Assignment 2 – Detailed Marking RUBRIC Version 2.0

Part	Description	Marks	FAIL <40%	PASS 40-49%	GOOD 50-59%	V.GOOD 60-69%	Excellent 70+%
	OVERALL						
A1	Inclusion of completed Readme	(1)	Completed properly and included in submission. Name, student number, brief description of the game, **how it works (keys)**, **problems. Nothing else!**				
A2	Inclusion of completed Reflection	(5)	0 Not included. Correcting Terminated * Interview student	2 Minimal reflection Marks filled in	3 Some items not filled in correctly		4,5 **Line numbers** and examples provided for everything, with marks and commentary
A3	Inclusion of completed Video structured as specified	(5)	No video Correcting Terminated * Interview student	2 Video > 10mins supplied explaining some of the game. Prescribed Section missing or video <5 mins * Interview student			4,5 3 sections in the video 10 mins or less explains clearly all the code relating to key elements of the game.
B1	Comments	(1)	Examples of **both** line comments // and multiline /* */				
B2	indentation	(1)	Either manually indented correctly or **autoformatted in ALL files. Tidy up your code!**				
B3	naming	(1)	Meaningful **names** of variables and methods, camelCase, action words for methods, etc				
B4	structure of code	(1)	Main program, **multiple Class files**, readme, reflection.				
C	Use of standard programming control constructs (if, loop, nesting)	(5)	0 Doesn't meet the pass criteria	2 Examples highlighted showing **single** example If (switch acceptable too) AND Loops: at least one example of 2 different loop types using {while, for, do while} loops	3 Examples highlighted showing **multiple** examples of each of the following: If (switch acceptable too) Loops: **at least 2** of the following types included in the examples {while, for, do while}		4,5 highlighted showing **multiple** examples of each of the following: Nesting If (switch acceptable too) Loops: while, for, do while full marks for highlighting multiple examples of **all**
D1	**Working game**	(10)	Doesn't have feature of a working game	4 • Scoring • Lives	5 • High score table	6 • New Tournament	• 7+ • Multiplayer • New game (resets correctly)
D2	**Handling key events**,	(10)	0 Doesn't. Mouse only or nothing	4 Similar to pong Left/right			7 Working key handler Up down Left right 8 For 2 more 9 for 1 more 10 for 1 more
D3	**collision detection** /avoidance).	(10)	0 Doesn't meet the pass criteria	4 collision detection between 2 objects created from your classes (NOT mouse)	5 Edge conditions of the Game Board are handled		7 collision detection with 3 objects 10 collision detection with 4 + objects

Assignment 2 – Detailed Marking RUBRIC Version 2.0

E	**User-defined class**es (at least 2 classes **excluding Player class** from Pong):						
E1	**This is referring to the additional tabs defining classes i.e. CLASS DEFINITIONS** List the classes you've created (i.e. the tabs) and for each class detail (with line numbers) - fields - constructors - getters - setters - overloaded constructors	(10)	0 Not included or not working.	4 Just reusing (Ball/Paddle) Without getters & setters	5 Just reusing (Ball/Paddle) With getters & setters	6 For 2+ classes Only has - fields - constructors - getters - setters **Missing examples of overloaded constructor**	7 3+ classes, used in the game (not including Player), showing: fields, constructors, getters, setters, **overloaded constructor** -- 8-10 Additional marks for each additional class highlighting same
E2	- Class bespoke methods: no return value, return value, parameters, **overloading** **THESE EXAMPLES MUST ALL BE DIFFERENT TO THOSE PROVIDED IN E1**	(10)	0 Not included or not working. Example methods from main program provided - incorrectly	4 Just shows examples of class bespoke methods Example methods from main program will not considered	5 Criteria for V.Good but without examples of Returning a value	6 Criteria for excellent but without overloading examples	7-10 Multiple methods, at least 4 highlighted showing at least 1 of each of the following: Demonstrating No return value AND accepting parameters AND returning values AND Overloading more marks given for more examples of each
E3	- The user-defined class(es) are **used** in an appropriate manner This is how you have **used** the classes **by using object variables or object arrays, created in the main program**. It should **respect encapsulation rules** despite any shortcomings of processing. *(Start at the excellent level and work your way left)*	(10)	0 Game is written as a single program rather than taking an object oriented approach.	4 ignoring encapsulation rules	5 Criteria for v.good but **without:** overloading examples and/or calling of getters & setters	6 Criteria for excellent but **without:** overloading examples	10 Examples highlighting clearly: **Declaration** **Initialisation** Calling the **getters & setters** Calling an **overloaded constructor** These can be for an ordinary variable or an **array** variable
F	**Data structure:**						
F1	- Use of arrays (primitive/object) to store information	(10)	0-3 Not included or not working.	4 1 array of any primitive type with a fixed size setup and used	5 array of any non-primitive (i.e. object) type with a fixed size, values computed (e.g. random position) (e.g. grid position)	6 Array of any non-primitive (i.e. object) type, set to a fixed size, values filled by user input (e.g. filled from JOptionPane)	7 Array **size that is specified by some user input**, and **values filled by some user input** **e.g. number of balls** **8+** Includes **multiple array examples** for additional marks.
F2	- Calculations performed on the information stored (e.g. min, max, average, find) *(Use a loop to visit each element in the array and do something)*	(10)	0-3 Not included or not working.	4 Elements of the array are statically addressed by index **(no loop)**	5 Elements are looped through, using the known array length (**hard coded or variable**) Values used to perform check or calculation	6 element in the array is checked as part of a loop using .**length** each value is **compared** to some criteria no **updating of the array element**	7+ element in the array is checked as part of a loop using .**length** each value is **compared** to some criteria value of an array element is **updated**
	TOTAL MARK	(100)					
	COMPLEXITY MULTIPLIER *(subject to change)*		Basic *.8 Basic game missing stuff	Intermediate *.9 working game with lives		Advanced *1.0 fully functioning game with new game, lives, and scoring,	
	OVERALL PRE-INTERVIEW MARK						