1st SIT Coursework - 01 Question paper **Year Long 2024 2025**

Module Code:	**CS4001NT**
Module Title:	**Programming**
Module Leader:	**Mr. Jeevan Poudel / Mr. Mohit Sharma (Islington College)**

Coursework Type:	Individual
Coursework Weight:	This coursework accounts for ***60%*** of the overall module grades.
Submission Date:	Friday, 16 May 2025
Coursework given out:	Week 12
Submission Instructions:	Submit the following to the Itahari International College's MST portal before 01:00 PM on the due date: ● A report (document) in pdf format in the MST portal or through any medium which the module leader specifies.
Warning:	London Metropolitan University and Itahari International College takes plagiarism very seriously. Offenders will be dealt with sternly.

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed overleaf. Please sign below to say that you have read and understand these extracts:

Extracts from University *Regulations on Cheating, Plagiarism and Collusion*

Section 2.3:*"The following broad types of offence can be identified and are provided as indicative examples ….*

(i) Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.

(ii) Falsifying data in experimental results.

(iii) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.

(iv) Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.

(v) Collusion to present joint work as the work solely of one individual.

(vi) Plagiarism, where the work or ideas of another are presented as the candidate's own.

(vii) Other conduct calculated to secure an advantage on assessment.

(viii) Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation, and computer programs.

2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. "$e = mc^2$ (Einstein 1905)". A *reference* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

School of Computing, FLSC

CONTRACT CHEATING

Contract cheating (also known as assessment outsourcing, commissioning or ghost writing) is when someone seeks out another party, or AI generator service, to produce work or buy an essay or assignment, either already written or specifically written for them or the assignment to submit as their own piece of work.

Contract cheating undermines the integrity of the academic process and devalues the qualifications awarded by the university. Students are reminded that academic integrity is a fundamental principle of our institution. Engaging in contract cheating not only impacts the individual's academic record but also the reputation of the university.

Students are encouraged to seek support if they are struggling with their coursework. The university offers a range of resources, including academic counseling, tutoring services, and workshops on study skills and time management. Utilizing these resources can help students achieve their academic goals without resorting to dishonest practices.

Penalty:

- Failure in the Module: The student must re-register for the same module, and the re-registered module will be capped at a bare pass.

- Ineligibility to Continue on the Course: Where re-registration of the same module, or a suitable alternative, is not permissible, the student will not be able to continue on the course. Additionally, the following penalty will be applied to the student's final award:

    - Undergraduate Honors: The student's final classification will be reduced by one level.
    - Unclassified Bachelors: Downgraded to Diploma in Higher Education.
    - Foundation Degree: Distinction downgraded to Merit; Merit downgraded to Pass; Pass downgraded to Certificate in Higher Education.
    - Masters: Distinction downgraded to Merit; Merit downgraded to Pass; Pass downgraded to Postgraduate Diploma.

Reporting and Consequences:

Instances of contract cheating will be thoroughly investigated, and students found guilty will face the penalties outlined above. It is the responsibility of every student to ensure that their work is their own and to avoid situations that could lead to accusations of academic misconduct.

By adhering to these standards, students contribute to a fair and equitable academic environment, ensuring the value and recognition of their qualifications are maintained.

Aim

The aim of this assignment is to implement a real-world problem scenario using the Object-oriented concept of Java that includes creating a class to represent a **GymMember**, together with its two subclasses to represent **RegularMember** and **PremiumMember** respectively.

GUI part of coursework focuses on expanding the project by integrating a new class to implement a graphical user interface (GUI). The goal is to create a system that stores Gym Member details within an ArrayList. This class will contain the main method and will be executed via the command prompt. Additionally, you are required to submit a report explaining the functionality and structure of your program.

Deliverables

Create a new project in **BlueJ** and create three new classes (**GymMember, RegularMember and PremiumMember**) within the project. **RegularMember** and **PremiumMember** are **subclasses** of the class **GymMember**.

For the GUI implementation part, add a new class to your project named **GymGUI**. Once your solution is complete, submit the **GymGUI.java** file along with the **GymMember.java, RegularMember.java,** and **PremiumMember.java** files. Ensure that no other files from the project are included. Additionally, submit your report in **PDF format.**

Scenario

You are developing a system to manage members of a gym that offers different membership plans and tracks attendance and loyalty points. Each member is identified by a unique ID and has personal details such as their name, location, phone number, email, membership start date, and membership plan. You have to keep track of how often members visit the gym (attendance) and their accumulated loyalty points.

NOTE: - The program should include following classes (with no additional attributes or methods).

Program (55 marks)

1. Parent Class: GymMember [5 marks]

The **GymMember** is a parent class as **abstract** class which has following attributes with their respective data type and with protected access modifier as mentioned below:

id - int	**DOB - String**
name - String	**membershipStartDate - String**
location: String	**attendance - int**
phone - String	**loyaltyPoints - double**
email – String	**activeStatus – boolean**
gender – String	

The constructor of this class accepts **id, name, location, phone, email**, **gender**, **DOB**, and **membershipStartDate** as parameters. The attributes **attendance** and **loyaltyPoints** are initialized to **zero** and a**ctiveStatus** is set to **false.** Additionally, assign **id, name, location, phone, email, membershipStartDate, plan** and **price** with the **parameter values.**

Each attribute has a **corresponding accessor method**.

An abstract method **markAttendance()** is created to track attendance of the member.

A method **activateMembership()** sets **activeStatus to true** when the membership needs to be activated or renewed and a method **deactivateMembership()** sets the **activeStatus to false** if the membership needs to be deactivated or paused.

[Note: The membership must be activated first in order to deactivate. Also, the membership is deactivated by default]

Also, method **resetMember()** is created to set **activeStatus** of the member to **false, attendance** is set to **zero** and **loyaltyPoints** to **zero.**

A **display** method should output (suitably annotated) the **id, name, location, phone, email, gender, DOB, membershipStartDate, attendance, loyaltyPoints** and **activeStatus.**

2. Subclass: RegularMember [10 marks]

The **RegularMember** class is a subclass of **GymMember** class, and it has **six private** attributes with following data types:

attendanceLimit - **int (final)**

isEligibleForUpgrade - **boolean**

removalReason - **String**

referralSource - **String**

plan - **String**

price - **double**

The constructor accepts nine parameters which are **id**, **name**, **location**, **phone**, **email**, **gender, DOB**, **membershipStartDate** and **referralSource**. A call is made to the superclass constructor with eight parameters. The attribute **isEligibleForUpgrade** is set to **false**, **attendanceLimit** is set to **30, plan** is set to **basic**, **price** is set to **6500** and removalReason is set to empty. Also, assign other attributes with corresponding parameter values.

Each attribute of RegularMember class has a corresponding **accessor** method.

Implement the abstract method **markAttendance()** to increment the **attendance** value by 1 each time this method is invoked. Also, the **loyaltyPoints** should be increased by **5** points.

There is a method named **getPlanPrice()** with return type **double** to retrieve the price of the provided plan. This method accepts **plan** as a parameter and returns its corresponding price. The following are the pricings of available gym plans:

basic 6500

standard 12500

deluxe 18500

The **switch** statement must be implemented for this specific scenario to return the corresponding plan's price.

[Note: if the invalid plan is passed as an argument then the getPlanPrice() method should return -1]

There is a method **upgradePlan**() with return type **String**. This method is used to upgrade the plan subscribed by the member. The method accepts **plan** as a parameter. If the member is eligible for upgrading the plan, then the plan and price should be updated accordingly by calling **getPlanPrice()** method.

Also, the system should validate and display appropriate message if same plan is chosen that the member is currently subscribed to.

Note: If getAttendance() >= attendanceLimit then set isEligibleForUpgrade to true

[Hint: if the method getPlanPrice() returns -1 then an appropriate message should be displayed. The upgradePlan() method should return the appropriate message as the return type of the method is String**]**

Now, create **revertRegularMember()** method which accepts **removalReason** as a parameter. A super class method: **resetMember()** is called here and **isEligibleForUpgrade** set to false, **plan** is set to basic and **price** is set to 6500. Also, assign other attributes with corresponding parameter values.

A method to **display** the details of the **RegularMember** is required. It must have the **same signature** as the display method in the parent class. It will call the method in the super class to display all the attributes of super class. Also, display the value of attributes: **plan** and **price.** Additionally, display **removalReason** if its value is not empty.

3. Subclass: PremiumMember [10 marks]

The **PremiumMember** class is a subclass of **GymMember** class, and it has three attributes:

premiumCharge	**- double(final)**
personalTrainer	**- String**
isFullPayment	**- boolean**
paidAmount	**- double**
discountAmount	**- double**

The **constructor** accepts **nine** parameters which are **id, name, location, phone, email, gender, DOB, membershipStartDate,** and **personalTrainer.** A call is made to the superclass constructor with eight parameters. The attribute **premiumCharge** is set to **50,000**, **paidAmount** is set to 0 and **isFullPayment** is set to **false**, and **discountAmount** is set to **zero.** Also, assign other attributes with corresponding parameter values.

Each attribute of **PremiumMember** class has a corresponding **accessor method.**
Here, a method **payDueAmount()** of return type **String** is created to pay the due amount. It accepts **paidAmount** as a parameter. Received paidAmount from the parameter should be added on **paidAmount** attribute.

Following validations should be made inside this method:-

- if payment is already full **(this.isFullPayment == true),**a suitable message should be displayed and returned.
- if the total paid amount is **more** than premiumCharge, a suitable message should be displayed and returned.

Now, If the payment is successful, a suitable message should be displayed and returned including the **remainingAmount** to be paid. The value of **isFullPayment** will also be updated accordingly.
[remainingAmount = premiumCharge – this.paidAmount]

[**Hint:** if the total paidAmount is equal to the premiumCharge, then isFullPayment should be set to true otherwise false**.]**

There is a method named **calculateDiscount()**. This method is used to calculate discount amounts based on payment status. If the payment is full then the premium member will get a 10% **discount** on the premium plan's charge, otherwise there would not be any discount. Here, if the discount is calculated successfully, the value of the **discountAmount** attribute will be updated and a suitable message will be displayed.

[Note: If isFullPayment == true then discountAmount = 10% of plan's price]

Create a method name **revertPremiumMember()** which calls the super class **resetMember()** method. Additionally, the value of **personalTrainer** is set to empty, **isFullPayment** is set to **false,** and **paidAmount**, and **discountAmount** are set to **zero**.

A method is required to **display** the details of the **PremiumMember** and it must have the same signature as the display method in the parent class. It will call the method in the super class to display **all the attributes of super class**. Additionally, it should display the value of **personalTrainer, paidAmount** and **isFullPayment.** Also, the remaining amount to be paid should be calculated and displayed with suitable annotation. Also, **discountAmount** will only be displayed, if the payment is done completely.

[**Hint:** remainingAmount = premiumCharge – this.paidAmount]

GUI Implementation – Part [25 Marks]

Your GUI should include the same components, but you have the **flexibility** to choose a different layout if it enhances aesthetics or usability. The GymGUI class must maintain an **ArrayList (not a simple array)** of the GymMember class, which will hold both RegularMember and PremiumMember **objects.**

[Note: Only one ArrayList should be created.]

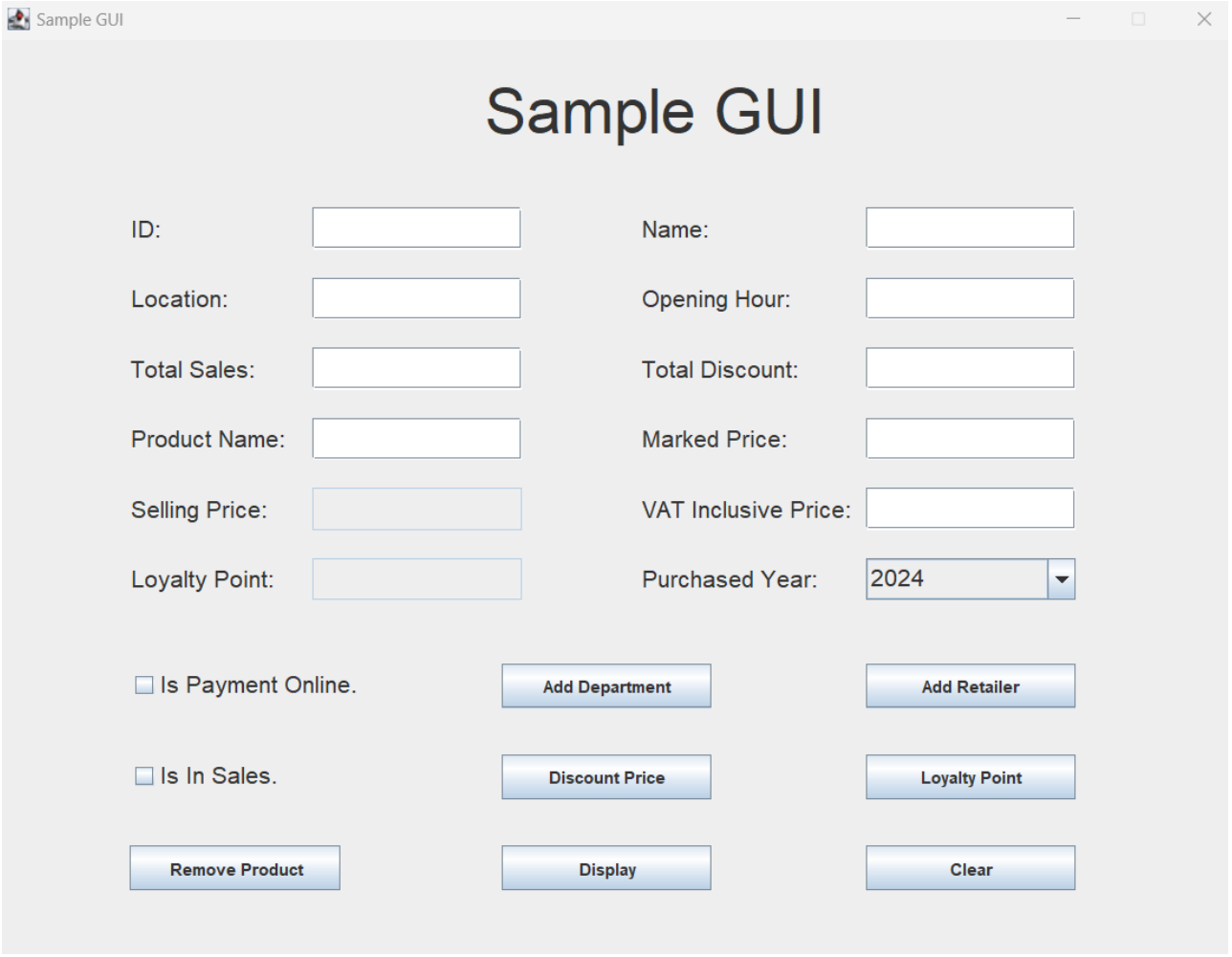
There should be following text fields for entering:

1. Id
2. Name
3. Location
4. Phone
5. Email
6. Gender
7. DOB
8. Membership Start Date
9. Referral Source
10. Paid Amount
11. Removal Reason
12. Trainer's Name

Also, there should be a **non-editable text field** for **regular plan price, premium plan charge** and **discount amount.**

The **gender** should be displayed as a **radio button.**

The **Plan (Basic, Standard, Deluxe)** attribute should be displayed as a **JComboBox** for plan selection for regular members only.



The GUI should have the following buttons to perform specific task:

1. Add a Regular Member

When this button is pressed, the input values from the text fields (ID, Name, Location, Phone, Email, Gender, DOB, Membership Start Date,Referral Source) will be used to create a RegularMember object. This object will then be added to an ArrayList of the GymMember class.

Note: Member ID duplication is prohibited. Each member, whether Regular or Premium, must have a unique Member ID]

2. Add a Premium Member

When this button is pressed, the input values from the **text fields** (ID, Name, Location, Phone, Email, Gender, DOB, Membership Start Date, Personal Trainer) will be used to create an object of Premium member and this object will then be added to an ArrayList of the GymMember class.

[Note: Member ID duplication is prohibited. Each member, whether Regular or Premium, must have a unique Member ID.]

3. Activate Membership

The **Member Id** should be entered in the GUI to activate membership. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system, then **call** the method to activate membership from GymMember class.

4. Deactivate Membership

The **Member Id** should be entered in the GUI to deactivate membership. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system, then **call** the method to deactivate membership from GymMember class.

5. Mark Attendance

First, the **Member Id** should be entered in the GUI to mark attendance. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system, then call the method to mark attendance from GymMember class.

[Note: if only active status of member is true, proceed with mark attendance]

6. Upgrade Plan

The Member Id is **entered,** and the required Plan **selected** in the GUI. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system, then **call** the method to upgrade plan from RegularMember class. Upgrading the plan is only possible if the member is active.

[Hint: An object of GymMember is cast as RegularMember.]

7. Calculate Discount

To calculate a discount, the Member Id is **entered**. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system, then **call** the method to calculate discount from PremiumMember class.

[Hint: An object of GymMember is cast as RegularMember.]

8. Revert Member

To remove a member, the MemberID is entered in the GUI. When this button is pressed the, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system and then **call** the method to revert member from both RegularMember and PremiumMember class.

[**Note: there should be two separate buttons for reverting the members of each class]**

9. Pay Due amount

To pay the due amount, the **MemberID , and amount to be paid** is entered in the GUI. When this button is pressed, the input value of Member Id is **compared** to the existing Member Id. If the entered Member Id is **valid** and **exists** in the system and then **call** the method to pay the due amount of the Premium gym member.

10. Display

When this button is pressed, the information of all the members of the respective class should be displayed in a separate frame/panel in the GUI.

11. Clear

When this button is pressed the entered values from text fields should be cleared and the check box should be unchecked.

12. Save to File

When this button is pressed all the member details stored in the arraylist should be written to file named “MemberDetails.txt” as shown in the figure below:

ID	Name	Location	Phone	Email	Membership Start Date	Plan	Price	Attendance	Loyalty Points	Active Status	Full Payment	Discount Amount	Net Amount Paid
1	John Doe	New York	1234567890	john@example.com	2023-01-01	1 month	4000	10	50.00	true	NA	NA	4000
2	Jane Smith	Los Angeles	0987654321	jane@example.com	2023-02-01	12 months	55000	15	150.00	false	true	5500.00	49500

*[**Hint:-** You can use this string format as reference and adjust accordingly: String.format("%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s %-10s %-15s %-10s %-15s %-15s %-15s\n", "ID", "Name", "Location", "Phone", "Email", "Membership Start Date", "Plan", "Price", "Attendance", "Loyalty Points", "Active Status", "Full Payment", "Discount Amount", "Net Amount Paid");]*

13. Read from File

When this button is pressed all the member details written in a text file should be read and displayed in separate frame/panel in an organized format.

Additional Information to be included:

- Retrieve the values from each text field by using the getText() method.
- For numeric variables, extract the text from the text field, convert it to an integer, and return the result as a whole number.
- Use try and catch blocks to handle potential NumberFomatException with that may occur during the conversion of the string to an integer or double.
- In case of successful operations, display an appropriate success message to the user using a message dialog box.
- In case of incorrect input, display an appropriate error message to the user using a message dialog box.

Report (44 marks)

Your report should describe the process of development of your classes with:

a. A **class diagram** for each of the classes and combined one showing relationship among them. **[5 marks]**

b. Pseudocode for methods of each classes
[Note:- don't include getter/setter methods] **[8 marks]**

c. A short description of what each method of given classes. **[8 marks]**

d. You should give evidence (through inspection tables and appropriate screenshots) of the following testing that you carried out on your program:

Test 1: Compile and run the program using command prompt/terminal. **[1 mark]**
Test 2: Adding regular member and premium member respectively. **[2 marks]**
Test 3: Test case for mark attendance and upgrade plan. **[4 marks]**
Test 4: Test case for calculate discount, pay due amount and reverting members.
[4 marks]
Test 5: Test case for saving and reading from a file. ` **[3 marks]**

e. The report should contain a section on error detection and error correction where you give examples and evidence of three errors encountered in your implementation. The errors (syntax, semantic or logical errors) should be distinctive and not of the same type. **[3 marks]**

f. The report should contain a conclusion, where you need to include the following things:
   - Evaluation of your work,
   - Reflection on what you learned from the assignment,
   - What difficulties do you encounter and
   - How you overcame the difficulties.

[4 marks]

The report should include a title page (including your name and ID number), a table of contents (with page numbers), an introduction part that contains a brief about your work, and a listing of the code (in an appendix). Marks will also be awarded for the quality of writing and the presentation of the report.

[3 marks]

Viva

Note: ***If a student would be unable to defend through VIVA his/her coursework, s/he might be penalized with 50% of total coursework marks.***

Marking Scheme

Marking criteria		Marks
A.	**Coding Part**	**55 Marks**
	1. Creating GymMember Class 2. Creating RegularMember Class 3. Creating PremiumMember Class 4. Creating GymGUI class 5. Program Style	5 Marks 10 Marks 10 Marks 25 Marks 5 Marks
B.	**Report Structure and Format**	**45 Marks**
	1. Class Diagram 2. Pseudocode 3. Method Description 4. Test-1 5. Test-2 6. Test-3 7. Test-4 8. Test-5 9. Error Detection and Correction 10. Conclusion 11. Overall Report Presentation/Formatting	5 Marks 8 Marks 8 Marks 1 Marks 2 Marks 4 Marks 4 Marks 3 Marks 3 Marks 4 Marks 3 Marks
	Total	**100 Marks**

Milestone 1 (Sunday, 9 March 2025)

- OOP Implementation, GUI and ArrayList

  → Implementation of Object Oriented Programming Principles such as Inheritance, Encapsulation, Polymorphism and Abstraction.

  → Student must perform given test cases to reflect whether the program is working properly or not.

  → Creating and initiliazing an arraylist to store newly created objects of the class, perform traversal to access and manipulate the objects.

  → Implementation of GUI components using AWT and SWING packages to create aesthetically pleasing and fully functional user interfaces

Milestone 2 (Sunday, 13 April 2025)

- Implementation of Event handling, Exception Handling

  → Use of AWT package to define specific event listeners and its corresponding handler methods to specify what happens when an event occurs.

  → Implementation of try-catch blocks to manage and handle exceptions.