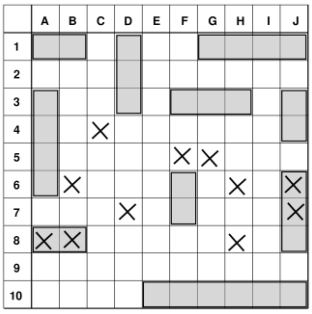


Battle Ship "With a Twist"

Introduction

This assignment is due by 9 am Sunday of Week 12 (21st October, 2018). It is worth a total of 25% of the marks for your final assessment in this unit. **Heavy penalties will apply for late submission.** This is an individual assignment and must be your own work. You must attribute the source of any part of your code which you have not written yourself. Please note the section on plagiarism in this document.

This assignment is structured as a 20% + 5% assessment. The code submitted for this assignment will be assessed for 20% of the marks and a unit test during the tutorial time in Week 12 will be assessed independently for 5% of the marks allocated. Please note that the unit test is a hurdle requirement for the assignment. Failing to complete the unit test will result in no marks being allocated for the entire assignment.

The assignment must be done using the BlueJ environment.

The Java source code for this assignment must be implemented according to the **Java Coding Standards for this unit.**

Any points needing clarification may be discussed with your tutor in the lab classes.

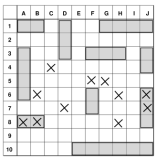
Specification

For this assignment you will write a program which will allow a person to play a game called Battleship with a Twist. While many variations of this game are present, the one being implemented for this assignment is a specific variation to ensure all beginner programmers can achieve this outcome.

This section specifies the required functionality of the program. **Only a text interface is required for this program**; however, more marks will be gained for a game that is easy to follow with clear information/error messages to the player.

The aim of *Battleship with a Twist* is for a player to try and destroy all of the computer opponents ships before the computer destroys all of the players ships placed on the playing grid.

The game will allow the player to place a number of ships onto the playing grid based on x and y coordinates entered by the user. The computer opponent will also do the same randomly. Each ship



will only occupy a single coordinate, as such, two ships cannot be placed at the same set of coordinates. To keep things interesting, each ship when places is given a random number between 1 – 5 which indicates its overall strength. This means that the same ship has to be hit that many times in order for it to be destroyed. Once a ship has been destroyed it is taken out of play on the grid. Each hit on a ship get points for the player or the computer. The game ends when all the ships are destroyed.

Game play

The following section outlines the game play for each of the involved participants.

```
Beginning Round 1
Player Score: 0
Computer Score: 0
Displaying the Player Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~0~~~~~~~~~~0
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~0~~~~~~~
~~~~~~~~~~~~~~~~

---------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~


Player to make a guess
Ship x Position (0 - 14):
```

Round Number

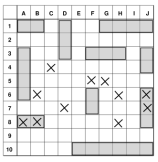
Player and Computer Score

Player Grid

Player Ship

Computer Grid

Computer Ships Hidden



The Playing Grid

The playing grid will be of a fixed size for the duration of the game. On starting the game, the game will load the following preferences from a file called "gamesettings.txt".

Grid Width and Height: 5
Multiple Hits Allowed: true
Computer Ships Visible: false
No of Ships : 3

The file will represent this default data as follows: 5,true,false,3

Your program must work based on the parameters specified in this file. (Note: default values not used)

```
+=============================================================================+
|                                                                             |
|              Welcome to the Battleship Game -- With a Twist!!               |
|                                                                             |
+=============================================================================+
The game will use the grid size defined in the settings file.
Playing grid size set as (15 X 15)
Maximum number of ships allowed as 3
Multiple hits allowed per ships set as true
Computer Ships Visible : ON
Press any key to continue...
```

Manoeuvring the Grid:

To move the various elements along the grid, the grid is to be represented as coordinates. Each position on the grid is given a coordinate in the format x,y.

No two ships can occupy the SAME coordinate at any given time.

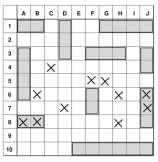
This is only applicable to each player. For e.g. the player can place a ship at 2,3 and the computer can place its ship at 2,3. But both the player and computer cannot place another ship at the same coordinate.

The Grid

The grid will use the following notation:

~ – to indicate water and an unoccupied grid. The computer ships until hit will also display this symbol.
O – to indicate a player ship to the player if one is placed at those coordiantes
D – a ship which has been hit at least once to indicate a damaged ship.
X – a destroyed ship.

The computer's ships MUST be made visible or be hidden from the player based on the flag set in the settings file above. When the computer ships are hidden, the D and X status of each ship MUST be shown on screen when hit or destroyed. However undamaged ships are shown as ~ symbol.



Game Setup:

After loading the game settings from the file, the system will then ask the user to enter the following information for each ship:

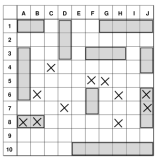
1. Ship Name – String between 3 and 15 characters long.
2. x Coordinate – X coordinate position between 0 and the maximum grid size from the settings file.
3. y Coordinate – y coordinate position between 0 and the maximum grid size from the settings file
4. If any coordinate is incorrect (e.g. alphabetic or outside range of grid) an error message should be displayed.

```
Loading player settings:
Please enter the details for the 1 ship:
ShipName:
Colorado
Ship x Position (0 - 14):
13
Ship y Position (0 - 14):
5
Please enter the details for the 2 ship:
ShipName:
Eisenhover
Ship x Position (0 - 14):
13
Ship y Position (0 - 14):
9
Please enter the details for the 3 ship:
ShipName:
Stockholm
Ship x Position (0 - 14):
abc
Ship x Position Must Be Numeric
Ship x Position (0 - 14):
14
Ship y Position (0 - 14):
18
Ship y Position Must Be between 0 and 14
Ship y Position (0 - 14):
4


Loading computer settings:
Computer settings generated!
Press any key to continue...
```

Coordinate Validations

The number of ships set is determined from the game settings file.

After the player ships are set, the computer ships are automatically generated.



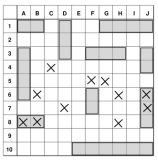
Turn Based Game Play:

Note: For demo, default grid size is not used, and enemy ships are shown on the screen.

Turn 1: The player is asked to make a guess, the system will then check if the player hits a computer ships or misses completely. The system will then display the appropriate message to the user.

```
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~0~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~

-----------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~0~~~~~~~~~
~~~~~~~~~~~~~~~~0
~~~~~~~~~~~~~~~~~
~~~~~~~~~~0~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~


Player to make a guess
Ship x Position (0 - 14):
4
Ship y Position (0 - 14):
13
PLAYER MISSSSS!!!!
Computer to make a guess

Computer x guess: 7
Computer y guess: 6

COMPUTER MISSSSS!!!!
Press any key to continue...
```

After the players guess, the computer is asked to make a guess, and the system checks to see if the computer makes a hit or misses the player ships.



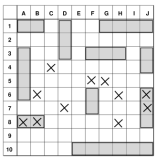
Turn 2:

The player is again asked to continue making a guess at the coordinates. Appropriate validations must be done as before for the entered coordinates.

```
-------------------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~0~~~~~~~~
~~~~~~~~~~~~~~~0
~~~~~~~~~~~~~~~~
~~~~~~~~~~0~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~


Player to make a guess
Ship x Position (0 - 14):
abc
Ship x Position Must Be Numeric
Ship x Position (0 - 14):
16
Ship x Position Must Be between 0 and 14
Ship x Position (0 - 14):
4
Ship y Position (0 - 14):
12
PLAYER MISSSSS!!!!
Computer to make a guess

Computer x guess: 8
Computer y guess: 8

COMPUTER MISSSSS!!!!
Press any key to continue...
```

Coordinate Validations

After the player guess, the computer is asked to make a guess and the system validates the guess to check for a hit or a miss.



Turn 3:

The player makes a hit.

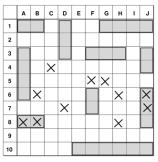
```
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~


Player to make a guess
Ship x Position (0 - 14):
14
Ship y Position (0 - 14):
5
PLAYER HITTTTT!!!!
Computer to make a guess

Computer x guess: 8
Computer y guess: 11

COMPUTER MISSSSS!!!!
Press any key to continue...
```

The system displays the appropriate message on the screen and at the start of the next turn, the grid layout is changed as follows:

```
Beginning Round 5
Player Score: 10
Computer Score: 0
ENEMY SHIPS ONLY SHOWN WHEN DEMO MODE IS  ON!!
Displaying the Player Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~O
~~~~~~~~~~~~~~O~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~O~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

------------------------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~O~~~~~~~~
~~~~~~~~~~~~~~~D
~~~~~~~~~~~~~~~~
~~~~~~~~~O~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
```

Scores are updated
Each hit is 10 points

Ship status changed to D



Turn x:

The player finally destroys an enemy ship.

```
Beginning Round 9
Player Score: 50
Computer Score: 0
ENEMY SHIPS ONLY SHOWN WHEN DEMO MODE IS  ON!!
Displaying the Player Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~O
~~~~~~~~~~~~~~O~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~O~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

------------------------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~O~~~~~~~~
~~~~~~~~~~~~~~~X
~~~~~~~~~~~~~~~~
~~~~~~~~~~O~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

Unfortunately, Comp Ship 2 has been destroyed!
```

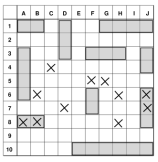
Score indicates that ship had a strength of 5.

Ship status changed to X

Message to state the ship has been destroyed

Two Ships Destroyed:

```
Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~D~~~~~~~~
~~~~~~~~~~~~~~~X
~~~~~~~~~~~~~~~~
~~~~~~~~~~X~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

Unfortunately, Comp Ship 3 has been destroyed!
Player to make a guess
Ship x Position (0 - 14):
```



All Ships Destoryed:

```
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

--------------------------------------------------

Displaying the Computer Grid
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~D~~~~~~~~
~~~~~~~~~~~~~~~X
~~~~~~~~~~~~~~~~
~~~~~~~~~X~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~

Unfortunately, Comp Ship 3 has been destroyed!
Player to make a guess
Ship x Position (0 - 14):
6
Ship y Position (0 - 14):
4
PLAYER HITTTTT!!!!
Computer to make a guess

Computer x guess: 14
Computer y guess: 1

COMPUTER MISSSSS!!!!
Press any key to continue...

Congratulations!! Player Wins!!
```

Winning Message

On completing the game, the system will write the outcome to a file called “gameoutcome.txt”.

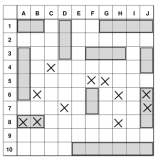
The outcome file will have the following output:

Player wins. Final Score Player (110) and Computer (0)

Game Rules

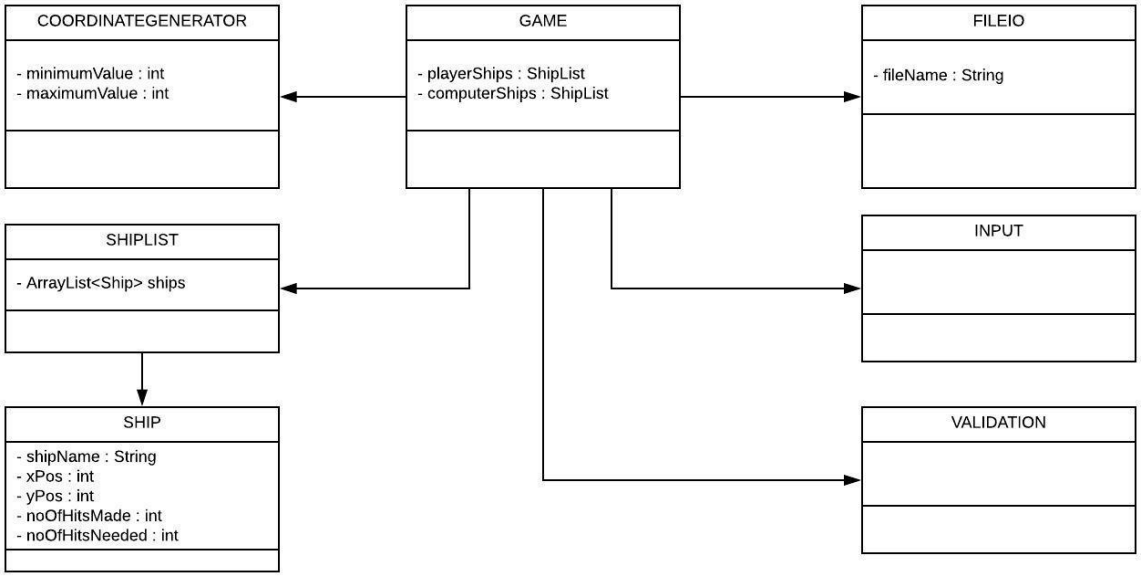
The following game rules MUST be adhered to when playing the game:

- No two ships can occupy the same X coordinate and Y coordinate for each player
- Guesses must be within the specified range
- Each hit gets a score of 10 points
- Each player gets one guess per turn
- The hull strength of each ship is determined randomly between 1 – 5. This indicates the number of hits required for the ship to be destroyed.



Program design

The objective of this assignment is for students to understand coding, as well as to get the basics of design. The following class diagram is **proposed** for this assignment.



Your program **MUST** consist of the following classes:

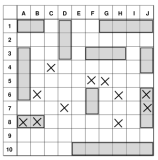
1. Game
2. ShipList
3. CoordinateGenerator
4. Ship

The following classes are desirable and will score marks if implemented. But if object interaction is still challenging, these classes can be integrated into the other 4 classes mentioned above.

5. FileIO
6. Input
7. Validation

You MUST follow good programming practices and use loops where required to ensure good program design.

Design changes to the above must be discussed with your tutor prior to proceeding.



Game class

The Game class will specify the attributes and behaviours of the game. An object of the Game class will have the following fields only:

- *playerShips* – (ShipList) an object of the ShipList class.
- *computerShips* – (ShipList) an object of the ShipList class.

This class is responsible for initiating the game, reading the file, loading the settings, interacting with the other classes, and writing to the file when the game ends. **Reading and Writing to the file must only occur ONCE.**

ShipList class

The ShipList class will specify the attributes and behaviours of all the ships within the game. An object of the ShipList class will have the following fields only:

- *ships* – (ArrayList<Ship>) – stores objects of the class Ship within an ArrayList collection.

This class is responsible for creating an arraylist which stores each ship within the player grid for the respective player or computer.

CoordinateGenerator Class

The CoordinateGenerator class will specify the attributes and behaviours of generating random coordinates which will be used by the Game class. An object of the CoordinateGenerator class will have the following fields only:

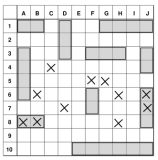
- *minimumValue* – (int) stores the lower limit of the number to be generated.
- *maximumValue* – (int) stores the upper limit of the number to be generated.

This class is responsible for generating a random number which can be used for deciding the X and Y coordinates for each ship and even for deciding the random hull strength of each ship within the game.

Ship Class

The Ship class will specify the attributes and behaviours of all ships within the game. An object of the Ship class will have the following fields only:

- *shipName* – (String) stores the name of the ship between 3 – 15 characters.
- *xPos* – (int) stores the x position of the ship on the grid.
- *yPos* – (int) stores the y position of the ship on the grid.



- *noOfHitsMade* – (int) stores the number of times the ship has been hit.
- *noOfHitsNeeded* – (int) stores the number of times the ship needs to be hit to be destroyed. This number if randomly generated by the game for each ship for both the player and the computer.

This class is responsible for recording a new ship added to the grid.

Additional Classes:

FileIO Class

The FileIO class will specify the attributes and behaviours for reading and writing to a file. An object of the FileIO class will have the following fields only:

- *filename* – (String) the name of the file to be read or written to.

This class is responsible for reading and writing to a file only. The date read should be passed to the calling class to handle.

Input Class

The Input class will specify the attributes and behaviours for reading input from the user via the keyboard. An object of the Input class will have no fields. However, all methods included can be made class methods.

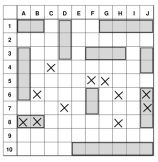
Validation Class

The Validation class will allow the system to validate all user inputs accepted via the keyboard from the user. An object of the Validation class will have no fields.

Hints and Suggestions

The following hints and suggestions may be useful in designing your program:

- To facilitate a clean layout of your game, consider looking at the tutorial notes in the homework exercises on how to clear the terminal screen in BlueJ (Week 4).
- When reading and writing to a file with multiple values, consider looking up the String class for the split method to understand how it can be used to separate multiple values.
- Your program MUST validate string and numeric inputs for this assignment. Consider using exception handling well in order to ensure your program does not crash.
- **YOUR PROGRAM MUST NOT CRASH NOT MATTER WHAT THE USER DOES.**
- File input will always be treated as correct and the input file will not be modified directly.



Test Strategy

For this assignment, you are required to produce and submit a partial Test Strategy for the program.

Your Test Strategy will be only for one class - the Ship class.

There is no need to produce Test Strategy for any other classes you have used in your program.

You must provide a Test Plan, plus detailed sets of Test Data, Expected Results and Actual Results for the student class.

Assessment

Coding Assessment (10%)

Assessment for this assignment will be done via an interview with your tutor. The marks will be allocated as follows:

- 35% - Object-oriented design quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of the object's state. This include marks for the coding standards.
- 10% - Test Strategy.
- 55% - Program functionality in accordance to the requirements.

You must submit your work by the submission deadline on the due date (a late penalty of 20% per day, inclusive of weekends, of the possible marks will apply - up to a maximum of 100%). There will be no extensions - so start working on it early.

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the FIT9131 Java Coding Standards.

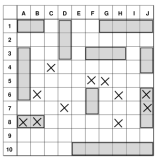
Requests for extensions must be directed to the lecturer with proof and supporting work completed.

Unit Test Assessment (5%)

This assessment will be a written assessment and will be held during the tutorial times in the week of submission (Week 12). **Due to this being the last week of the semester, do keep in mind the unit test will be held PRIOR to the submission date**. Keep in mind that students are expected to have finished ALL their code by the submission deadline, and more importantly have to have done their own work.

The unit test will be a closed book test wherein students will be required to WRITE code to answer a few questions. The questions will cover material from Weeks 1 – 10 of the lectures. Students should bear in mind that BlueJ or any other electronic device will not be permitted, so students need to know HOW to write their own code in order to finish the unit test hurdle.

Unit test for DE / Off-Campus students will be scheduled for Week 12 / SWOT Vac (voluntary), but due to the examination environment of the unit test, an equivalent moodle quiz or a skype assessment may be administered which may be slightly different than the on-campus format of the assessment.



Interview

You will be asked to demonstrate your program at an "interview" following the submission date. At the interview, you will be asked to explain your code/design, modify your code, and discuss your design decisions and alternatives. Marks will not be awarded for any section of code/design/functionality that you cannot explain satisfactorily (the marker may also delete excessive in-code comments before you are asked to explain that code).

In other words, you will be assessed on your understanding of the code, and not on the actual code itself.

For **on-campus students**, interview times will be arranged in the tutorial labs in Week 11. It is your responsibility to attend the lab and arrange an interview time with your tutor. **Any student who does not attend an interview will receive a mark of 0 for the assignment.** The actual interviews will take place during in Week 14 / SWOT Vac (voluntary).

For **off-campus learning (OCL)** students, the interviews will be organised during week 11 and will take place online via Skype or other video facility during Week 14 / SWOT Vac (voluntary). It is your responsibility to make yourself available for an interview time. **Any student who does not attend an interview will receive a mark of 0 for the assignment.**

Submission Requirements

The assignment must be uploaded to Moodle by 9 am Sunday of Week 12 (21st October, 2018).

The submission requirements for Assignment 2 are as follows:

A .zip file uploaded to Moodle containing the following components:

- the BlueJ project you created to implement your assignment.
- a completed **Assignment Cover Sheet**. This will be available for download from the unit's Moodle site before the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.

The .zip file should be named with your Student ID Number. For example, if your id is 12345678, then the file should be named 12345678_A1.zip. Do not name your zip file with any other name.

It is your responsibility to check that your ZIP file contains all the correct files, and is not corrupted, before you submit it. If you tutor cannot open your zip file, or if it does not contain the correct files, you will not be assessed.

Marks will be deducted for any of these requirements that are not complied with.

Plagiarism

Cheating and plagiarism are viewed as serious offences. In cases where cheating has been confirmed, students have been severely penalised, from losing all marks for an assignment, to facing disciplinary action at the Faculty level. Monash has several policies in relation to these offences and it is your responsibility to acquaint yourself with these.

Plagiarism (http://www.policy.monash.edu/policy-bank/academic/education/conduct/plagiarism-policy.html)