Informatics and Computer Science	**Coursework Brief Proofing & Printing Confirmation Sheet**
Module Title **Programming in Java**	Module Code **24CSCI04C**
Module Leader **Dr. Khaled Alsheshtawi**	Semester **Two**
Proofed by **Dr. Randa Alanwar**	

I hereby confirm that:

- This coursework brief has been proof-read (spelling and grammar) ☑
- This coursework brief assesses the ILOs for the module ☑
- This coursework brief follows the approved template ☑
- All questions (and sub questions) have their marks specified ☑

Signed (Proof Reader): _______ Dr. Dr. Randa Alanwar ___________

Signed (Module Leader): _____ Dr. Khaled Alsheshtawi ___________

Informatics and Computer Science	**24CSCI04C** **Assignment 1** **2024/2025**
Module Title **Programming in Java**	
Module Leader **Dr. Khaled Alsheshtawi**	Semester **Two**
Assessment Weight **25 % of the total course mark**	Due Date **Week 6**

Instructions to students:

1. This is a group assignment; each group consists of (4 to 6) students. Kindly follow the instructions of the TAs regarding registration of the group.

2. Submission: The submission is via the e-learning system only, by 12:00 pm on the deadline day.

3. Assessment: Assessment evaluates the class diagram, the submitted java program code, oral discussions, and teamwork performance.

4. Feedback: Will be given through generic feedback in labs and specific written feedback for each group two weeks after submission.

5. Along with the submitted project, you need to submit: a fully completed and signed *Coursework submission form and* a *Statement of Academic Honesty Form*. You can only submit your own work. Any student suspected of plagiarism will be subject to the procedures set out in the GAR.

6. AI tools and online resources are allowed; however, their use must not exceed 30%. Students must specify the tools and resources used, the reasons for their usage, and the specific benefits gained from them.

I. Assignment Objective:

The objective of the assignment is to implement an automated system that utilises the OOP concepts and the Java programming language. Choose **ONE** of the following systems:

1. School Management System
2. Pharmacy Management System

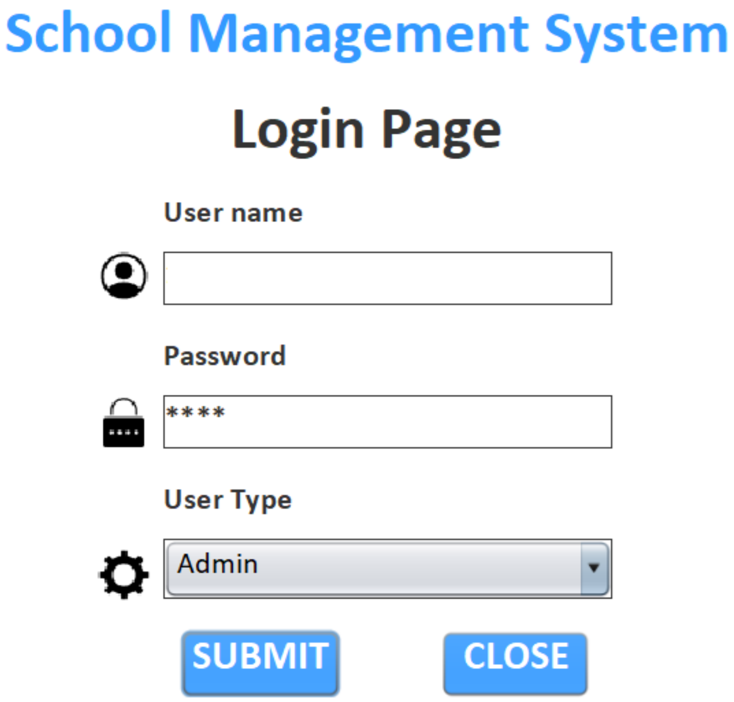
I.1 School Management System

The School Management System is a Java application that stores records of schools be it related to students, teachers, and staff.

Abstract: This application's objective is to help the school management system in managing the data easily. The manual system could be a complicated one when it comes to keeping the records so, there comes the role of this project. It holds personal records of students, teachers, and staff. This system contains modules for different roles be it admin, student, staff, and teacher. Here, the admin is responsible for maintaining the records in files.

Technologies Required: Swing, Java Graphical User Interface (GUI), JFrame, AWT, files.

Samples of the proposed system screenshots



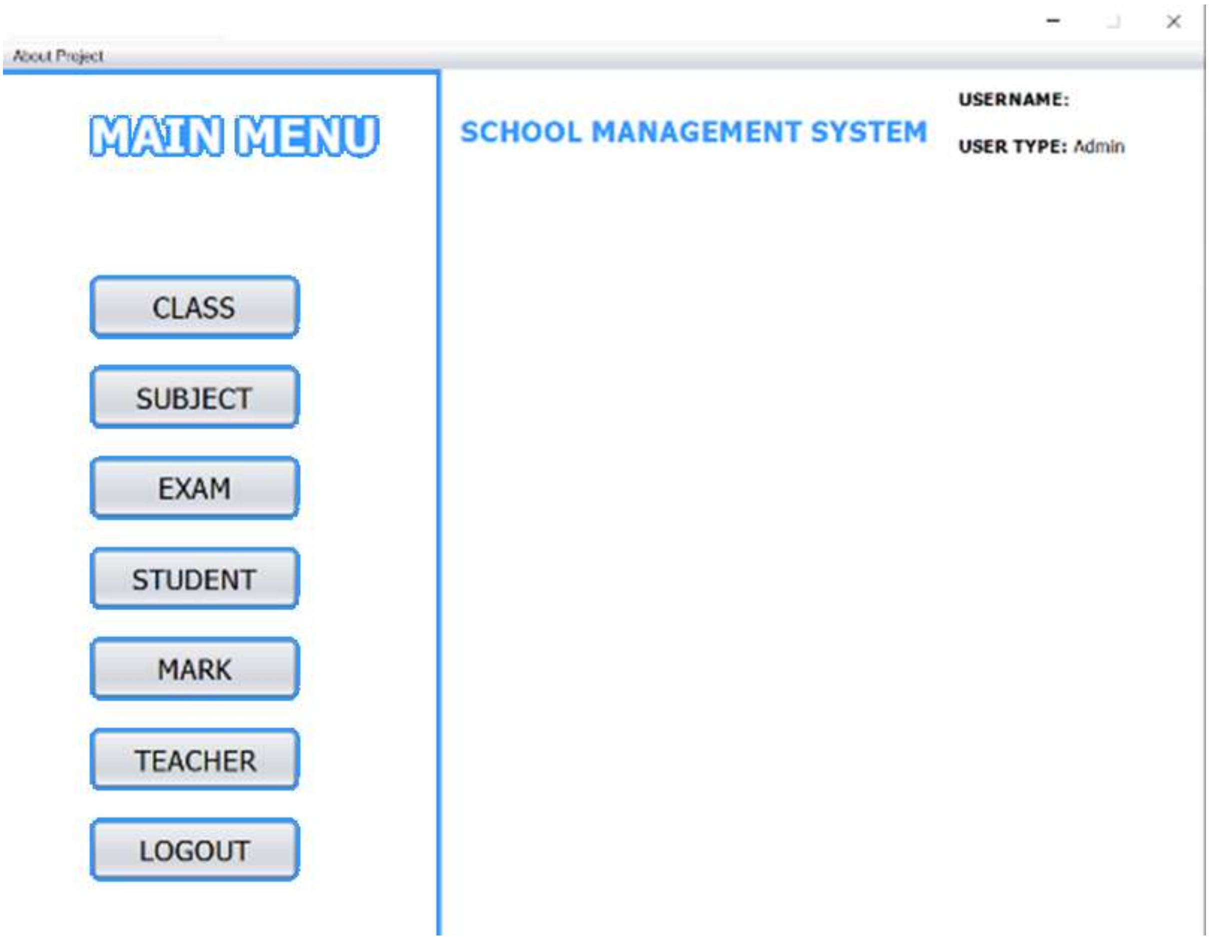
School Management System

Login Page

User name:

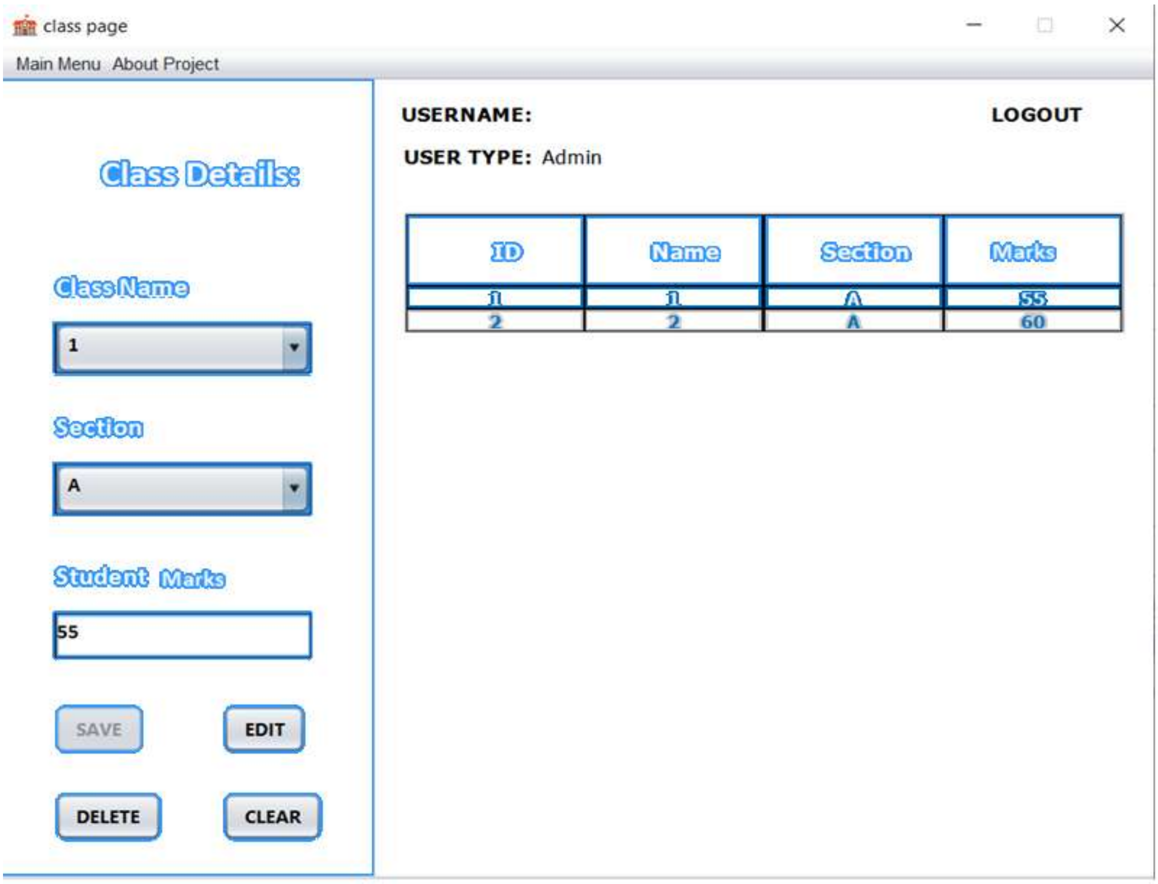
Password: ****

User Type: Admin



USERNAME:

USER TYPE: Admin



USERNAME:

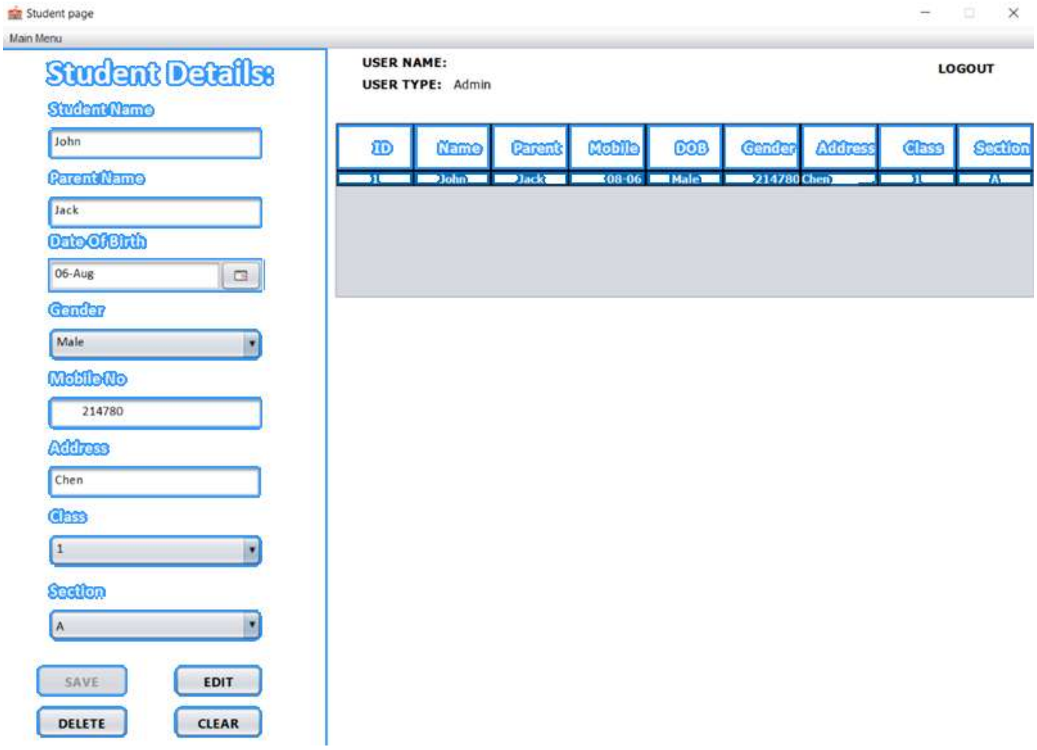
USER TYPE: Admin

Class Name: 1

Section: A

Student Marks: 55

ID	Name	Section	Marks
1	1	A	55
2	2	A	60



Student page

Main Menu

Student Details:

Student Name: John

Parent Name: Jack

Date Of Birth: 06-Aug

Gender: Male

Mobile No: 214780

Address: Chen

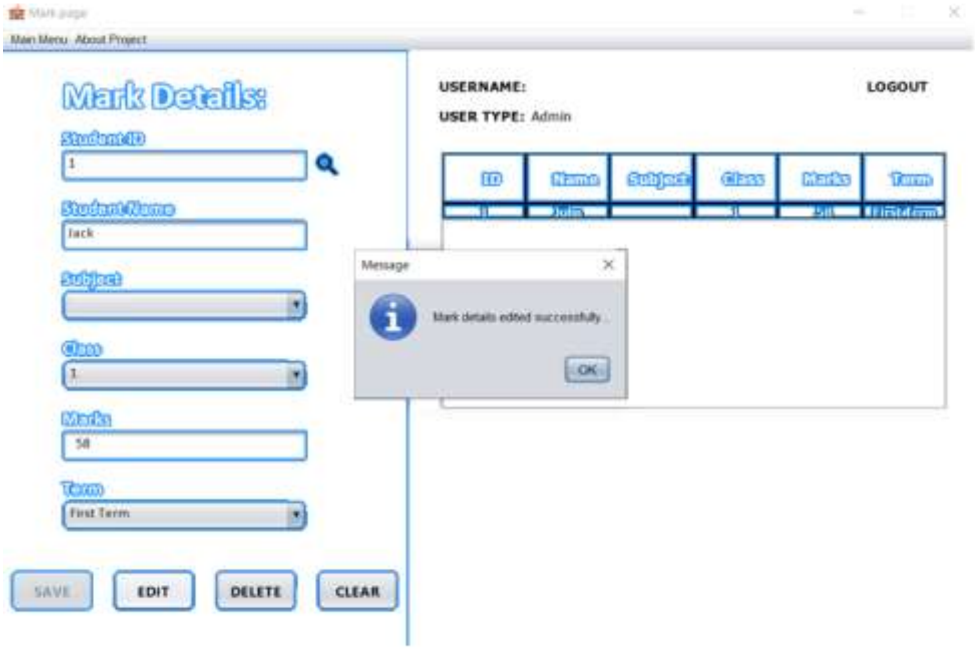
Class: 1

Section: A

USER NAME:

USER TYPE: Admin

ID	Name	Parent	Mobile	DOB	Gender	Address	Class	Section
1	John	Jack	08-06	Male	214780	Chen	1	A



Mark page

Main Menu About Project

Mark Details:

Student ID: 1

Student Name: Jack

Subject:

Class: 1

Marks: 58

Term: First Term

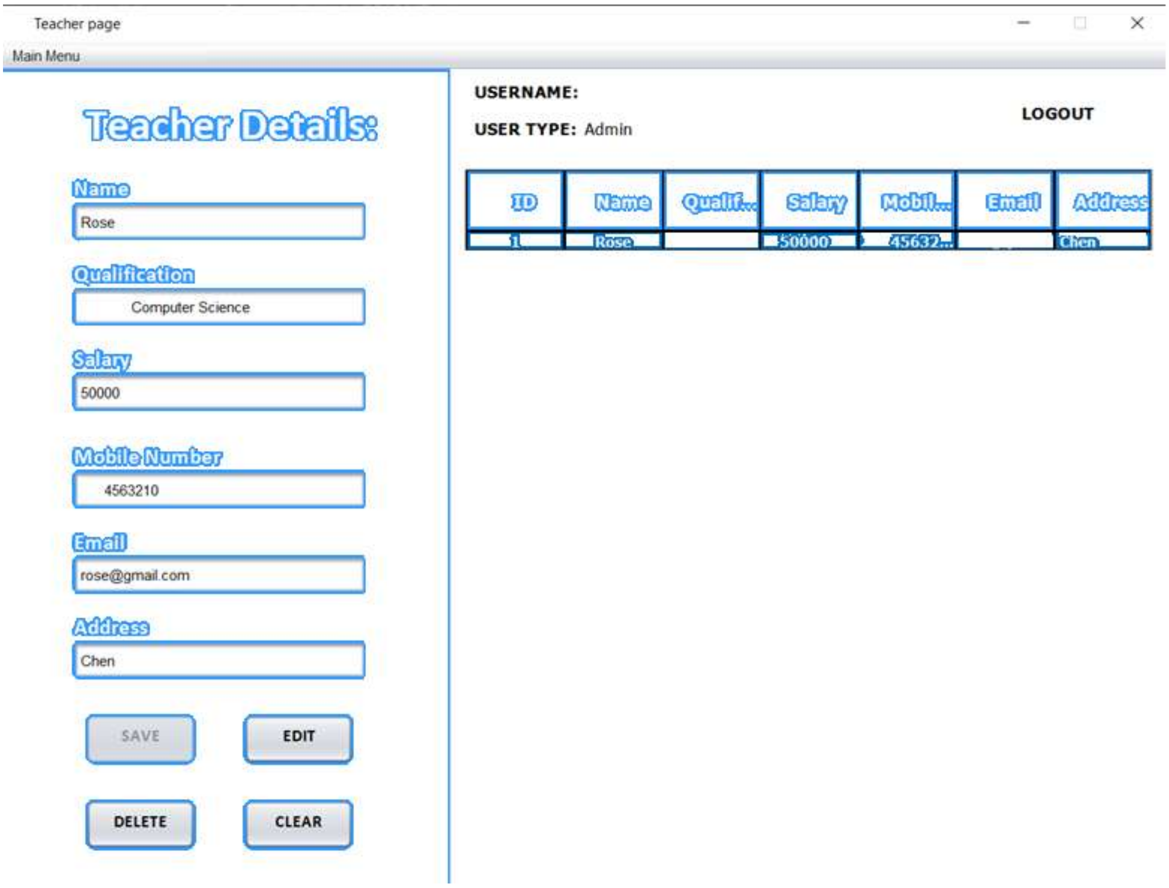
USERNAME:

USER TYPE: Admin

ID	Name	Subject	Class	Marks	Term
1	John		1	58	First term

Message

Mark details edited successfully...



Teacher Details:

Name: Rose

Qualification: Computer Science

Salary: 50000

Mobile Number: 4563210

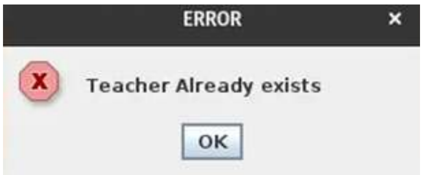
Email: rose@gmail.com

Address: Chen

USERNAME:

USER TYPE: Admin

ID	Name	Qualif...	Salary	Mobil...	Email	Address
1	Rose		50000	45632...		Chen



Teacher Already exists

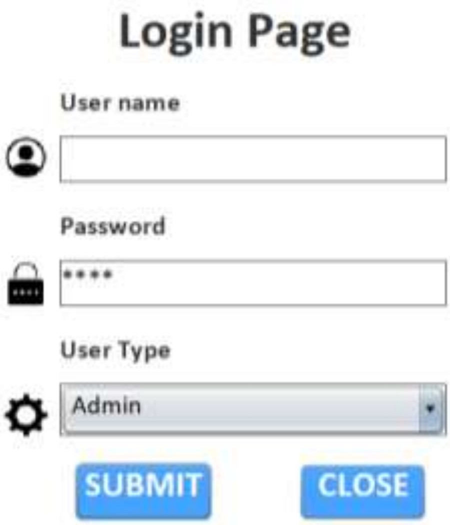
I.2 Pharmacy Management System

Pharmacy Management System is a web-based application built using Java that offers you the facility to order medicines, consult doctors and keep track of all your orders online by just signing up with a registered mail id.

Abstract: This application is of great help to the users who regularly goes for body check-up because this application gives you the comfort of consulting with a doctor at your comfort place. It comes with an excellent and friendly user interface comes with an automated billing system. It has an integrated chat feature where you can consult with a doctor regarding your health and it also tells you details of medicines and you can also track the status of ordered items.

Technologies Required: Swing, Java Graphical User Interface (GUI), and JFrame, AWT, files.

Samples of the proposed system screenshots

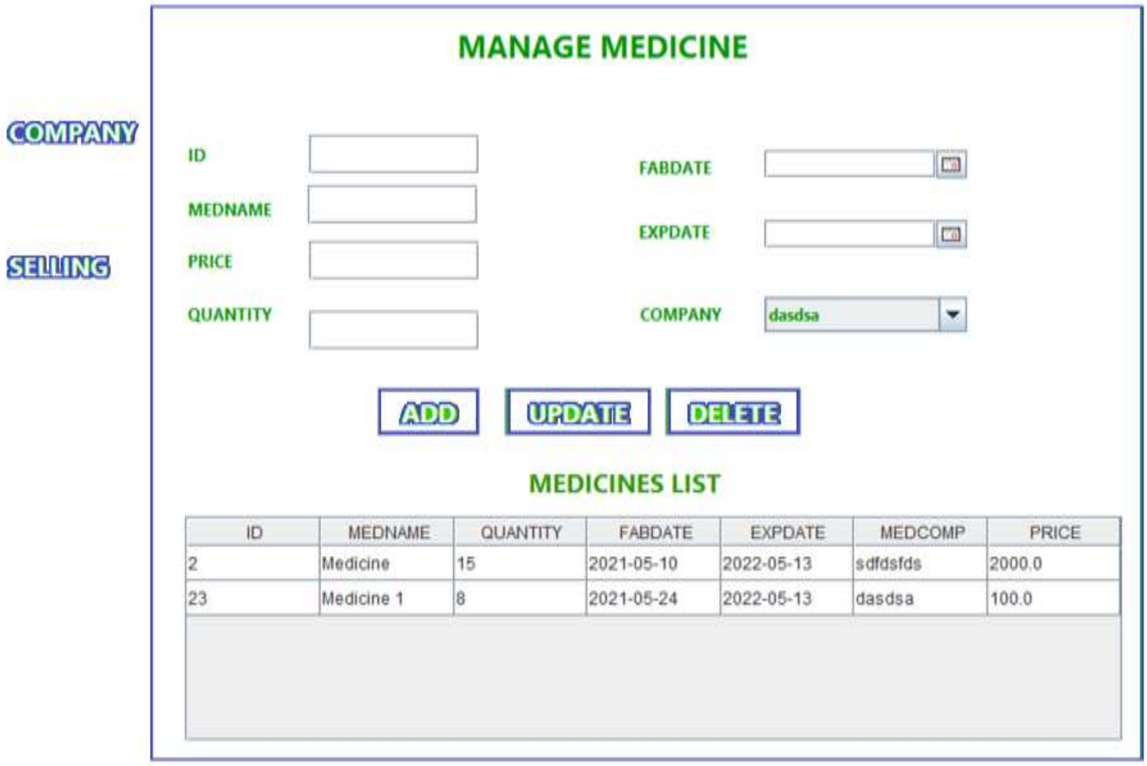


Login Page

User name:

Password: ****

User Type: Admin

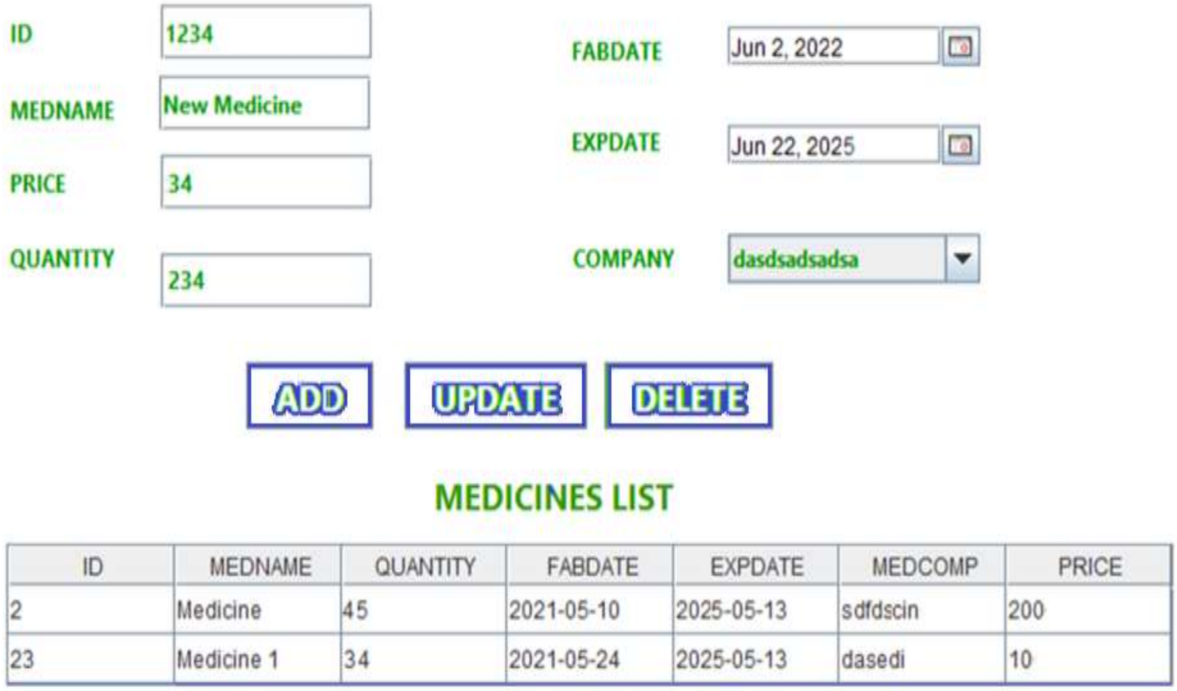


MANAGE MEDICINE

COMPANY: dasdsa

MEDICINES LIST

ID	MEDNAME	QUANTITY	FABDATE	EXPDATE	MEDCOMP	PRICE
2	Medicine	15	2021-05-10	2022-05-13	sdfdsfds	2000.0
23	Medicine 1	8	2021-05-24	2022-05-13	dasdsa	100.0



ID	1234	FABDATE	Jun 2, 2022
MEDNAME	New Medicine	EXPDATE	Jun 22, 2025
PRICE	34		
QUANTITY	234	COMPANY	dasdsadsadsa

MEDICINES LIST

ID	MEDNAME	QUANTITY	FABDATE	EXPDATE	MEDCOMP	PRICE
2	Medicine	45	2021-05-10	2025-05-13	sdfdscin	200
23	Medicine 1	34	2021-05-24	2025-05-13	dasedi	10

II. Deadlines and Deliverables:

You need to decide on the working order of your chosen system, and hence select the classes you will utilize. On Week 5, you are to submit the group members and your selected project. Starting week 6, you are to discuss in the lab a draft of your classes with your TA if you have questions.

Requirements: Analysis, Design and Classes Implementation

1. Implement the classes necessary to develop the system. Think carefully about the attributes, methods, modifiers, and the classes’ relations.
2. You have to make use the Inheritance, Interfaces and polymorphism in your design.
3. You are expected to make use of the Java Collections API by selecting from it the data structures suitable for your application.
4. Implement the main program suitable for testing the classes implemented (no graphical user interface is required in this assignment).

III. Submission: Each group needs to submit a pdf file and their source code via the e-learning system as follows:

1. Team members and a description of work division among them.
2. Detailed class diagram.
3. A zipped project file with the Java source code.

IV. Grading Standards:

Every criterion will make up an approximate percentage of the grade given to a single programming problem as indicated in the "Approx. % of Grade". Points will be assigned for a particular criterion roughly along the lines of the guidelines of the "Excellent," "Good," "Satisfactory," and "Not Met" evaluations.

For example, a team assignment that was marked as "Good" in the Requirements and Delivery criterion, "Satisfactory" for Coding Readability criterion, and "Excellent" in all other areas would receive: 0.8*0.2 + 0.6*0.2 + 1*0.2 + 1*0.2 + 1*0.1 + 1*0.1 = 88%.

****** As a special case, if a program does not meet the specifications at all / is entirely incorrect, no credit will be received for the other criteria either.***

1. <u>Requirements and Delivery</u>: (Approx. % of Grade: 20%)

1.1. (**Excellent**: **100**%):
- Ability to analyse problem and identify requirements. Completeness between 90% and 100% of the requirements.
- Delivered on time, and in correct format.

1.2. (**Good**: **80**%):
- Ability to analyse problem and identify requirements. Completeness between 80% and 90% of the requirements.
- Delivered on time, and in correct format.

1.3. (**Satisfactory**: **60**%):
- Ability to analyse problem and identify requirements. Completeness between 70% and 80% of the requirements.
- Delivered on time, and in correct format.

1.4. (**Not Met**: **<40**%):
- Ability to analyse problem and identify requirements. Completeness less than 70% of the requirements.
- Delivered on time, but it not in correct format.

2. <u>Coding Readability</u>: (Approx. % of Grade: 20%)

2.1. (**Excellent**: **100**%):
- Includes name, date, and assignment title.
- Excellent use of white space and (indentation, blank lines).
- Creatively organized work.
- Excellent use of variables (no global variables, unambiguous naming).

2.2. (**Good**: **80**%):
- Includes name, date, and assignment title.
- Good use of white space and (indentation, blank lines).
- Organized work.
- Good use of variables (no global variables, unambiguous naming).

2.3. (**Satisfactory**: **60**%):
- Includes name, date, and assignment title.
- White space makes program fairly easy to read.
- Organized work.
- Good use of variables (few global variables, unambiguous naming).

2.4. (**Not Met**: **<40**%):

- No name, date, or assignment title.
- Poor use of white space and (indentation, blank lines).
- Disorganised and messy.
- Poor use of variables (many global variables, ambiguous naming).
-

3. <u>Documentation</u>: (Approx. % of Grade: 20%)

3.1. (**Excellent**: **100**%):

- Comments are embedded in the program, they are comprehensive, descriptive, and give the reader full understanding of the program's operation
- Clearly and effectively documented including descriptions of all class variables.
- Specific purpose noted for each function, control structure, input requirements, and output results.

3.2. (**Good**: **80**%):

- Comments are embedded in the program, they cover important elements, describe those elements well, and give the reader a general understanding of the program's operation
- Clearly documented including descriptions of all class variables.
- Specific purpose is noted for each function and control structure.

3.3. (**Satisfactory**: **60**%):

- Very few comments are embedded in the program, they cover very few elements, descriptions are terse, and the reader cannot understand some portions of the program's operation
- Basic documentation has been completed including descriptions of all class variables
- Purpose is noted for each function

3.4. (**Not Met**: **<40**%):

- Very limited or no documentation included
- Documentation does not help the reader understand the code

4. <u>Code Design Efficiency</u>: (Approx. % of Grade: 20%)

4.1. (**Excellent**: **100**%):

- Solution is efficient, easy to understand, and maintain
- The program design uses appropriate structures. The overall program design is appropriate.
- The code could be reused as a whole or each routine could be reused
- Satisfies all of the following: Input/Output, Control Structures, Libraries, Classes Breakdown and their relationships, Methods, Attributes.

4.2. (**Good**: **80**%):

- The code is fairly efficient without sacrificing readability and understanding
- The program design generally uses appropriate structures. Program elements exhibit good design.
- Most of the code could be reused in other programs.
- Satisfies most of the following: Input/Output, Control Structures, Libraries, Classes Breakdown and their relationships, Methods, Attributes.

4.3. (**Satisfactory**: **60**%):

- A logical solution that is easy to follow but it is not the most efficient
- Not all of the selected structures are appropriate. Some of the program elements are appropriately designed.
- Some parts of the code could be reused in other programs.
- Suffers from Libraries.

4.4. (**Not Met**: **<40**%):

- A difficult to understand and inefficient solution. Code is huge and appears to be patched together.
- Few of the selected structures are appropriate. Program elements are not well designed.
- The code is not organized for reusability.
- Suffers from Classes Breakdown and their relationships and Libraries.

5. <u>Program Output / Runtime</u>: (Approx. % of Grade: 10%)

5.1. (**Excellent**: **100**%):

- Executes without errors excellent user prompts, good use of symbols, spacing in output.
- Thorough and organized testing or input validation has been completed

5.2. (**Good**: **80**%):

- Executes without errors
- User prompts are understandable, minimum use of symbols or spacing in output
- Most testing or input validation completed

5.3. (**Satisfactory**: **60**%):

- Executes without errors
- User prompts contain little information, poor design
- Some testing or input validation has been completed

5.4. (**Not Met**: **<40**%):

- Does not execute due to errors
- User prompts are misleading or non-existent.
- No testing has been completed, or no input validation

6. <u>Oral Presentation / Fielding Questions</u>: (Approx. % of Grade: 10%)

6.1. (**Excellent**: **100**%):

- The team present their program to the TAs clearly, confidently, and accurately on all portions of the code and its operation.
- The team answer all questions from the TAs about the program and its operation clearly, smoothly, confidently, and convincingly.
- The team provide alternative solutions.

6.2. (**Good**: **80**%):

- The team present their program to the TAs in an understandable way, with slight hesitations, and mostly accurate on all portions of the code and its operation.
- The team answer all questions from the TAs about the program and its operation clearly, smoothly, confidently, and convincingly.
- The team explain the entire program correctly as it is.

6.3. (**Satisfactory**: **60**%):

- The team present their program to the TAs in an slightly confused way, with noticeable hesitations, and occasionally accurate on portions of the code and its operation.
- The team answer all questions from the TAs about the program and its operation somewhat clearly, slight hesitations, confident but reserved, and most times convincingly.
- The team explain a little program design.

6.4. (**Not Met**: **<40**%):

- Student presentation of their program to the TA is not comprehensible, many hesitations, and not accurate with any portion of the code and its operation.
- The team answer only a few questions from the TA about the program and its operation and the answers are lacking in clarity, there are long hesitations, they are unconvincing.
- The team is unable to explain the program design.

Module Specification – (Programme Specs Ver. 6.0)

Module Code: CSCI04C	**Title:** Programming in Java	
Level: C	**Modular weight:** 10	**Faculty/Dept.:** ICS/CS
Pre-requisite modules: CSCI02P – Introduction to Programming and Problem Solving		
Reassessment: No restriction		
Module Leader: Dr. Khaled Alsheshtawi		
Semester taught: Two		
Date of latest revision: March 2022		

Aims

The aim of this module is to provide basic grounding in Java language programming and lay a firm foundation from which other modules, that require Java programming, may rely upon. Students will learn to write moderately complex Java code, understand the basics of the language API, and development practices (the object-oriented development process, test-driven development, and refactoring).

Intended Learning Outcomes

On completion of this module students should be able to:

Knowledge and understanding

1. Define concepts, principles and philosophy of Java language, and the Java technologies relating to web applications. [A2]

Intellectual Skills

2. Develop an analytical approach to interpreting intermediate problem specifications and a systematic approach to problem solving. [B2]

Practical and Professional Skills

3. Design and develop solutions to a number of intermediate programming problems using Java. [C2]
4. Use Java API (AWT, Swing, Collection…etc.) efficiently to solve intermediate programming problems. [C7]

General and Transferable Skills

5. Develop interpersonal skill of self-learning, planning, time management, and communication on individual and team programming projects. [D1, D2]

Employability

This module will provide opportunities for students to:

1. Understand the importance of being self-motivated in order to progress the area of work. [A5].
2. Carry out a range of complex ICT activities related to their work that involve application software. [B.3.1]
3. demonstrate effective planning, prioritisation and organisation to plan activities and carry them through effectively [C.2.2]

4. demonstrate effective team working by building and developing appropriate relationships with academic staff, peers, colleagues and people within the organisation. [C.3.2]

Indicative Content

- Java classes and methods.
- OOP concepts in Java.
- Graphical User Interface Interfaces. I/O in Java.
- Collection framework.
- Handling events and exceptions.
- Java applications & applets.
- Unit Testing

Methods of Learning, Teaching and Assessment

Total student effort for the module: 100 hours on average over one semester.

Type of session	ILOs Covered	Typical Student Effort		
		Typical number in the semester/s	Typical hours per week	Total hours
Lecture	1-4	12	2	24
Tutorial	-	-	-	-
Laboratory	2-5	12	2	24
Private study	1-4	-	-	52

Assessment

Assessment Type	Weight %	ILOs Assessed	Exam Semester	Exam/ Written Coursework Length
Two group programming assignments, (4 - 6 students).	50	3-5	2	-
One unseen written exam	50	1-3	2	2 hours

Methods of Feedback

In response to assessed work:

- Specific written feedback will be provided for assignment one in written form and returned to students along the marked coursework.
- Generic feedback in laboratory for assignment two and marked coursework with the marking schema published on E-Learning.
- Generic feedback for unseen exam on e-learning.

Developmental feedback generated through teaching activities:

- Dialogue between students and staff in labs and lectures

Indicative Reading List

- Paul Deitel and Harvey Deitel. Java How to Program, 11th edition. Prentice Hall, 2022.
- Liang, Y Daniel. Introduction to Java Programming-Comprehensive Version, 12th Edition. Pearson, 2019.