OOP Programming Assignment
JUNE 2025

1. General Instructions

The general guideline for this assignment is as follows:

1. Any evidence of **plagiarism or collusion** will be taken seriously. University regulations will be fully enforced in such cases, and **ZERO marks** will be awarded to all parties involved.
2. The assignment carries a total of **100 marks** and contributes **20%** to the overall course grade.
3. This is a **group assignment** with a maximum of **TWO (2)** students per group. Students are free to choose their group members, and group members can be from different practical groups.
4. The deadline for assignment submission is **12 September 2025, Friday, before 11:55 pm** via **WBLE**.
5. **Submission Requirements**
   Each group must submit:
   - ONE (1) copy of the complete **Java source code files** (.java)
   - **UML class diagrams** in .docx or .pdf format
   - ONE (1) **presentation video (maximum 10 minutes)** that briefly explains the functionality of each class and the output of the application.

   All files should be compressed into a **single ZIP file** (*.zip) with a maximum file size of **30MB**. The ZIP file should be named using the format:

   Group_YourGroupNumber.zip

   Example: If your group number (as registered via the Excel sheet on WBLE) is 1, your file should be named:

   Group1.zip
6. **Late Submission Penalty**
   - Submission one day after the deadline: 10 marks deduction
   - Submission more than one day after the deadline: WILL NOT BE ACCEPTED

2. Background

In this assignment, you are required to develop a Hospital Management System (HMS) in Java using Eclipse IDE and object-oriented programming concepts. The main objective of the HMS is to manage activities in a hospital, including patients, doctors, administrative staff, medicine, facilities, and laboratories. Therefore, this HMS should contain tables for all records of patients, doctors, staff (i.e., nurses, pharmacists, security, etc.), medicine, facilities, and laboratories.

3. Requirements

Your assignment is to develop a program to manage the HMS described below.

3.1 Program Functions (25 marks)

Your program must fulfill the following functional specifications:

- Display a Welcome to the HMS and the current date and time when the program starts.
- Display a main menu contains all the six selections (Doctors, Patients, Medicine, Laboratories, Facilities, Staff).
- Declare, create, and initialize arrays or ArrayLists for each of the following categories: `doctors`, `patients`, `medicine`, `laboratories`, `facilities`, and `staff`. You are free to give any name during array initialization, but you are required to **initialize at least three values** for each of the array. For the `doctors` array, the first two entries (ID and Name) must correspond to your **group members' names and IDs** (use only the **last three digits** of each student ID).
- Provide an intuitive input method that allows users to add new entry to the selected category or display existing list of entries from corresponding arrays of the classes.
- After each operation, prompt the user to choose whether to:
  - **Return to the previous selection** (i.e., to add another new entry)
  - **Return to the main menu** (i.e., containing all six selections)
  - **Exit the program**

3.2 Object-Oriented Design (40 marks)

You much use **Java and apply object-oriented programming techniques** in your implementation. Your design should adhere to good object-oriented design principles such as:

- Single responsibility – a class should have responsibility for only one specific functionality, and this responsibility should be clearly encapsulated within the class.
- Open/closed principle – classes or modules should be open for extension but closed for modification, allowing your program to be scalable and maintainable.
- Efficiency and simplicity – design your code to be efficient, concise, and free from redundancy. Keep your implementation simple and focused.

The following are the basic classes that **MUST BE INCLUDED** in your program. The **responsibilities** of each class, along with suggested **data fields** and **methods**, are provided. You are free to choose appropriate **data types** and **method signatures**, and may **add additional classes** if necessary.

Staff:

The **`Staff`** class is used to represent a staff member. It includes the following

- Four `String` data fields named `id`, `name`, `designation`, and `sex`. A field named salary that stores `salary` (`int`) of the staff.
- A method named `newStaff()` that prompts user to enter staff's id, name, designation, sex and salary.
- A method name `showStaffInfo()` to show staff information in the following format:
```
[id]     [name]     [designation]     [sex]     [salary]
```

Doctor:

The **`Doctor`** class is used to represent a doctor object. It includes the following:

- Five `String` data fields named `id`, `name`, `specialist`, `workTime`, and `qualification`. A field named `room` that stores room number (`int`) of the doctor.
- A method named `newDoctor()` that prompts user to enter doctor's id, name, specialization, work time, qualification and room number.
- A method name `showDoctorInfo()` to show the doctor's information in the following format:
```
[id] [name] [specialist] [work time] [qualification] [Room No.]
```

Patient:

The **`Patient`** class is used to represent a patient object. It includes the following:

- Five `String` data fields named `id`, `name`, `disease`, `sex`, and `admitStatus`. A field named `age` that stores age (`int`) of the patient.
- A method named `newPatient()` that prompts user to enter patient's id, name, disease, sex, admit status and age.
- A method name `showPatientInfo()` to show the patient's information in the following format:
```
[id]    [name]    [disease]    [sex]    [admitStatus]    [age]
```

Medicine:

The **`Medicine`** class is used to represent a medicine object. It includes the following:

- Three `String` data fields named `name`, `manufacturer`, and `expiryDate`. Two `int` data fields named `cost` and `count`.
- A method named `newMedicine()` that prompts user to enter name, manufacturer, expiry date, cost and number of unit.
- A method name `findMedicine()` to show the medicine's information in the following format:
```
[name]    [manufacturer]    [expiry date]    [cost]
```

Lab:

The **`Lab`** class is used to represent a laboratory object. It includes the following:

- A `String` data fields named `lab`. A data fields named `cost` that stores cost (`int`) of the facility.
- A method named `newLab()` that prompts user to enter facility and cost.
- A method name `labList()` to show the lab information in the following format:
  `[lab]    [cost]`

Facility:

The **`Facility`** class is used to represent a facility object. It includes the following:

- A `String` data fields named `facility`.
- A method named `newFacility()` that prompts user to enter facility.
- A method name show`Facility()` to show the facility information in the following format:
  `[facility]`

HospitalManagement:

The **HospitalManagement** class serves as the main application class that controls the flow of the Hospital Management System (HMS). It is responsible for displaying messages, handling user input, and coordinating interactions between objects. This class should contain the **main()** method and include the following::

- Declare and create an array of **Doctor** objects with a capacity of **25**.
- Declare and create an array of **`Patient`** objects with a capacity of **100**.
- Declare and create an array of **`Lab`** objects with a capacity of **20**.
- Declare and create an array of **`Facility`** objects with a capacity of **20**.
- Declare and create an array of **`Medicine`** objects with a capacity of **100**.
- Declare and create an array of **`Staff`** objects with a capacity of **100**.
- Initialize each array with **at least three (3) sample objects** to represent existing data for doctors, patients, labs, facilities, medicines, and staff.
- Implement **selection controls** (e.g., `switch-case` or `if-else`) and/or **loops** to support the functional requirements described in **Section 3.1**.

3.3 Interface (5 marks)

- Application extension – Extend the application by adding new type of staff classes. A good design should allow extending the application with minimum changes to the main components of the program.
- Add exception handling to handle runtime errors (for example, incorrect data type of the input value entered by user)
- Graphical user interface (GUI) – Enhance the application by providing easy to use and visually appealing interface using **JavaFX technology**

3.4 Recording Video (30 marks)

- **Introduction**: Introduce yourself by stating your name, student ID, and the title of the assignment. Make sure your face is visible in the video.
- **Program Description**: Provide an overview of the program, explaining what the program is about, how it works, and present **all possible test cases** by showing relevant program outputs.
- **UML Class Diagrams**: Include and explain the **UML class diagrams** to illustrate the overall **structure of the program.**