CST8116 Lab Exercise 03 (22S)

# Instructions

* The five parts of the Software Development Process as presented by Cay Horstmann [1] will be used as the basis for this lab exercise.

1) Understand the problem

2) Develop and Describe an Algorithm

3) Test Algorithm with Simple Inputs

4) Translate the Algorithm into Java

5) Compile and Test Your Program

# What is the problem to solve for this exercise?

* “A school would like educational software to help students check their answers to math problems related to right angle triangles. The program should allow the student to enter the adjacent and opposite sides of the right-angle triangle, and output a summary of the triangle listing the lengths of the 3 sides, as well as the length of the hypotenuse, perimeter, and surface area. All number outputs should be formatted to 4 decimal places.”
* The programming company you work for has a requirement that the initial prototype should be object oriented with a customized class with constructors, accessors, mutators (gets, sets) and at least one (or more) worker method(s) to generate results or perform calculations.
* The first class RightAngleTriangle has had the design created by another programmer, you need to implement the class and then created a second class, e.g. Exercise03 with method main to complete the program.
* For this prototype, the name of the author of the program needs to be output on screen as well.

# Part 1 Understand the Problem

* Examine the Detailed UML Class diagram, and provide an outline for methods calculateHypotenuse(), calculatePerimeter(), calculateArea(), and createReport(). Include an example of each math calculation as well.

Reference: Wikipedia.org (January 15, 2022). Right triangle. Last Accessed Jan 30, 2022. Retrieved from

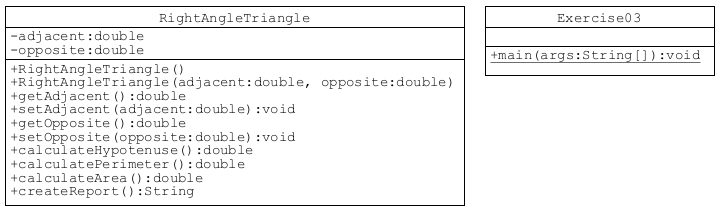
<https://en.wikipedia.org/wiki/Right_triangle>

See sample formulas for hypotenuse and area from that website, cite your reference in your source code.

* remember that the perimeter is the sum of the three sides.

Tip: The method that calculates the Perimeter, and the method that generates a String representation of the RightAngleTriangle both need the hypotenuse. Call the calculateHypotenuse() method inside of these methods, rather than re-creating the logic for calculating the hypotenuse.

# Detailed UML Class Diagrams (See Appendix for plain-text versions)



# Part 2a Pseudocode

* Write pseudocode to design and document these methods of class RightAngleTriangle
  + calculateHypotenuse()
  + calculatePerimeter()
  + calculateArea()
  + createReport()

You can provide a comment or note in your write up that the field values used in these class members are class-level variables as per the UML Class diagram.

* Write pseudocode for method main of the main program:
  + Prompt the user for inputs
  + Place the data into an instance of your class
  + Provide outputs resulting from using the worker methods.

## Part 2b Flowchart

* Create flowcharts to design and document these methods of class RightAngleTriangle
  + calculateHypotenuse()
  + calculatePerimeter()
  + calculateArea()
  + createReport()
* Create a flowchart for method main which:
  + Prompt the user for inputs
  + Place the data into an instance of your class
  + Provide outputs resulting from using the worker methods.
* Place these into your MS Word document as images.

For pseucode and flowcharts, your mathematics will involve calculating a power, and a square root. Rather than write it out like math, e.g. x2 or √42 use methods instead e.g.

power(x, 2) // x raised to 2

squareRoot(42)

Provide a comment that the power method and squareRoot method would be taken from the programming language API, you will likely use the Math class with a method for calculating the power of a number, e.g. 42 and the square root (see the API documentation for class Math and review method pow and sqrt).

Alternatively you can use Math.pow(x, 2) and Math.sqrt(42) in the pseucode, and flowchart but document with a note or comment that these come from the Java API Math class.

Method createReport() needs to concatenate a string similar to:” adjacent 2.0000, opposite 3.0000, hypotenuse 3.6056” using the field varaibles, and by calling method calculateHypotenuse() for the third value.

* In pseucode / flowchart this is simple concatenation with + and a note that output needs to be formatted to 4 decimal places.
* In Java, for your program, reference Hybrid 04 on how to use String.format to build a String which will be returned.

# Part 3 Test Plan for Algorithms

* Create a table in your MS Word document with headings input, expected output, actual output, and description.
* This test plan is testing the logic of method main and can be used as a starting point.
* What happens if a negative value is used? Document the output using a calculator (you are not required to prevent negative values). Use different values for your own testing table.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Actual Output | Description |
| 2.0  3.0 | adjacent 2.0000, opposite 3.0000, hypotenuse 3.6056  Hypotenuse: 3.6056  Perimeter: 8.6056  Area: 3.0000  Program by *your name* | adjacent 2.0000, opposite 3.0000, hypotenuse 3.6056  Hypotenuse: 3.6056  Perimeter: 8.6056  Area: 3.0000  Program by *your name* | A hand trace of the logic produced the correct results. |
|  |  |  |  |

# Part 4 Translate the Algorithm into Java

* Create an Exercise03 project in Eclipse
* Create the requested classes, and use your pseudocode and flowcharts to guide creation of the methods.

# Part 5 Compile and Test Your Program

* Compile and run your program.
* Be prepared to demonstrate your program in the lab period, in the week before the formal lab submission is due.
* Re-use your testing plan and test your program code, note that you are not required to round outputs and can use the default output formatting.
  + Test and document invalid input like negative values, and text-input in addition to testing and documenting valid data.

# Lab Demonstration Notes

* Your lab professor will ask you to demonstrate your program in lab, typically the week before the formal submission
* Your lab professor may also ask you a brief question on your code.

# Microsoft Word Document Format

See the template example (from lab exercise 1) and use the suggested headings below:

* Understand the Problem
* Pseudocode(s)
* Flowchart(s)
* Test Plan Algorithm
* Test Plan Program

# Submission Requirements

* You will need to submit your MS Word document by the due date as specified in Brightspace.
* Follow your lab professor’s submission guidelines.

# Grading (8 Points)

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Missing / Incorrect (0) | Below Expectations (0.5) | Meets Expectations (1) |
| Understand the Problem | Missing or incorrect. | Partly correct. | Briefly outlines the necessary steps, in order, as an overview. Has example of math needed to process input into output, if applicable to the problem statement. |
| Pseudocode(s) | Missing or incorrect. | Partly correct. | Correct format, steps are in correct sequence and lead to correct outputs. |
| Flowchart(s) | Missing or incorrect. | Partly correct. | Correct format, correct shapes used, steps are in correct sequence, matching pseudocode and lead to correct outputs. |
| Test Plan for Pseudocode and Flowchart | Missing or incorrect. | Partly correct. | Has correct format as shown in the lab handout, has test values and expected and actual outputs. |
| Demo in lab period | Missing or student could not answer any questions correctly. | Student program may not compile or run correctly, student partly answers question(s) related to their program code, or answer(s) are partly correct. | Student program compiles, and runs correctly, student correctly answers question(s) related to their program code. |
| Source Code: \*.java file(s) Comments and Conventions | Missing or poorly done. | Missing a comment-header from one or more of class declaration and / or method main declaration. Code loosely follows Java coding conventions for identifiers, indentation. | File comment header with student full name is present. Class and method declarations have comment headers. Code closely follows Java coding conventions for identifiers, indentation. |
| Source Code:  \*.java file(s) program structure and logic. | Missing or poorly done or program does not follow from the pseudocode, and flowchart(s). | Program may have small syntax mistakes and will not compile, and / or produces incorrect output when run. Program loosely follows the student’s pseudocode and flowchart(s). | Program has correct syntax and program logic that produces correct output. Program closely follows the student’s pseudocode and flowchart(s). |
| Test Plan for Program | Missing or poorly done or is only an unchanged copy of the provided algorithm test plan. | May not have correct format, does not verify that the program outputs match expectations. | Has correct format as shown in the lab handout, verifies that the program outputs match, and documents variations in output including samples of invalid inputs. |

# References

[1] Cay Horstmann. (2019). Big Java Early Objects. 7th Ed. Wiley.

[2] Joyce Farrell. (2018). Programming Logic & Design Comprehensive. 9th Ed. Cengage Learning.

# Appendix Sample of Program Run

Right angle triangle program

Enter adjacent length: **2**

Enter opposite length: **3**

adjacent 2.0000, opposite 3.0000, hypotenuse 3.6056

Hypotenuse: 3.6056

Perimeter: 8.6056

Area: 3.0000

Program by Stanley Pieda

The user inputs were formatted as black-color with bold-font with yellow highlighting in this document, the default color used in Eclipse for user input is a light-green.

# Appendix Plain Text UML Class Diagrams via UMLet

## Class RightAngleTriangle

RightAngleTriangle

--

-adjacent:double

-opposite:double

--

+RightAngleTriangle()

+RightAngleTriangle(adjacent:double, opposite:double)

+getAdjacent():double

+setAdjacent(adjacent:double):void

+getOpposite():double

+setOpposite(opposite:double):void

+calculateHypotenuse():double

+calculatePerimeter():double

+calculateArea():double

+createReport():String

## Class Exercise03

Exercise03

--

--

\_+main(args:String[]):void\_