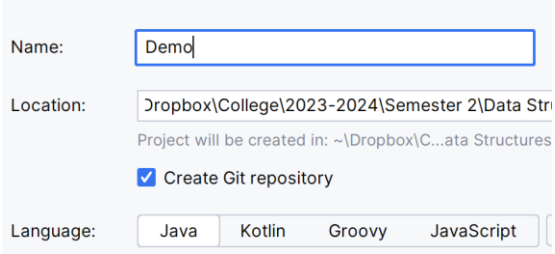
Exercise Sheet Set-Up:

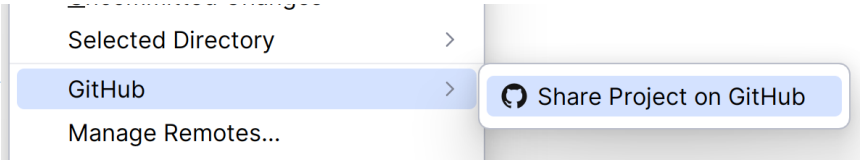
Before you begin programming:

1. Create a new account on GitHub (if you haven't already got one)
2. Create a new project in IntelliJ and set it to include a git repository. You should use this project for all exercise sheets (not CA submissions) in the module.



3. Upload your git repository to your github account

In IntelliJ's Git menu →



Revision, JavaDoc and Git Exercises

Before you get into algorithms, you need to remember all the skills you built last year. These include user input, conditional logic, loops and method creation. You will use these skills to form the basis of working with Git and JavaDocs.

Loops

Exercise 1.1)

Write a new program (file) that:

- Repeatedly takes in numbers from the user. Input should terminate when the user enters -1.
- Once -1 has been entered, the program should print out the sum and average of the entered numbers

Commit and push your code:

- When the program correctly stops taking in numbers when -1 is entered.
- When the program correctly displays the sum and average correctly

Exercise 1.2)

Write a new program (file) that:

- Takes in two numbers from the user.
- Adds up all the numbers between those the user entered.
- Outputs the result.

Example: If a user enters 3 and 7, then the program should print out 25 (i.e. 3+4+5+6+7).

Your program should still work correctly, even if the user enters the bigger number first (i.e. 7 and then 3).

Commit and push your code:

- When the program takes in the two numbers and adds up all values (but only works with the small number first)
- When the program takes in the two numbers and adds up all values no matter which order they are in

Writing & using `static` Methods

Before you start:

Create a Java class called CalcMax.java.

Exercise 2.1)

Write a static method called getValidInteger() in CalcMax.java that:

- Takes in a String as a parameter
- Uses this text as prompt to the user.
- Confirms the data being entered is a number.
  - If the data is a number, it should be stored.
  - If it is not a number, the user should be informed their entered information was inappropriate, then prompted to enter a value again (using the same text as before).
  - The value entered should be confirmed as a number. If it is not, this process should repeat.
- Returns the entered (and validated) number

Write a JavaDoc for this method. It should:

- Explain what the method does (one line description)
- Explain what happens when the user does not enter a number (on a new line, in a longer description)
- Explain what the parameter is used for (@param tag)
- Explain what the method returns (@return tag)

Commit and push your code:

- When your method takes in a number and returns it
- When your method validates (checks) that the value entered is a number and handles where it's not
- When you have added your Javadoc (your commit message for this should use the "docs" label)

Exercise 2.2)

Write a static method called findMax() in CalcMax.java that:

- Takes in three ints as parameters
- Returns the largest of the three numbers

Note: You may not use any built-in methods from Java to accomplish this.

Write a JavaDoc for this method. It should:

- Explain what the method does (one line description)
- Explain what each parameter is used for (@param tag for each number)
- Explain what the method returns (@return tag)

Commit and push your code:

- When your method is complete
- When you have added your Javadoc (your commit message for this should use the "docs" label)

Exercise 2.3)

Write a new program in CalcMax's main() method that:

- Takes in 3 numbers from the user
  - You should use your `getValidInteger()` to validate each of the numbers as they are entered
  - As `getValidInteger()` was written in the same file as this program (CalcMax.java), you do not need to use anything to call it on, i.e. it can be used as:

    ```
    int num1 = getValidInteger("Please enter an integer:");
    ```
- Displays the highest of the entered numbers.
  - You should use your `findMax()` method to decide which number should be displayed.

Commit and push your code:

- When your program successfully uses `getValidInteger()` to take in 3 numbers
- When the program is complete

Exercise 2.4)

Write a static method called findMax() in CalcMax.java that:

- Takes in an array of ints as a parameter
- Returns the largest number in the array

Note: You may not use any built-in methods from Java to accomplish this.

This is an example of an OVERLOADED method, as there are multiple methods with the same name, but each has a different set of parameters.

Write a JavaDoc for this method. It should:

- Explain what the method does (one line description)
- Explain what the parameter is used for (@param tag for the array)
- Explain what the method returns (@return tag)

Commit and push your code:

- When your method is complete
- When you have added your Javadoc (your commit message for this should use the "docs" label)

Exercise 2.5)

Write a static method called getValidInteger() that:

- Takes in a String prompt and two ints representing the upper and lower limits (inclusive) of allowable values
- Uses the text as a prompt to the user.
- Confirms the data being entered is a number, and that it is within the specified range.
    - If the data is a number AND within the specified range, it should be stored.
    - If it is not a number, the user should be informed their entered information was inappropriate, then prompted to enter a value again (using the same text as before).
    - If it is a number but is not within the specified range, the user should be informed that their value is outside the allowable range, then prompted to enter an appropriate value (using the same text prompt as before)
    - The new value entered should also be confirmed as a number and within the specified range. If it is not, this process should repeat.
- Returns the entered (and validated) number

This is another example of an OVERLOADED method, as there are multiple methods with the same name, but each has a different set of parameters.

Write a JavaDoc for this method. It should:

- Explain what the method does (one line description)
- Explain what happens when the user does not enter a number, or the number is outside the range (on a new line, in a longer description). Your explanation should also mention that the range is inclusive, i.e. it will allow the upper and lower boundary values to be entered
- Explain what the parameters are used for (@param tag for each parameter)
- Explain what the method returns (@return tag)

Commit and push your code:

- When your method takes in a number and returns it
- When your method validates (checks) that the value entered is a number and handles where it's not
- When your method validates (checks) that the number entered is within the specified range values
- When you have added your Javadoc (your commit message for this should use the "docs" label)

Utility Classes

Exercise 3.1)

Copy your getValidInteger() methods to a class called InputUtility.java.

Exercise 3.2)

Modify your answer to exercise 2.3 to use the getValidInteger() method from your InputUtility.

Commit and push your code when this is done. Your commit message should start with "refactor: <*Your message*>".