ICS3U Summative – Game Design

Instructions:
Design a text based strategy game using Java that follows a famous board game style but that has a set of newly defined rules created by you (the programmer)

See list of potential games to model/build off website.

Program requirements:
1. All work must be clearly documented (Mindomo and Dr. Java (Not Repl), Gantt chart, GitHub Classroom)
2. All code must be well documented (comments) and neatly formatted as per conventions (i.e. variable naming conventions).
3. The program should be organized using Methods and fully detailed in (Mindomo and Dr. Java).
4. You must maintain a Mindomo chart plan showing how you organized the program (using a flow chart diagram).
5. **The program should make use of all programming concepts covered in the course. Objects may NOT be used without prior approval (i.e. No ArrayLists).**
6. The program should be as crash-proof as possible (Use Try-Catch for Level 4 – Don't assume anything)
7. You must submit a test plan and test data which demonstrates how you checked for stability.

Grading:
• 15% of your final mark will consist of the initial game design incl. external documentation (i.e. Gantt Chart, Flow chart diagram in Mindomo, pseudocode, GitHub ongoing code maintenance, ongoing planning.) This will be included in your term mark.
• The internal documentation, code and quality of this summative will count for approximately 85% of your final summative mark and will be assessed through multiple means (i.e. interview, digital submission on GitHub, Mindomo documentation)
Your mark will reflect the way in which you have integrated the programming concepts that you have learned in this course. (eg. arrays, subroutines (methods), repetition, selection, string methods)
• Your final mark will also be adjusted to reflect the level of the task you have taken on. (i.e. Higher level for properly implementing a difficult concept or difficult new rule or developed random/skilled computer players – See Mr. Linseman if this is not feasible)

The final overall grade is based on your classroom production which will be monitored.

Interviews: Interviews **may** be used to properly assess your understanding of the parts of your program. During the interview(s), which will take place throughout or towards the end of the semester in class, be prepared to fully explain all parts of your program's code. If you are unable to explain how each portion of your game code works, this will negatively affect the overall grade. Therefore, do not simply plagiarize programs already out there or a mark of **ZERO** will result!

Also, during the interview, be prepared for **sample questions** pertaining to structures within your code (e.g. Arrays, Methods, Variable choices, Selection statements, Organization schemes, Error handling)

A sample question during an interview could also be more general such as the following:
"Write down an example of a method that passes a 1-D integer array and returns a Boolean depending on the sum of the array elements."

The Rubric below will be used by the teacher to grade the 85% portion of the summative.

	Achievement	Insufficient				Limited			Approaching			Sufficient			Insightful			
Expectation		**I**	**-**	**R**	**+**	**-**	**1**	**+**	**-**	**2**	**+**	**-**	**3**	**+**	**-**	**4**	**+**	**++**
A1	**Uses ALL basic data types (String, int, float/double, boolean, char)**																	
	using assignment statements correctly																	
	arithmetic & Boolean operations, ordering																	
	One (or Two) dimensional arrays (3U requires 1-D only**)**																	
A2	**Sequencing & Selection** (incl. Efficient design of if-else-else if)																	
	Incorporate proper IPO in program																	
	Repetition (for loops, while & do while, incl. nesting)																	
A3	use existing subprograms in API (i.e. Math.sqrt()) as well as implement use of **String Methods**																	
	Write subprograms (Methods)																	
	parameter passing, appropriate scope																	
A4	error detection & correction – Able to interpret compiler errors independently																	
	Applies proper conventions for naming, indenting, commenting & internal documentation																	
	test & validate using a full range of test cases (Crash proof) Level 4: Try-Catch																	
B	**Software Development (15%)**																	

Note: Expectation B is based on all spot checks along the way and your ability to use the planning tools provided to organize your computer program and submit all parts appropriately. It is incorporated into the term mark.

Teacher Comments: See GitHub Classroom Readme File.

Overall:

Code for Summative (Level for portion out of 85%): _______

Summative Planning (Level for portion out of 15% - Part of Term): _____