ENSF 409
Winter Semester 2021
Assignment 10

Assignment Instructions

Complete and submit the assignment according to the instructions below. This assignment is due on Tuesday, April 06, 2021 by 5:59 PM Mountain Time.

Debugging and fixing code is an important software development skill. Learning from your past mistakes allows you to become a stronger and more detail-oriented engineer. In this assignment, you will have the opportunity to fix your past assignment code and UML diagrams.

1. Select two past submissions from Assignments 3 to 7 and Quiz 2 where you did not earn a perfect score.
2. Use your graded test feedback, course resources, and the discussion boards to correct your original submission.
3. Write a half-page reflection that explains how you determined your past errors and what you did to fix them. For example, did you write unit tests? Did you seek help during office hours to better understand your code's performance? Did you have to rewrite your entire program or just a particular method?
4. You must upload both corrected submissions in their corresponding exercise/quiz directories as specified in their original handouts. Both the directories should be in a single folder with your ID number as usual. Note that packages were not used until Assignment 7. If you are resubmitting the UML diagram from Assignment 4, the PDF should be saved in a separate exercise directory as well.
5. Your reflection must be in PDF format and must be included in your ID number folder (not within a particular exercise directory). A well-written reflection will be counted as 1% towards your participation grade.
6. Your Assignment 10 grade will be averaged equally between your two resubmissions. Note that your original grades for Assignments 3 to 7 and Quiz 2 will not be replaced.

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*.

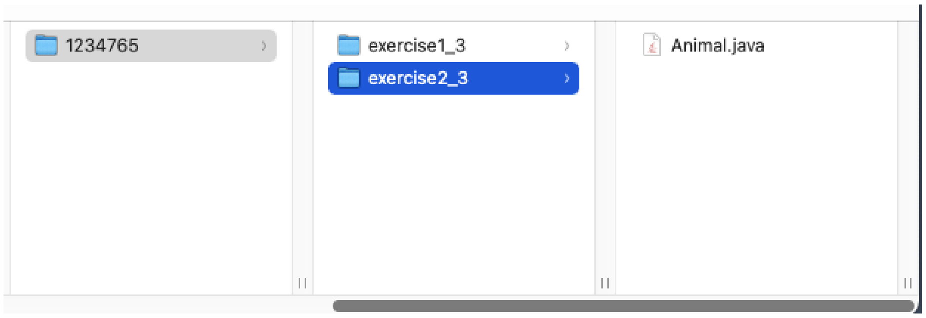
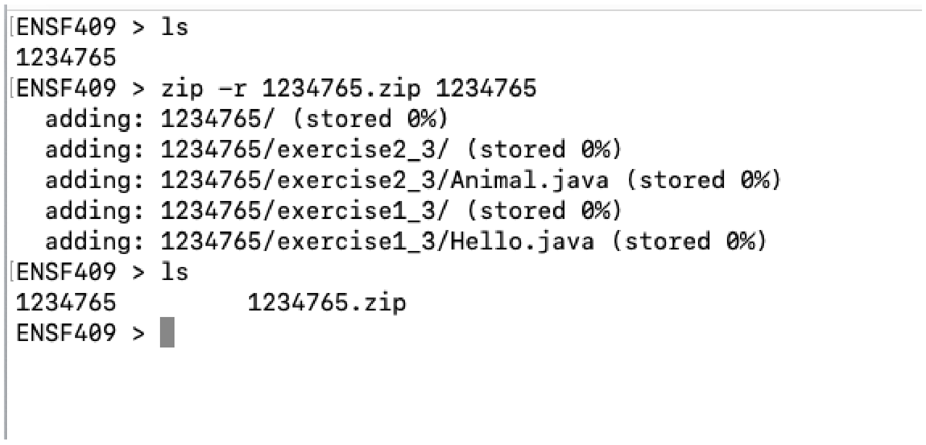
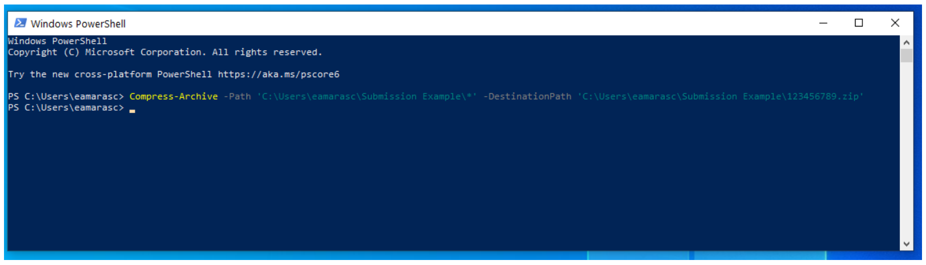


Figure 1: The folder structure for Assignment 1



```
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



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\eamarasc> Compress-Archive -Path 'C:\Users\eamarasc\Submission Example\*' -DestinationPath 'C:\Users\eamarasc\Submission Example\123456789.zip'
PS C:\Users\eamarasc> 
```

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 21

The following exercises are described in the video for Lesson 21.

Exercise 21.1

1. Modify the code in `03_Calculator` to create Lambda expressions for division, subtraction, and addition
2. Check your answer against the code in `04_Calculator`

```
public class Calc {
  public interface MathWithInts {}
    public int operation(int a, int b);
  }

  public int twoDigitOp(int a, int b, MathWithInts op) {
    return op.operation(a, b);
  }

  public static void main(String[] args) {
    MathWithInts multiplication = (a, b) -> a * b;

    Calc myApp = new Calc();
    System.out.println("2 * 10 = " +
      myApp.twoDigitOp(2, 10, multiplication));
  }
}
```

Tip: The Calc interface does not specify what operation should be performed, and neither does twoDigitOp. This information is contained solely in the Lambda expression.

Exercise 21.2

1. Create an ArrayList of Integers.
2. Print the results of the following operations, in this order:
   (a) Multiply each number by four
   (b) Exclude numbers greater than 100

(c) Sort the numbers

(d) Add together all the digits in the number

(e) Compare your code with the example in `06_Exercise`

Try changing the order of operations to see if the results match your expectations.