argument (e.g., *cat*)
3. Run the program with two command-line arguments (e.g., *cat panther*)
4. Run the program with zero command-line arguments

```
public class Creature {
    /**
    @param args Handles command-line argument
    */
    public static void main(String[] args) {
        System.out.println("This is a placeholder for Creature ” + args[0]);
    }
}
```

Exercise 2.2

1. Edit `Creature.java` to incorporate the code in the example
2. Be sure your documentation and comments are up-to-date, and that you are using correct capitalization
3. Compile the program and examine the contents of the directory

```
public class Creature {
    public static void main(String[] args) {
        Animal myAnimal = new Animal();
        String myType = myAnimal.animalType();
        System.out.println("This is a placeholder for Creature " + myType);
    }
}

class Animal {
    private String animalType = "dog";

    public String animalType() {
        return animalType;
     }
}
```

Exercise 2.3

1. Create a file which implements the class shown in the UML class diagram.