Lab: Java

Lab Overview

This lab consists of multiple “small” Java exercises, and the last exercise serves as a mini project to bring together everything from the Java sections.

Submission Instructions

Create a new feature branch in your TEKbootcamp repository. Commit and push frequently. When your work is done and ready for review, please merge the feature branch into the develop branch.

Following instructions for CodeGrade submission:

1. Click on the "Launch exercise" button on the exercise page at TEKsystems Academy.
2. In the new CodeGrade window, click on the "Connect Git" button.
3. Select GitHub as your Git host. Connect your TEKbootcamp repository to the exercise*.

After selecting a repository, this repository will be cloned to CodeGrade as a first submission. After this, you can start to use Git like you usually would. Every time you push to the develop branch, it will automatically result in a new submission in CodeGrade. Pushes made after the submission deadline will not be taken into consideration.

* You will only have to log in and authorize CodeGrade once; after that, it will be available for all your other assignments inside CodeGrade too.

WARNING: Be sure to connect the proper repository to the exercise. You will not be able to connect to another one immediately using the CodeGrade connection. You can undo your Git connection with CodeGrade by revoking access to the External Tool "CodeGrade" in your GitHub account only.

Grading Expectations

A solution that meets all the requirements in this lab constitutes a grade of Proficient (3/5). To earn a higher grade, your solution must also apply the code quality concepts you've learned up to this point. Additional functionality that does not improve the quality of the code and only serves to make your assignment look more impressive will not help to raise your grade.

Plagiarism Warning

This assignment is a demonstration of your understanding of the topics covered to-date in the boot camp. You may need to reference tutorials or Q&A sites such as StackOverflow to complete parts of the assignment. It is acceptable to use tutorials/Q&As to learn how to add specific functionality to your project, but you cannot simply copy a tutorial repository, follow a single tutorial from start to finish, or copy the code from a Q&A site to develop your application. See guidelines below for using tutorials:

- Do not fork a tutorial repository. This would be considered plagiarism and result in an automatic grade of zero and potential removal from the boot camp.
- Do not simply follow the tutorial from start to finish.

- Do not copy code from a Q&A site and then change it for your application (a few lines of code are permissible)
- You must be able to explain how all functionality included in your project works.
- List ALL tutorials or aids used, either in your README file, and/or as code comments. Include:
  - The name of the tutorial,
  - Which section(s) you used, and
  - The functionality you learned from the tutorial.
- Failure to identify tutorials/Q&A responses will result in significant reduction of your grade.
- If you work with other boot camp members – or instructional team members – to find solutions, that is acceptable, but must also be clearly identified

Estimated Duration

15-30 Hours

Resources

None

Project: Java

Project Overview

This will be the project for all the exercises in this Lab

Project Setup Instructions

- Using IntelliJ, create a Maven Project:
  - Project SDK - 11
  - Create from archetype - maven-archetype-quickstart
  - Name – java_exercises
  - Location - [student repository root]/java_exercises
  - GroupId - com.teksystems.bootcamp.java_exercises
  - ArtifactId: java_exercises
  - Version - 0.1-SNAPSHOT): 1.0-SNAPSHOT
  - Leave the remaining settings at their defaults and ‘Finish’ creating the project
- In IntelliJ, open the ‘java_exercises” folder

Exercise 1: Additional Inputs

Description

In this exercise you will create a Java console application that will accept continuous integer inputs and will add those inputs to a Total then display that Total. The program will continue to accept input until a non-integer is input by the user; in which case the program will output the sum and exit.

File Location

Under the com.teksystems.bootcamp namespace, create a new package named additionalInputs, and add appropriate classes to provide the required functionality.

Instructions:

Create the following classes and methods:

- NumberAggregator
  - public int addNumber(int number);
    - Adds the input(int) to a running total and returns the current value of the total

Only main() should interact with the user and write to the console.

Include unit tests for the NumberAggregator.

Exercise 2: Find Characters

Description

In this exercise you will create a Java console application that will accept two string inputs and will find characters from the first input inside the second:

- The first input is characters to search for.
- The second input is a string to search for those characters in.
- The output will be a listing of the positions for every unique character in the search list.

File Location

Under the com.teksystems.bootcamp namespace, create a new package named findCharacters, and add appropriate classes to provide the required functionality.

Instructions:

Create the following classes, methods and fields:

- CharFinder
    - public static HashMap<Char, List<int>> findMatchPositions(String charsToFind, String stringToSearch);
        - returns a list of unique characters in the first argument and positions where those characters were found in the second argument

Only main() should interact with the user and write to the console.

Include unit tests for CharFinder.

Exercise 3: Factor by Two

Description

In this exercise you will create a Java console application that will accept an integer input and will output the number of twos that are found when that number is factored. The application will continue to run and process additional input until the user exits by typing in a non-integer value.

File Location

Under the com.teksystems.bootcamp namespace, create a new package named factorByTwo, and add appropriate classes to provide the required functionality.

References:

https://www.w3schools.com/java/java_recursion.asp

https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

Instructions:

Create the following classes, methods and fields:

- Node
  - public abstract int getCountOfTwos();
- ValueNode (extends Node)
  - public constructor
    - takes an int
  - public int getCountOfTwos();
    - returns 1 if the value is 2, else 0
- PointerNode (extends Node)
  - public constructor
    - takes two Nodes
  - public int getCountOfTwos();
    - returns sum of getCountOfTwos() from both Nodes.
- NodeCreator
  - public static Node createNode(int number);
    - returns a ValueNode if the number is 2 or not divisible by 2
    - returns a PointerNode and uses createNode() to generate the constructor arguments

Only main() should interact with the user and write to the console.

Include unit tests for all concrete classes.

Exercise 4: Grid Hopper

Description

In this exercise you will create a Java application that "jumps" between squares on a 2-dimensional grid, based on a location pointer (in each grid square) that indicates the next grid coordinate to move to. The user will input a number to define the height/width of a square grid (ex. If 2 is input then the grid will be a 2x2 array of randomly generated location pointers). The application will output the generated board, "jump" between the squares starting at (0,0) and then output the final square to the console.

Files Location

Under the com.teksystems.bootcamp namespace, create a new package named gridHopper, and add appropriate classes to provide the required functionality.

References:

https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

Instructions:

Create the following classes and methods:

- LocationPointer
    - Contains an (int targetX, int targetY) that indicates an index location on the 2D array
  - int getTargetX
  - int getTargetY
- BoardGenerator
  - static LocationPointer[][] createBoard(int size, int seed);
    - Generates a square 2D array of LocationPointers, using the seed to randomly populate the board with random LocationPointers
    - LocationPointers should randomly range between 0 and (size -1) for both targetX and targetY
    - the same size and seed should always generate the same 2D array and set of LocationPointers
- BoardHopper
  - static LocationPointer hop(LocationPointer[][] board)
    - starts at 0,0, reads the LocationPointer, and "jumps" to the indicated array index
    - returns the LocationPointer from the final position
    - stops when the LocationPointer in the current grid position points to itself, or after 100 jumps

Only main() should interact with the user or write to the console.

Include unit tests for concrete classes.

Exercise 5: Expression Solver

Description

In this exercise you will create a Java application that solves a user-provided mathematical expression.

Files Location

Under the com.teksystems.bootcamp namespace, create a new package named expressionSolver, and add appropriate classes to provide the following functionality.

Instructions

Write a program that will solve a mathematical expression using a class-based solution, and that accepts the following characters:

- numbers 0-9
- math symbols * / + -
- modulo: **%**
- exponent: **^**

When the user enters an expression and presses Enter, the program should output the solution result to the console, solved based on precedence of mathematical operation, which is:

- exponents
- multiplication, division, and modulo from left to right
- addition and subtraction from left to right

ex: user enters 1 + 4 * 7 - 3 / 4 * 2, the console might (specific format is not required) output:

> *solving 1 + 4 * 7 - 3 / 4 * 2*
>
> *The solution is 27.5*

When the user enters an invalid expression, the program should print an appropriate error message.

When building the program, create the following classes and methods. These classes are based on the parts of a mathematical expression.

- Term: a component of the expression, either a constant, or another expression
  - public abstract double getValue();
- Constant (extends Term): a numerical value
  - constructor takes a double

- Expression (extends Term): consists of 2 Terms, and an Operation
  - constructor takes a Term, an Operation, and another Term
- Operation: a mathematical operation
  - public abstract double calculate(double val1, double val2);
- Derived operation classes:
  - AdditionOperation
  - SubstractionOperation
  - MultiplicationOperation
  - DivisionOperation
  - ModuloOperation
  - ExponentOperation
- ExpressionParser
  - public static Term parseExpression(String input);
    - returns a new Constant if there is no operation in the input
    - returns a new Expression, using parseExpression() to populate the Terms

Only main() should interact with the user or write to the console.

Include unit tests for concrete classes.