School of Computing and Information Technology
University of Wollongong

Session: **Autumn 2025**
Lecturer: **Dr Thanh Le**

CSIT213 Autumn 2025

Assignment 1

Due: Monday 28 April 2025, 11:30 pm
Total marks: 20 marks

Scopes

This assignment is related to Java classes' definitions and implementations, arrays, polymorphism, UML classes diagrams, and data input/output.

Please read the assignment specifications below carefully.

1. General Java coding requirements

- Create your programs with good programming style and form using proper blank spaces, indentation, and braces to make your code easy to read and understand.
- Create identifiers with sensible names.
- Add proper comments to describe your code segments where they are necessary for readers to understand what your code intends to achieve.
- Logical structures and statements are properly used for specific purposes.
- In **every source file** you submit for this assignment, you must include the following header:

```
/*-----------------------
Student name:
Student number:
Subject code: CSIT213
-------------------------*/
```

2. Submission and marking procedures

- A submission procedure is explained at the end of this document.
- A submission marked by Moodle as "Late" is treated as a late submission no matter how many seconds it is late. The policy regarding late submissions is detailed in the Subject Outline.
- An implementation that **does NOT compile** due to one or more syntactical or processing errors scores **NO marks**.
- All tasks must be solved individually without any cooperation from the other students. If you have any concerns, please consult your lecturer or tutor during lab classes or consultation hours. Plagiarism will result in a **failure** grade being recorded for the assessment task.
- The use of **GenAI** (e.g., ChatGPT or Microsoft Co-pilot) is **NOT permitted** for the assessment tasks.
- It is recommended to solve the problems before attending the laboratory classes to efficiently use supervised lab time.

Task 1 (5 marks)

This task requires developing a Java program that calculates the perimeter of a polygon using its vertex coordinates stored in a text file. This will help students practice control statements, data structures, file handling, and mathematical computations in Java.

1. Specifications

- The vertex coordinates of a polygon are stored in the text file *polygon.txt*. The first line of the file specifies the number of vertices (*N*) in the polygon. Each of the following *N* lines contains a vertex, represented by two integer values *(x, y)*, separated by a comma. A sample text file is provided below.

```
3          } 3 vertices
10, 10     }
200, 100   } Coordinates of the 3 vertices
10, 100    }
```

- Your program must calculate and display the perimeter by summing the Euclidean distances between consecutive vertices, ensuring that the last vertex connects back to the first.

  Hint: You may use the class `Point2D.Double` from `java.awt.geom` to define a vertex.

2. Deliverables

- **A single Java file** named *PerimeterCalculator.java*. **(4 marks)**

  Your program must:

  - follow the conventions for naming all classes, variables, and methods;
  - provide sufficient comments; and
  - use proper blank spaces, indentation, and braces to make your code easy to read and understand.

- **A PDF file** named *task_1.pdf* containing the compilation and testing results. **(1 mark)**

- Create **a ZIP file** named *task_1.zip* containing the Java file and the PDF file for submission.

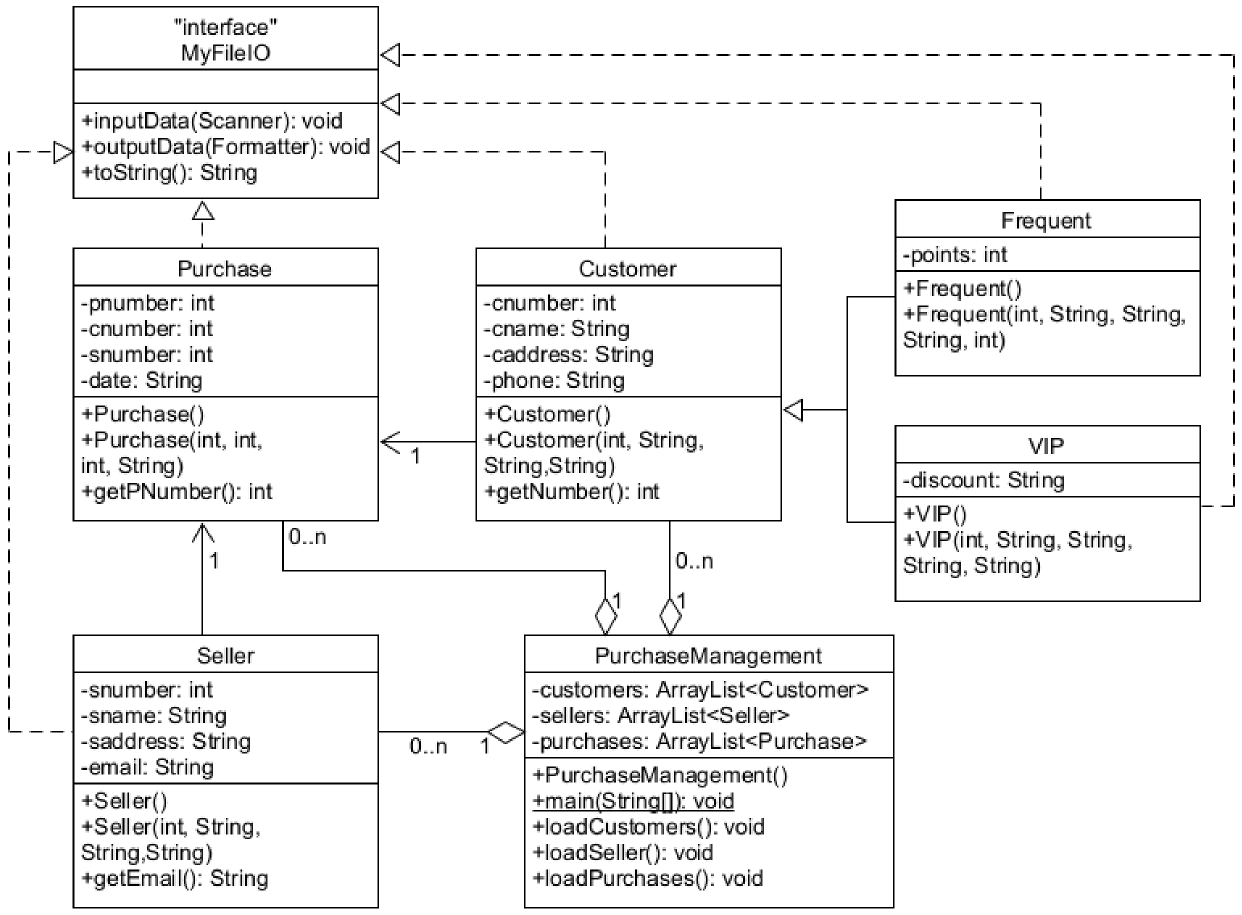
Task 2 (15 marks)

This task is based on a real-world application, where you are required to design and implement a Purchase Management System (PMS) in Java. This system helps a company manage its customers, sellers, and purchases.

1. Specifications

- When the program starts, the PMS initially loads data related to customers, sellers, and purchases from text files. It then displays a menu allowing the user to select an action: (1) display all information about the customers, sellers, or purchases; (2) search for a specific customer, seller, or purchase; (3) add new records; and (4) save all the data to text files.

- The UML class diagram for the PMS is shown below. You may add new classes, methods and attributes to the UML class diagram; however, you must **NOT** modify or remove any existing classes, attributes, or methods. Your Java implementation must be consistent with the UML class diagram.



- The program initially loads the data for customers, sellers, and purchases from **3 text files**: *customers.txt*, *sellers.txt*, and *purchase.txt*, respectively. The data is then stored in the containers by calling the methods *loadCustomers()*, *loadSellers()*, and *loadPurchases()*.

- Each field in the text file is separated by a comma "," and a space " ". The format of the file *customers.txt* is as follows:

```
F, 1, Willy, 41 Station Street Wollongong NSW 2500, 0462153975, 63
```

Name Address Phone

- Points for Frequent
- Discount (YES/NO) for VIP

Number

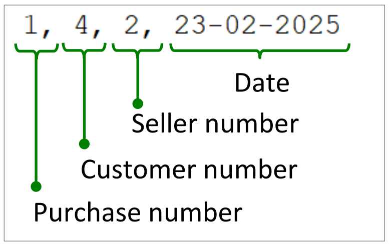
- Frequent customer (F)
- VIP customer (V)

- The format of the file *sellers.txt* is as follows:

```
1, Eadger, 187 Princes Highway Wollongong NSW 2500, eager@gmail.com
```

Number Name Address Email

- The format of the file *purchases.txt* is as follows:

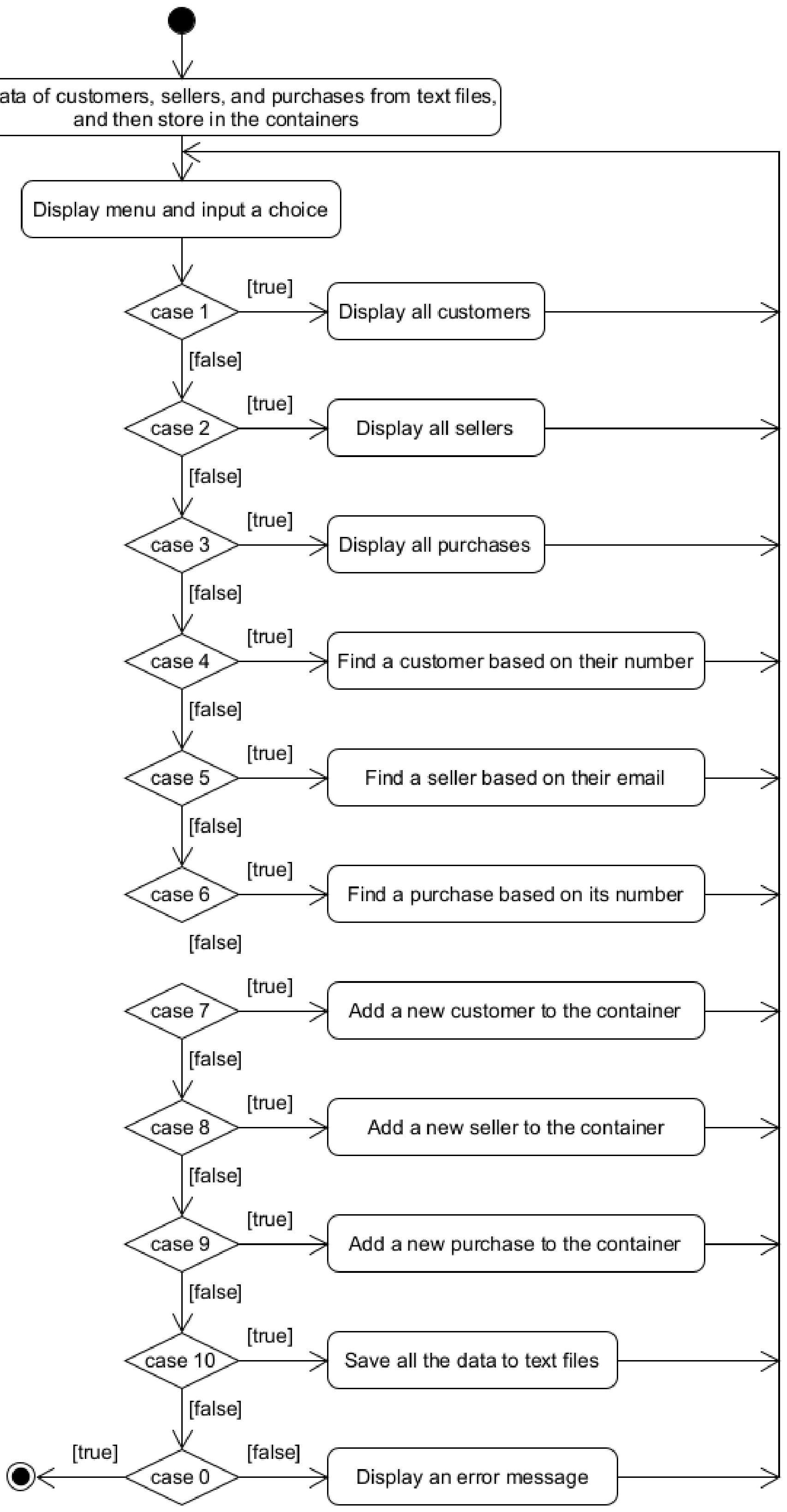


<u>Hint</u>: You may open a text file, and use the method `useDelimiter(", |\r\n|\n")` of a `Scanner` object before reading input data from the text file.

- After loading the data into the containers, the program displays a menu and allows users to choose an item until the input value is 0. A sample menu is provided below.

```
1. Display all customers.
2. Display all sellers.
3. Display all purchases.
4. Find a customer based on their number.
5. Find a seller based on their email.
6. Find a purchase based on its number.
7. Add a new customer.
8. Add a new seller.
9. Add a new purchase.
10. Save all data into files.
0. Exit.
Input a choice (0-10):
```

- An UML activity diagram for the PMS is provided below.


Load data of customers, sellers, and purchases from text files, and then store in the containers
Display menu and input a choice
case 1
[true]
Display all customers
[false]
case 2
[true]
Display all sellers
[false]
case 3
[true]
Display all purchases
[false]
case 4
[true]
Find a customer based on their number
[false]
case 5
[true]
Find a seller based on their email
[false]
case 6
[true]
Find a purchase based on its number
[false]
case 7
[true]
Add a new customer to the container
[false]
case 8
[true]
Add a new seller to the container
[false]
case 9
[true]
Add a new purchase to the container
[false]
case 10
[true]
Save all the data to text files
[false]
[true]
case 0
[false]
Display an error message

2. Deliverables

- **Five Java files** (or more if needed): *PurchaseManagement.java, Customer.java, Seller.java, Purchase.java,* and *MyFileIO.java*. **(13 marks)**

  Your program must:

  - be consistent with the UML class diagram;
  - follow the conventions for naming all classes, variables, and methods;
  - provide sufficient comments;
  - use proper blank spaces, indentation, and braces to make your code easy to read and understand;
  - follow the specified implementation steps; and
  - be able to repeat the main menu until users exit the system.

- **A PDF file** named *task_2.pdf* containing: (1) the UML class diagram, and (2) the compilation and testing results. **(2 marks)**

  - Remember to use the CSIT213 palette in the UMLet tool.
  - Use the option *File->Export* as to export the UML class diagram into a file in BMP format.
  - Insert the BMP file into the document.

- Create **a ZIP file** named *task_2.zip* containing all the Java files and the PDF file for submission.

3. Submission

Submit your deliverables through Moodle:

- Access the Moodle at *http://moodle.uowplatform.edu.au*.
- Log in and select the site *"CSIT213 (S125) Java Programming"*.
- Scroll down to the section "*Assignments and Submissions*".
- Click on the link "*Assignment 1 - Submission*".
- Click on the button "*Add Submission*".
- Upload the two ZIP files, *task_1.zip* and *task_2.zip*, into the box.
- Click on the button *"Save changes"*.

**** END****