COMP1202 - Programming 1
Coursework 2

Module Code:	COMP1202		
Module Title:	Programming 1		
Module Leader:	Abobakr Khalil Al-Shamiri		
Assessment Type:	Individual Coursework	**Weighting:**	20%
Submission Due Date:	03/01/2024 04.00 PM (Malaysia time)		
Method of Submission:	Blackboard submission link is named "Coursework 2 Submission".		

This assessment relates to the following Module Learning Outcomes:

A. Knowledge and Understanding	A1. Simple object-oriented terminology, including classes, objects, inheritance and methods. A2. Basic programming constructs including sequence, selection and iteration, the use of identifiers, variables and expressions, and a range of data types. A3. Good programming style.
B. Subject Specific Intellectual and Research Skills	B1. Analyse a problem in a systematic manner and model in an object-oriented approach.
D. Subject Specific Practical Skills	D1. Design a short program, compile the program, debug the program and test the program. D2. Use simple programming environments to aid the above process.

Coursework Brief:

This coursework mainly consists of **Polymorphism, Inheritance, Interface, Abstract Class**, and **Software Design concepts**.

Submission:

Put all your codes (only .java files) in a single zip folder and submit it to the blackboard submission link by the specified deadline.

Feedback: 2-3 weeks after deadline.

General notes:

- Keep your code clean, less redundant, complete, and easy to understand:
  - Clean: No extra unnecessary codes
  - Don't write the same code twice. Instead, put the useful chunk of codes into methods with proper parameters.
  - Complete: Achieve the basic requirements stated in the question file, but feel free to add more!
  - Easy to understand: Put the proper naming for variables and methods according to the Java convention.
- You can show your distinctive skills by experimenting with new and different things!
- Don't forget to test your codes in the main() method

Rules:

- No work can be accepted after feedback has been given.
- You should expect to spend up to **15 hours** on this assignment.
- Please note the University regulations regarding academic integrity.
- Copying works from/to other students will be penalized under University of Southampton's rule.

CHOICES AND PROGRAM FEATURES

This coursework, compared to the previous ones, is more open-ended. Imagine that you will develop a program with GUI (graphical user interface) next semester, treat this coursework as a preparation for it, where you will design the classes.

Pick ONE (1) of the following choices to begin with, or choose other similar systems as long as it meets the marking criteria:

A. Movie Ticket Booking System (imagine the one in TGV Cinemas)
Possible features:
- There is a catalogue of movies with showtimes and ticket prices.
- Customers can select and book movie tickets.
- The system automatically calculates the total ticket cost.
- Inventory management for concessions, including snacks and drinks, with the ability to mark items as out-of-stock.

B. Food Ordering System (like Foodpanda)
Possible features:
- Show restaurant menus with items and prices.
- Let users explore restaurants and add items to their cart.
- Allow users to review and confirm their orders before proceeding to checkout.
- Highlight special restaurant offers and discounts.

C. University Course Registration System (assume online course registration)
Possible features:
- Browse a diverse catalogue of courses with schedule details and seat availability.
- Students can register for preferred courses.
- Automatic calculation of fees and associated costs for the selected courses.
- Manage course availability and seat capacities for a streamlined registration experience.

Then, design the classes and implement various features in your classes. Refer to the marking criteria below.

MARKING CRITERIA

Part 1. Relations Tree (7% Mark) -> Put in a Word Document

Draw a simple tree that shows the relation between the classes. The diagram must show the class type (abstract or concrete), and the relationship type (implements, extends, or composed of).

Your overall implementation **should include the following features of OOP**:

1. Inheritance
2. Polymorphism
3. Interface
4. Abstract class

Part 2. Code/Software Design (40% Mark)

Criteria for a good software design which will be assessed in this coursework:

A. Flexibility/scalability (5% mark):
Highly readable code, such as appropriate variable/method naming, and put comments on the important parts of the code

1. Usage of variables is preferred over constant numbers, especially for common settings
2. Usage of arrays is preferred over multiple variables with similar purpose
3. Usage of loops is preferred over lengthy IF-else

B. Efficiency (15% mark):

1. Code with high cohesion: Each method/class should be responsible for only one logical task, as much as possible
2. Code with loose coupling: Different classes or methods should not be very dependent on each other, unless it is used to reduce code duplication
3. Responsibility-driven design: Each class should be responsible for manipulating its own data; the data should not be modified much by other classes
4. Think carefully about loops; minimize the number of loops as much as possible.

C. Error handling (5% mark):

1. For now, try-catch statements are not required, although it's good to include some when necessary, such as checking whether an input is a valid number (refer to lab 10)
2. You should prevent NullPointerException by checking if a value (such as array) is null
3. You should prevent other types of runtime error that could happen in your program (think about a wide variety of possibilities), such as division by zero, index out of bounds, etc.

D. Error prevention (5% mark):

1. You should implement appropriate encapsulation strategies, such as getters and setters, private or protected, and when to create classes.
2. You should think carefully about making loops and conditions, whether they are robust enough for different kinds of inputs

E. Testing strategies/automation (10% mark):

1. Some important methods must be tested using automated test; for example, if you are classifying marks (0-100) to grades A, B, C, you must create an automated testing to test whether the method produces the correct output for each input.
2. You need to add some print statements in the methods that you test; you may disable (comment) the line after you have finished the testing

Part 3. The Features and Creativity (43%)

Refer back to the **CHOICES AND PROGRAM FEATURES** section, which describes what your program should have. The marking criteria for the features are as follows:

A. Base Classes Structure (10% mark):

You have to define your classes using appropriate constructors, and getters and setters when necessary. A class can communicate with other classes when necessary.

B. Input and Data Storage (5% mark):

You may choose to create a program with choices (the typical 1. Input Data, 2. View) or just demonstrate the operations in main(). Here, you need to demonstrate the input process (such as inputting order for restaurant) and how do you keep the data in your program (using array or ArrayList or HashMap).

C. View, Delete, Modify Data (13% mark):

These 3 operations are important for any applications. Demonstrate these operations in your program, such as modifying food or order data for the restaurant program. Remember to consider the Code Design aspects.

D. Creativity - Additional Operations (15% mark):

Here, you need to implement additional aspects for your program. Some ideas for you (but not limited to) are listed below. Implement **ONE (1)** additional operations.

- **Save and load** the data to/from files. You may learn about it here https://www.w3schools.com/java/java_files.asp.

- **Print reports**, such as an academic transcript (for academic information system). For example, an academic transcript will include all the courses (and the marks) that he/she enrolled in the past semesters. At the bottom of the report, there is some information that indicates the performance of the student, such as "First class honours".

- **Do statistical operations**. Example for academic information system:
  - Get the student with the highest/lowest average mark
  - Calculate the average of average marks of all students
  - Draw a histogram of average marks. Histogram is a frequency plot. You can draw it using asterisks.

Part 4. Storytelling (10%) -> Put in a Word Document

Every programmer learns from mistakes. Write a story about:

- What problems did you encounter during the development of this coursework; what makes you stuck for many hours trying to figure out what's wrong
- How did you manage to solve the problems
- What are the things that you are really proud of about your program; imagine this is a business pitching session

------ ***End of file*** ------

Any work submitted after the deadline's time will be subject to the standard University late penalties unless an extension has been granted, in writing by the Senior Tutor, in advance of the deadline. Details on the University's late penalties can be found here:
• https://www.southampton.ac.uk/~assets/doc/quality-handbook/Late%20Submission.pdf