CST8116 Lab Exercise 05 (22S)

# Instructions

* The five parts of the Software Development Process as presented by Cay Horstmann [1] will be followed 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

* Also refer to your textbook by Joyce Farrell [2] for help with pseucode, flowcharts for loops.

# What is the problem to solve for this exercise?

* Assume that in the context of a work-place another programmer had started working on this project, and did not complete it before leaving the company to start a new job elsewhere.
* This other programmer completed the typical understand the problem, pseudocode, and partly completed some trace tables (See the companion file). They also created some starter code which is incomplete (see provided class files).
* You are to create flowcharts for the three specified methods, you can use the existing pseudocode as a guide.
* Complete the trace tables which examine the loop processing.
* Finish the Java programs, compile and take screen shot(s) of the running program.
  + **You are not permitted to add additional methods, or otherwise change the structure of the program. You must work within the confines of the three specified methods.**
* Copy and paste your trace tables, and update the notes column to document that the program was tested and formats output to three decimal places as requested.

# Part 1 Understand the Problem

* See the source code and the companion file, this has been provided to you.

# Part 2a Pseudocode

* This has been provided to you, see the companion file.

# Part 2b Flowchart

* Write flowcharts for each of the three methods requested in the companion file.
* You may follow and / or copy from the provided pseudocode.
  + Tip: Copy and paste the pseucode into a simple text editor like Notepad first, to remove formatting, then copy and paste into your flowchart software.
* Note: For loops are diagrammed like a while loop for the purposes of flowcharts.

# Part 3 Test Plans for Algorithms

* Complete the provided trace tables and alter the test values to hand-trace through your flowcharts.
* Reference hand-trace-tables as per Cay Horstmann Section 6.2 (pages 179 to 182)
* Suggestion: Keep the tests short, i.e. no need to use guess input values of 1 and document 27 of these, verify that the guess fuel counts down.
* Tip: What happens if the user enters zero or a negative number as their guess? There may be a problem with the logic of the program, note that you are not asked to fix the problem only to identify it.

# Part 4 Translate the Algorithm into Java

* Create an Exercise05 project in Eclipse
* Using the starter code (copy the files into your src folder in Eclipse):
  + Update all of the comment sections as per the requirements of our course, see week 1 lecture notes
  + Use the provided pseudocode or your flowcharts and complete the three methods.

# Part 5 Compile and Test Your Program

* **Run your program and take a screen shot of the running program, after it has completed a run.**
  + **Exercise 05 does not have a demonstration in lab, your screen shot will be used instead of the demonstration. Make sure your program output contains your full name, which is also visible in the screen shot.**
* Follow your hand trace tables using your documented inputs and screen shot the completed program.
* Re-use your algorithm test plans with the debugging tools and verify your program code.
  + Set a break point just before each loop and step into each line observing the loop control variable as it changes, using the debugger like this can be used in conjunction with hand-traces.
  + Tip: Step over any method calls so you can focus on the loop itself.

# Question(s)

1. (Reflective) Examine method reset() and re-write the for loop as a while loop using pseucode and submit this as part of your write up in the MS Word document.
   * Which is your preferred version of the loop, and why? (Compare and contrast from perspective of ease of use and ease of reading between a for-loop, and a while-loop that both solve the same problem).
2. (Technical) When using a for loop the loop control variable is written in-line within the loop header.
   * What is the scope of this variable (class-scope, local-scope, or block-level-scope)?
   * Can you access the loop control variable to print the last value it held below the loop body (yes/no)?
   * You may answer these two questions by selecting the word or phrase that answers the question.

# Microsoft Word Document Format

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

Flowchart(s) for each problem method

Test Plan Algorithm – Hand Traces

**Screen Shot of Program Execution**

Test Plan Program – Hand Traces

Question Answers

# Submission Requirements

* You will need to submit your MS Word document and .java code files 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 (1.0) |
| Flowchart(s) | Missing or incorrect. | Partly correct. | Correct format, correct shapes used, steps are in correct sequence, matching pseudocode and lead to correct outputs.  Demonstrates while loop, do-while loop, for loop. |
| Test Plan for Pseudocode and Flowchart (Hand Traces) | Missing or incorrect or exactly the same as provided in the handout. | Partly correct. | Has correct format as shown in the lab handout, documents the state of variables used inside of the loop, as well as loop control variables and exiting the loop. |
| **Screen shot executing program** | Missing or incorrect or student name not part of program output. | Screen shot shows student name but program might not have compiled or run properly. | Screen shot shows student full name as in ACSIS. Program runs properly and valid input(s) and expected output(s) are shown. E.g. demonstrates continuation of indeterminate loops, and output from determinate loop. |
| 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. This includes Java conventions for indentation of nested selection structure(s). |
| 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). Program logic for repetition structure(s) is correct. |
| Test Plan for Program (hand-trace) | Missing or poorly done or is only an unchanged copy of the handout sample test plans or unchanged copy of the algorithm test plan created by student. | May not have correct format, does not correctly document a trace of the logic of the repetition structures in use. | Has correct format as shown in the lab handout, does correctly document a trace of the logic of the repetition structure including exit condition. |
| Reflective Question | Missing or poorly done. | Partly correct. | Student compares a while loop to a for loop with regards to ease of use and readability. |
| Technical Question | Missing or poorly done. | Partly correct. | Fully correct. |

# References

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

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