Super Fantastic Kiosk

Due Date: 10pm on Friday in Week 7 (23rd April 2021)

Introduction

This assignment is worth **10%** 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 entirely your own work. You must attribute the source of any part of your code which you have not written yourself. ***Your program will be checked with a code similarity detector***. Please note the section on plagiarism in this document.

The assignment must be done using the BlueJ environment.

The Java source code for this assignment must be implemented in accordance with the ***Java Coding Standards*** for this unit.

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

Specification

For this assignment you will write a program that simulates a rather simplistic ***Super Fantastic Kiosk***. This section specifies the required functionality of the program. ***Only a simple text interface (using the BlueJ Terminal Window) is required for this program***; however, more marks will be gained for a program that is easy to follow with clear information/error messages to the player.

The aim of your program is to ***simulate*** a physical kiosk commonly found in shopping malls, where a customer picks an item to purchase, from a small selection of items available. Your program will display a menu which allows the customer to select various options to simulate the various kiosk operations. For this kiosk, only **5 items** (with fixed prices) are available for the customer to pick. The kiosk also offers an option for the customer to pick a random item (see Item Table in the ***Program Logic*** section below).

To purchase any item, the customer must first create an "***order***", by providing his/her** name and entering some "***credits***". Your program will keep track of the customer's credit balance and should stop him from purchasing any items which are too expensive. The program will also keep a record of all the items he has purchased, and the total cost of the items. At any point, the program can print out these statistics on request.

For this assignment, the program will only handle *ONE* customer at any one time. However, the customer can be changed while the program is executing. Every time a customer is changed (via **Option (1)**), the current order is finalised and a new order is created.

** *for the rest of the document, "he/his" will be taken to mean "he/she/his/her", etc.*

Program Logic

The ***Super Fantastic Kiosk*** program begins with a welcome message followed by a menu with the following options:

```
Welcome to Super Fantastic Kiosk
==========================
(1) Create A New Order
(2) Buy More Credit
(3) Purchase An Item
(4) What Have I Ordered So Far?
(5) Collect My Order
(6) Display Help
(7) Leave Kiosk
Choose an option:
```

Option (1) prompts the customer to create a ***new order***, by entering a **name** (for the customer) and a starting **credit**. The customer's name must not be blank, and the credit must be more than 0. If this option is chosen again after an order has already been set up or the order is in progress, a "new" order is created (i.e. with a new customer, and new credit). Note that the "new" order replaces the "old" order – i.e. there is only ever one order (and hence one customer) at any one time.

Option (2) allows the customer to "buy" more credit to add to his existing credit.

Option (3) simulates a ***purchase*** operation. The customer should be shown a list of available items to pick from (see Item Table below). If the customer picks Item #6, the computer generates a random number between 1 and 5 and then use that number to pick one of the 5 available items for him.

Item Table

Item Number	Item Name	Item Cost
1	Pen	$10
2	Book	$20
3	DVD	$30
4	Mouse	$40
5	Keyboard	$50
6	---	(computer picks a random item from above)

All the items purchased are recorded with the current Order, until the customer selects **Option (5)** to collect the items he has purchased.

For the purpose of this assignment, the "items" are simply implemented as a single `String`.

If the user enters a number which is less than 1, or more than 6, it should be rejected, and his credit balance remains unchanged.

If the customer tries to purchase an item which costs more than his current credit balance, the operation should be rejected, and an appropriate error message printed.

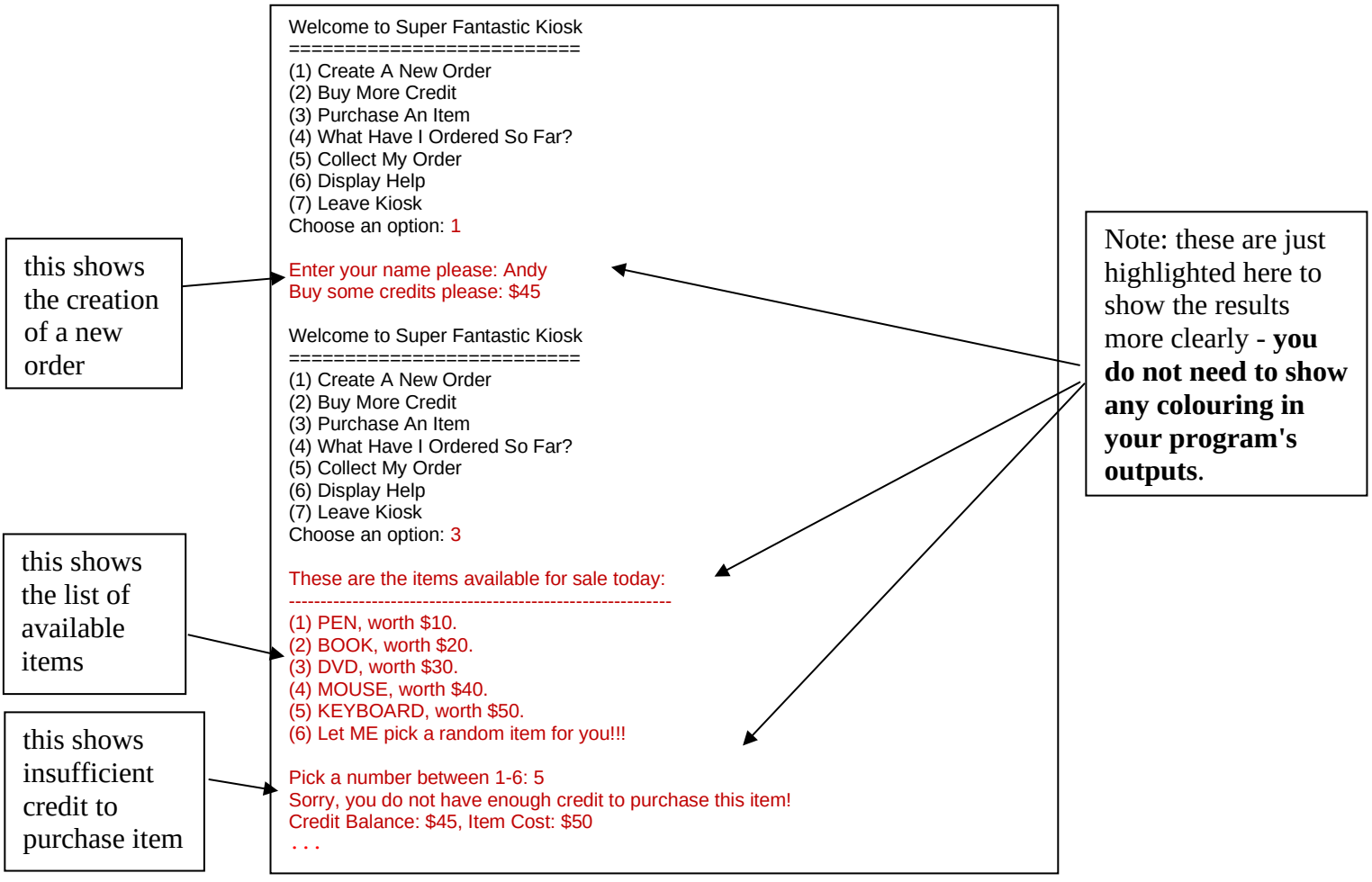
Option (4) displays some statistics about the customer's current order

Option (5) indicates that the customer has finished with his order, and wishes to collect the items purchased. This should also clear the current order, in preparation for the next order.

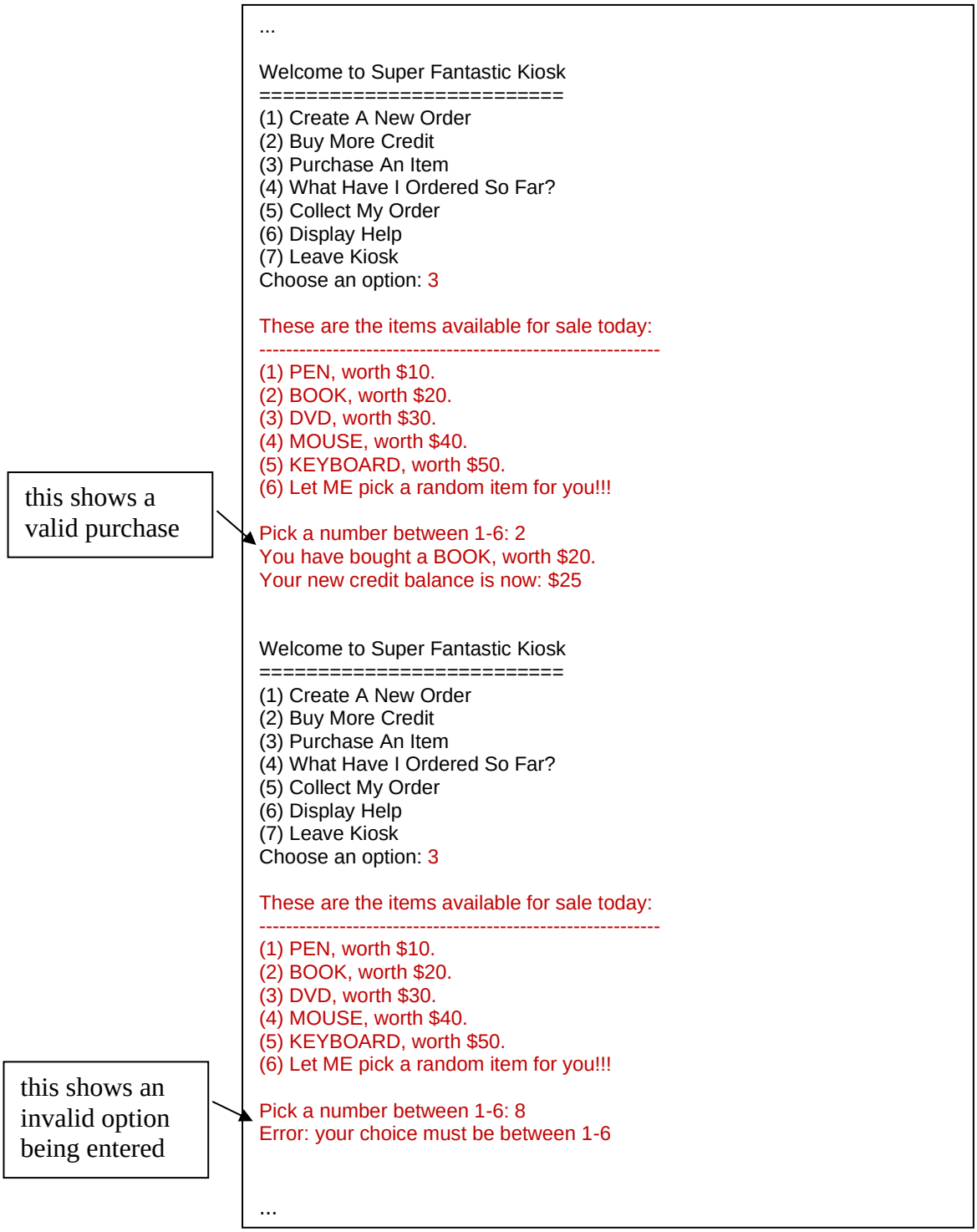
Option (6) displays some brief instructions regarding how to use the program. It must at least inform the customer of the rules of item selection (see Item Table under **Option (3)** above).

Option (7) exits the program. All customer/order statistics should be cleared.

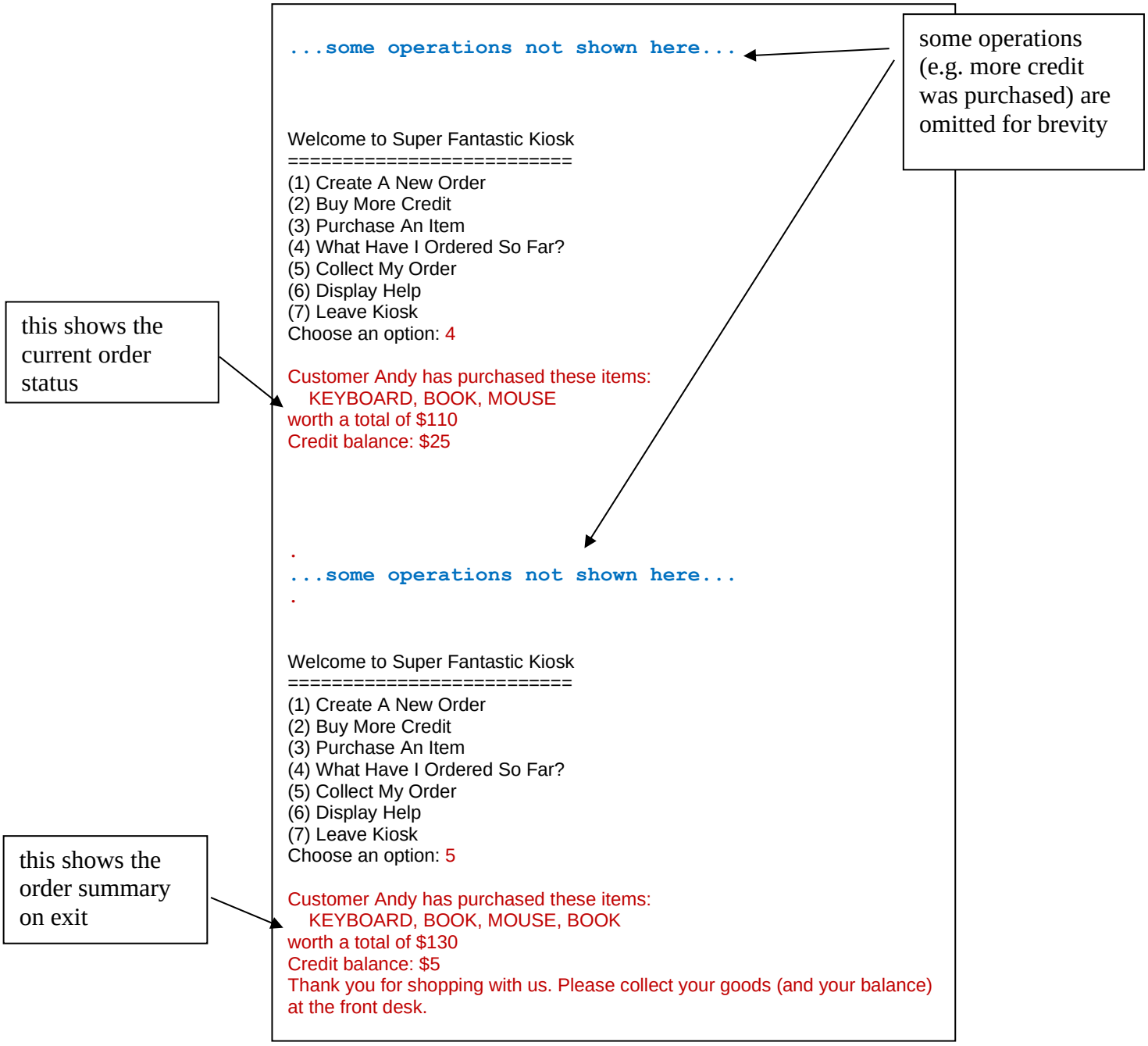
Sample screenshots of typical inputs/outputs (including invalid inputs) for **Options (1), (3), (4) & (5)**:



Screenshots continued….



Sample screenshots continued…



NB : all the above screenshots are related (ie. should be read from top to bottom), with some moves omitted for brevity.

Additional Notes:

The menu must be displayed repeatedly after each operation (as shown in the screenshots), until the user chooses **Option (7)**. Inputs other than 1-7 (including non-numeric characters) should be rejected, and an error message printed.

If the user chooses **Option (2), (3), (4) or (5)** before a new order has been created (**Option (1)**), an appropriate error message should be printed, with the operation aborted and menu re-displayed.

If the user chooses **Option (1)** when an order is in progress (i.e. the current customer has purchased some items), the existing order is finalised before the new order is created.

Your program must deal with invalid values entered by the user in a sensible manner.

For all the options, the inputs/outputs can be formatted in many different ways. Discuss with your tutor regarding how you should implement these in your program.

Assume that the items' costs, and the customers' credits, are all integers. Also assume that the user inputs are always of the correct data types (ie. when an integer is required, only an integer is entered, etc), except for the main menu options.

The sample screenshots above are merely meant to be examples. Your user interface need not be exactly as shown in those examples. However, you should discuss with your tutor about what to include in your user interface. Keep it simple and intuitive.

Program Design

Your program must demonstrate your understanding of the object-oriented concepts and general programming constructs presented in this unit.

The data type of each field in the classes must be chosen carefully in accordance with the requirements of the program described in the preceding sections, and you must be able to justify the choice of the data type of each field. You may want to include comments in the class to explain any assumptions made.

Your code should conform to the ***FIT9131 Java Coding Standards***; any violations will lead to marks being deducted.

Basic validation of values for fields and local variables should also be implemented. You should not allow an object of a class to be set to an invalid state. You should include appropriate constructor(s) for each class, and accessor/mutator methods for all the attributes.

You class design should include at least these classes:

`Kiosk`

`Customer`

`LuckyDipGenerator`

Discuss with your tutor what attributes/behaviour are suitable for these classes, and how they interact with each other.

You should make the following assumptions:

- a **`Kiosk`** object will be responsible for displaying the menus, accepting user responses, and performing all the requested operations. It will make use of one (1) **`Customer`** object and one (1) **`LuckyDipGenerator`** object.

- a **`Customer`** object will remember his own name, what items he has purchased (& their total costs). For this assignment, you can just store all the items purchased as a single **`String`** containing all their names, separated by some sensible delimiters,

  e.g. "**`Book, DVD, Mouse, Book, DVD, Book`**" - this means the customer has purchased 6 items so far. You can decide how to format the actual string.

- a **`LuckyDipGenerator`** object will be used to generate a number between 1-5. This is used by the program to choose the random item for the customer.

You can add any other appropriate classes in your design. However, if you do, you must discuss these with your tutor first. You should always start with a simple design, and then improve it if needed.

Assessment

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

- Object-oriented ***design*** quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of an object's state **30%**
- Adherence to ***FIT9131 Java Coding Standards*** **10%**
- Program ***functionality*** in accordance the requirements **60%**

You must submit your work by the submission deadline on the due date (a late penalty of 10% 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*.

Interview (Important: this is where most of your marks will come from)

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, your design, discuss design decisions and alternatives, and modify your code as required. ***Marks will not be awarded for any section of code or functionality that you cannot explain satisfactorily*** (your tutor 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 solely on the actual code itself.

Interview times will be arranged in the lab sessions in *Week 7*. It is your responsibility to attend the lab and arrange an interview time with your tutor.

The actual interview will take place during the week **after** week 7, via Zoom or other video facilities. ***You must have audio and video capabilities on your computer that can be used for the interview.***

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*** on or before the due date (as listed at the start of this document). The link to upload the assignment will be made available, along with a check-list of things to submit, in the ***Assessments*** section of the unit's Moodle site before the submission deadline. The submission requirements are as follows:

Your submission must be in the form of a **ZIP** file, uploaded to Moodle, and contains:

- the **BlueJ** project you created to implement your assignment. 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 your 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 failure to comply with any of these requirements. If you are unsure about the submission requirements (e.g. how to create a Zip file, or where does BlueJ store your Assignment project files), consult your tutor ***BEFORE*** the submission date.

During submission, you will be asked to accept an online submission statement. No physical **Assignment Cover Sheet** is needed.

Warning: **there will be no extensions to the due date**. Any late submission will incur the ***10%*** per day penalty. It is strongly suggested that you submit the assignment well before the deadline, in case there are some unexpected complications on the day (e.g. interruptions to your internet connection).

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.

Your program will be checked with a code similarity detector.

Academic integrity, plagiarism and collusion :
(https://www.monash.edu/students/academic/policies/academic-integrity)