ENSF 409
Winter Semester 2021
Assignment 02

Assignment Instructions

Complete and submit exercise 4.1 (Lesson 04) and 5.2 (Lesson 05). This assignment is due on January 25, 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*.

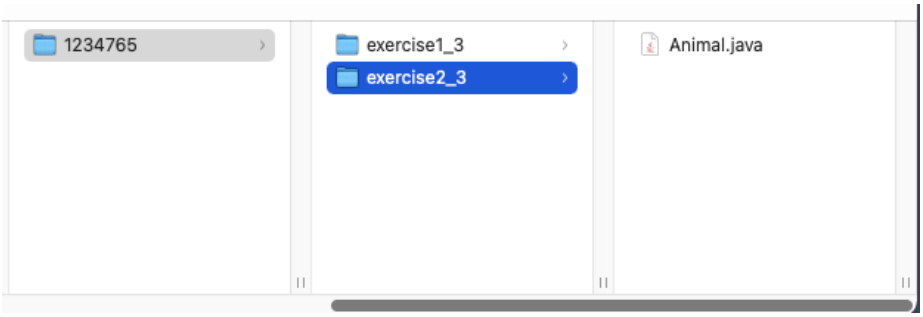

Figure 1: The folder structure for Assignment 1

```
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

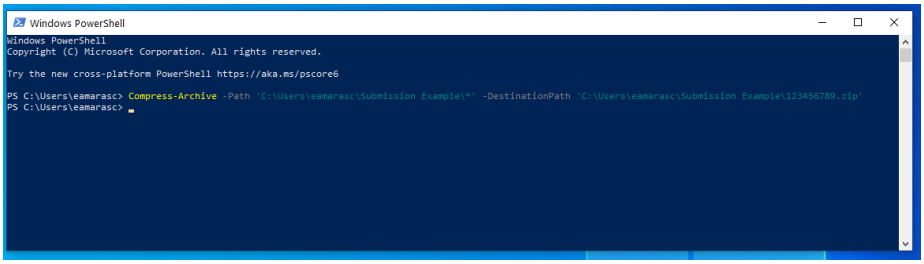

Figure 3: Creating the zip file in Windows

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 3

The following exercise is described in the video for Lesson 3.

Exercise 3.1

1. Write down what you expect the output of this program to be
2. Run the program
3. Observe the output. If your expected output did not match the actual output, work through the code to determine the extent of each variable's scope.

```
public class Confusion
{
  static int one = 11;
  private int two = 33;
  public void methodOne(int one)
  {
    Confusion test = new Confusion();
    this.one = 22;
    two = 44;
     System.out.println("Confusion.one: " + Confusion.one);
     System.out.println("test.one: " + test.one);
     System.out.println("test.two: " + test.two);
     System.out.println("two: " + two);
   }

  public static void main(String args[])
  {
     Confusion test = new Confusion();
     test.methodOne(5);
   }
}
```

ENSF 409

Exercises - Lesson 4

The following exercises are described in the video for Lesson 4.

Exercise 4.1

1. Look at the documentation on java.util.Arrays.
2. Create a class 'SimpleArrays' which has the characteristics specified to the right.
3. You may include additional methods, but no methods beyond those specified need to be included.
4. You may use additional methods provided by java.util.Arrays, beyond those specified.

- Has a constructor which can be called with one argument: a String
  - The constructor creates an instance variable (a 4-element array of Strings)
  - Uses the `java.util.Arrays` method fill to populate all elements of the array with the String
- Overloads the constructor with a zero argument version
  - The array is populated with the default String "Hello, ENSF 409"
- Provides a method arrayConcat
  - Accepts an array index; if no array index is supplied, 0 is used by default
  - This method should have default behavior if the index supplied exceeds the bounds of the array (IndexOutofBoundsException error)
  - Returns a String consisting of all the elements of the array, from the provided index to the end, concatenated, separated by the # character (no whitespace should be introduced)
- Provides a method arrayCrop
  - Accepts two integers as arguments, corresponding to starting and ending array indices. The starting and ending indices are inclusive.
  - Returns a String consisting of all the elements between the two indices specified, concatenated, separated by the # character (no whitespace should be introduced)
  - If the ending integer is smaller than the starting integer, switch the two (use the start as the end, and the end as the start)
  - If either integer is not a valid index, return "Fail"
  - If the two integers are the same, return "Match"

Tip: `arrayCrop()` and `arrayConcat()` have some common behavior, which should not be implemented twice

Important: When you use dynamic array allocation for an array of objects, you only create references, not objects

Important: Remember to document your code

Exercise 4.2

1. Write down what the output of this code snippet will be, without running it.
2. Check if you were correct.
3. If you were correct, try setting some of the values to see if the output matches expectations.
4. If your answer was wrong, review the lesson on multidimensional arrays and modify the code until you can predict the output.

```
int intNested[][][];
intNested = new int[2][3][];
intNested[0][0] = new int[3];
intNested[0][1] = new int[4];
intNested[1][0] = new int[5];
intNested[1][1] = new int[1];
intNested[0][2] = new int[2];
intNested[1][2] = new int[1];
for(int i=0; i < intNested.length; i++) {
   for(int j=0; j < intNested[i].length; j++) {
      for(int k=0; k< intNested[i][j].length; k++) {
          System.out.print("intNested[" + i + "][" + j + "][" + k + "] ");
          System.out.println(intNested[i][j][k]);
      }
   }
}
```

Reminder: This is a snippet, not a complete program. You will need to add a class and method to compile and run the code.

ENSF 409

Exercises - Lesson 5

The following exercises are described in the video for Lesson 5.

Exercise 5.1

1. Type in the program as shown and add appropriate documentation and comments
2. Save the file as `JavaStrings.java`
3. Create two more strings of your choice using two different constructors from the String class
4. Find the length of all five Strings. Calculate and print the total sum of all the String lengths

```
...
public class JavaStrings {

    public static void main(String[] args) {

        String animalFact1 = "Horses are mammals.";
        String animalFact2 = new String("Elephants are mammals.");
        String animalFact3 = new String (animalFact1);
    }
}
```

Exercise 5.2

1. Look at the documentation on the String class.
2. Create a single class 'JavaStrings' which includes the three methods specified below.
3. You may include additional methods, but no methods beyond those specified need to be included.
4. You may use additional methods included in the String class beyond those specified.

- Provides a method addTogether
  - Accepts two Strings as arguments
  - The method should trim the leading and trailing whitespaces from each String and add them together
  - Returns the length of the trimmed, concatenated String
- Provides a method idProcessing
  - A veterinary clinic needs to create unique identifiers for its animal clients. 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)
  - Write a method that will:
    * Accept three Strings and an integer as arguments (first name, last name, pet name, year)
    * Returns the desired identifier as a String
- Provides a method secretCode
  - A famous pizzeria wants to encode its secret recipe. All ingredients are referred to using code words. Vowels (a, e, i, o, u) are replaced with a "z" and only the first three letters are used (e.g. tomato = tzmztz = tzm)
  - Write a method that will:
    * Accept a String as an argument
    * Return the corresponding secret code String
    * You may assume that all ingredients are at least three letters long

Tip: Be sure to check the documentation for methods that might be helpful.

Exercise 5.3

1. Consider the String provided below
2. Using a regular expression, count how many instances of the letter E are in the String (both upper and lowercase)
3. Use StringTokenizer to count and print each token in the String when delimited by whitespace
4. Use StringBuilder to insert a colon after "409" and print the updated statement

```
...
public class JavaStrings {

    public static void main(String[] args) {
   String myString = "ENSF 409 Principles of Software Development";
     }
}
```