CSCI 2134 Robustness and Conciseness Code Review Checklist

|  |  |  |  |
| --- | --- | --- | --- |
| Reviewers: | Lynda Ofume- B00738568 | | Taylor MacIntyre- B00752864 |
| Kiera Flint- B00727706 | |  |
| Date: | February 25, 2020 | Project: | Lab 5 |
| Files: | TimSim.java | | |

**Rating**: (**A**) Inconsistent with goals, but no effect on operations (**B**) Minor effect if not addressed (**C**)Major effect if not addressed (really bad)

|  |  |  |
| --- | --- | --- |
| **Input Validation** | Rating | Locations if present (File/Line #) |
| * Missing input validation on user input | C | LINE 34-39, 49-52- TimSim.java |
| * Missing parameter validation | B | LINE 42- TimSim.java |
| * Missing file or database input validation | N/A | N/A |
| * Missing check on input size | A | LINE 48- TimSim.java |
| * Missing check on input data ranges | A | LINE 34-35- TimSim.java |
| **Error Handling** | Rating | Locations if present (File/Line #) |
| * Method can fail but doesn’t notify caller | N/A | N/A |
| * Method returns non-descriptive error information | C | LINE 25- TimSim.java |
| * Ignored return code or exception | N/A | N/A- TimSim.java |
| * Empty catch block | N/A | N/A- TimSim.java |
| **Offensive Programming** | Rating | Locations if present (File/Line #) |
| * Error condition would be better handled with an assertion | C | LINE 55-67, LINE 71-75- TimSim.java |
| * Special case code that could be handled by the regular algorithm | A |  |
| **Code Replication** | Rating | Locations if present (File/Line #) |
| * Similar code cases in one method | A | N/A- TimSim.java |
| * Similar code to handle cases across methods | A | N/A- TimSim.java |
| * Expression in conditional repeated in the same expression | N/A | N/A- TimSim.java |
| * Method or code block too long or too complex | A | N/A- TimSim.java |
| * Changing one aspect of operations would appear to need many inter-related modifications | C | ENTIRE CLASS- TimSim.java |
| * Classes are overly dependent on one another | C | LINE 25-118- TimSim.java |
| **Efficiency** | Rating | Locations if present (File/Line #) |
| * Inefficient conditional or loop condition | A | N/A- TimSim.java |
| * Needlessly inefficient logic | A | N/A- TimSim.java |

# Assumptions or general observations:

|  |
| --- |
| 1. OVERLY DEPENDENT ON ONE ANOTHER: The way to mediate the code base would be to make sure that all classes for the program are running properly. 2. NO INPUT VALIDATION: Before we run the code make sure that we use “if” statements to make sure that the input valid 3. NO ASSERTIONS: Using assertions, catch statements, or return code will allow for the developer debugging to be able to catch where the program and/or class fails  * Overall the code is pretty effective, though the lack of input validation and the assertions allow for the code to crash and there is nothing to catch where the code fails while debugging. |

# Comments:

Briefly list the top three issues with the codebase being reviewed and describe what remediations you would recommend.