LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

AY: 2024-25 Spring

Credit: 30

Student Name: EIJKEYAL PAKHRIN

London Met ID: 24*****49**

College ID: NP*******022**

Assignment Due Date: Friday, August 29, 2025

Assignment Submission Date: Friday, August 29, 2025

Word Count: 7961

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

JavaTurnitin.pdf

Islington College,Nepal

Document Details

Submission ID **trn:oid:::3618:110053264**

Submission Date

Aug 29, 2025, 11:56 AM GMT+5:45

Download Date

15 Pages

3,208 Words

18,469 Characters

The combined total of all matches, including overlapping sources, for each database.

Match Groups

17 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources

1% Publications

7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review
Aug 29, 2025, 11:58 AM GMT+5:45

8% Overall Similarity

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

0 Integrity Flags for Review

Match Groups

- **17** Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
1% Publications
7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
www.synergisticit.com		1%
2	Submitted works	
Swinburne University of Technology on 2025-01-10		1%
3	Internet	
thelistacademy.com		1%
4	Internet	
www.drawio.com		<1%
5	Submitted works	
Zambia Centre for Accountancy Studies on 2023-01-11		<1%
6	Internet	
cornerskills.com		<1%
7	Internet	
boxoflearn.com		<1%
8	Internet	
www.coursehero.com		<1%
9	Internet	
teletalkbd.com		<1%
10	Submitted works	
Sim University on 2024-10-19		<1%

Contents

1.INTRODUCTION

Project overview

The aim of this Coursework is to implement a real-world problem scenario using objectoriented programming concepts in java. The program manages the management system of an art gallery, focusing on handling visitors with different types of gallery passes, tracking their visits, artwork purchases, reward and cancellations. The program consists of a parent class, ArtGalleryVisitor and two subclasses, standard and elite visitor to represent the different visitor types. However, a GUI(Graphical user Interface) class, which allows interactive management of visitors and their activities.

Purpose and Objectives

The main objectives of this coursework are:

I. To use object-oriented programming concepts such as inheritance, abstraction and polymorphism.

II. To manage visitor data ,visitor personal details like purchases, rewards point and cancellations.

III. To ensure the specific behaviors for different visitor types like standard and elite for standard it has less facilities like eligible for discount and earn rewards but in elite visitors it have more facilities like they can assigned a personal art advisor and has access to exclusive events.

By creating interactive GUI interface I t allows users to interactively add visitors, log visits, purchase art , calculate discounts and rewards, cancel purchases, generate bills and save or read visitor details from a file.

To manage and store visitor’s data efficiently using an ArrayList of ArtGalleryVisitor objects.

Scope of the work

The projects focuses on developing a system to manage visitors of an art gallery, including storing their details, visit history, purchases and reward points. This system differentiates between elite and standard visitors , providing specific feature on the basis of class such as discount, upgrades, personal art advisior and exclusive event access for elite class visitors. It also handles the cancellations with refund calculations and tracks purchase limits. A graphical user interface allows user to interactively add visitors ,log visits, artwork purchase, generate bills, and save or read visitors details from a text file. Overall, this project demonstrates the practical application of object-oriented programming concepts in java while providing a user-friendly interface for gallery management.

Technologies Used

Java

Java is a programming language which was developed by sun Microsystems 1995. It was developed from humble beginnings to power a large share of today's digital world by providing the reliable platform. Which allow user to provide new innovative products and digital services designed for the future to rely on java as well (Java/Oracle). It is used most because of its large online community of developers and it works diligently to enrich java functions. Java programming is mostly used due to its "write once , run anywhere" functionality. It is also platform-independent language which means it can used to develop end-to-end mobile or desktop applications and run different servers and operating systems (SynergisticIT).

Figure 1 Java

BlueJ

BlueJ is an integrated development environment (IDE) for the java programming language, it developed mainly for educational purposes and suitable for small scale software development. It runs with the help of java development kit (JDK). It gives simple interface to navigate, especially for beginners allows to visually see classes and their relationships. This helps to manage ArtGalleryVisitor system and debugging easier to detect and correct errors in the code. (BlueJ)

Figure 2 BlueJ

MS-word

Microsoft word is a word processing software developed by Microsoft. I used word to prepare report and it allowed me to organize all sections , format text, insert tables, figures and screenshots to make professional looking report (Microsoft). Its features such as headings, bullet points and page numbering made it easier to make structured report according to the report format and it exports as PDF for submission. It provides tools for professional formatting, spell-checking, and creating a clean, readable document.

Figure 3 MS-Word

Balsamiq

Balsamiq wireframes is graphical user interface website wireframe builder application. It allows user to design or show creativity pre built widgets using drag and drop (Balsamiq). I used balsamiq to design the GUI interface wireframe before implementation to make sure how it looked. It provides a quick way to reference interface layout showing how users will interact with system (wikipedia).

Figure 4 Balsamiq

Draw.io

Draw.io is technology used for drawing or building diagramming applications and the world's most widely used browser based end user diagramming software. It provides us to create flowcharts, diagrams, maps and many more but I used draw.io to create class diagrams, showing relationships, inheritance and interactions between classes. It helped me to visualize the structure for documentation purposes (Draw.io).

Figure 5 Draw.io

2.GUI wireframe

This GUI wireframe contains visual representations of the program's interface before actual implementation. It helps to plan the layout and placements of components and show how it will look and interact.

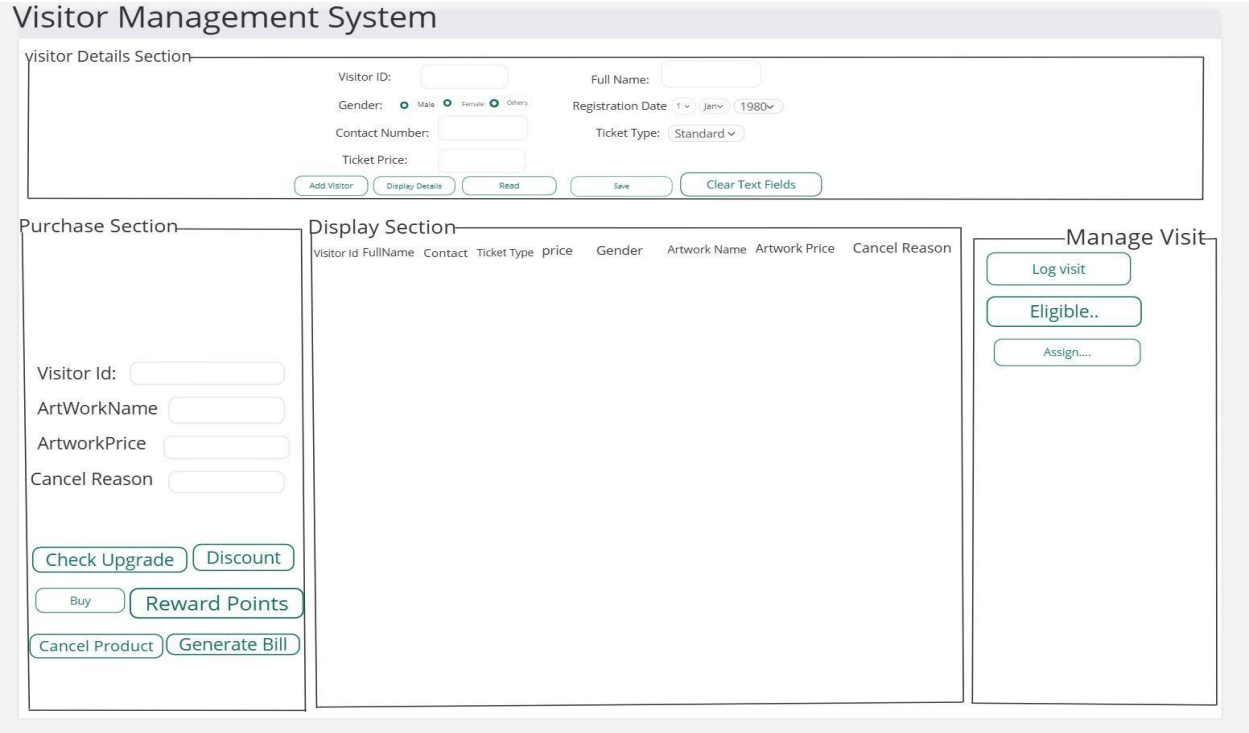
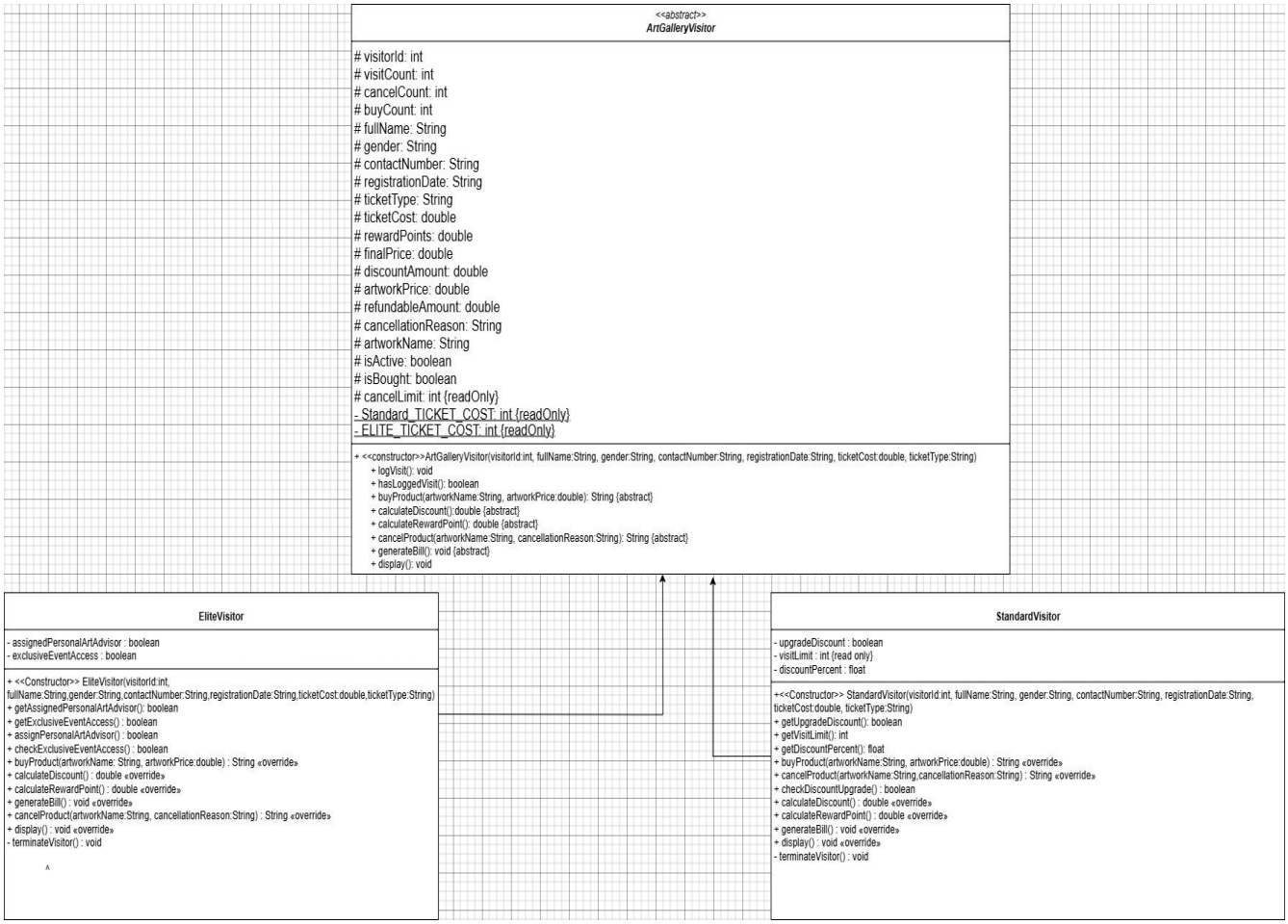


Figure 6 GUI wireFrame of Visitor Management System

The GUI maintains a single ArrayList of ArtGalleryVisitor objects, which stored both standard and elite visitors. It includes text fields for visitor ID, Full Name, Contact Number ,Ticket Price (non-editable), Artwork Name, Artwork Price and cancellation Reason, along with radio buttons for gender sections and combo drop down box for registration date and ticket type.

The GUI provides buttons to add visitors ensures the unique id, log visits, buy products, assign a personal art advisor for elite visitors, check upgrades for standard visitors, calculate discount and rewards points, cancel products, generate bills (displayed in GUI and saved as .txt), display visitor details, clear fields and save/read visitor data from file. Each function interacts with visitor objects ensuring proper methods calls.

3.Class Diagram



4.Pseudo code

METHOD addvisitor:

GET visitor ID from input

CHECK visitor ID is empty OR not a number OR negative

IF yes SHOW error message

STOP

READ full name from input

IF full name is empty OR contains numbers/symbols

SHOW error message

STOP

READ contact number from input

IF contact number is empty OR too long OR not a number OR negative

SHOW error message

STOP

GET registration date from day, month, year from combo box

GET gender from radio buttons

CHECK IF no gender is selected

SHOW error message

STOP

READ ticket type and ticket cost

IF ticket cost is empty OR not a number

SHOW error message

STOP

CHECK if visitor ID already exists

IF exists

SHOW "Duplicate Visitor ID"

STOP

IF ticket type is Standard

CREATE StandardVisitor

ELSE

CREATE EliteVisitor

TAKE artwork name and artwork price FROM textField

IF artwork price not a number

```
        SHOW error message
        STOP

    GET cancellation reason
    ADD visitor to visitorList
    SHOW "Visitor added successfully"
END method

logVisitor()
METHOD logVisitor
TAKE visitor ID from input
    IF visitor ID is not a number
        SHOW error message
        STOP

    SET found to false

    FOR each visitor in visitorList
        IF visitor ID matches
            CALL visitor.logVisit()  // add 1 to visit count
```

```
            SET isActive to true
            SHOW "Visit logged successfully"
            SET found to true
            STOP loop

    IF found is still false
        SHOW "Visitor ID not found"
END Method
DisplayDetails()
Execute displayDetails
    CLEAR the table

    CHECK visitor in visitorList
        IF visitor exists
            ADD visitor details to table
                (Visitor ID, Name, Gender, Contact Number,
                 Registration Date, Ticket Type, Ticket Cost,
                 Artwork Name, Artwork Price, Cancellation Reason)

    IF any error occurs
```

```
        SHOW "There is no data to show"
END METHOD

ClearFields()

METHOD clearFields
    CLEAR visitor ID field
    CLEAR full name field
    CLEAR contact number field
    CLEAR gender selection
    RESET ticket type selection to default
    CLEAR ticket cost field
    CLEAR artwork name field
    CLEAR artwork price field
    CLEAR cancellation reason field
    RESET day, month, and year selection to default
END METHOD


buy()

METHOD buy
    READ visitor ID from input
    IF visitor ID is empty OR negative
```

```
    SHOW error message and EXIT
  ENDIF

  IF visitor is not active (hasn't logged visit)
    SHOW message to log visit first and EXIT
  ENDIF

  TAKE artwork name and price from input
  IF artwork name OR price is empty
SHOW error message and EXIT
  ENDIF

  FOR each visitor in visitorList
    IF visitor ID matches
      IF visitor has not logged visit
        SHOW message to log visit first and EXIT
      ENDIF
      CALL visitor.buyProduct(artworkName, artworkPrice)
      SET visitor as bought
      SHOW purchase confirmation message
      EXIT loop
```

```
        ENDIF
    ENDFOR

    IF visitor not found
        SHOW visitor not found message
    ENDIF

    REFRESH visitor details table
END METHOD
AssignPersonalArtAdvisor()
METHOD assignPersonalArtAdvisor
    TAKE visitor ID from input
    IF empty, SHOW error and EXIT

    SEARCH visitorList for visitor with matching ID
        IF visitor found
            IF visitor is EliteVisitor
                ASSIGN personal art advisor
                SHOW success message
            ELSE
```

```
            SHOW "Not an Elite Visitor" message
        ELSE
            SHOW "Visitor ID not found" message
END METHOD


checkUpgrade()
METHOD checkUpgrade
    READ visitor ID from input
    IF invalid, SHOW error and EXIT

    SEARCH visitorList for visitor
        IF visitor found
            IF visitor has not bought, SHOW message and EXIT
            IF visitor is StandardVisitor
                CHECK discount upgrade
                SHOW appropriate message
            ELSE
                SHOW "Only Standard Visitors can upgrade"
        ELSE
            SHOW "Visitor ID not found"
```

```
END METHOD

. calculateDiscount()


FUNCTION calculateDiscount
    GET visitor ID from input
    IF invalid, SHOW error and EXIT


    FIND visitorList for visitor
        IF visitor found
            IF visitor has not bought, SHOW message and EXIT
            CALCULATE discount and final price
            SHOW discount and final price
        ELSE
            SHOW "Visitor ID not found"
END METHOD


calculateRewardPoints()
METHOD calculateRewardPoints
    GET visitor ID from input
    IF empty or invalid, SHOW error and EXIT
```

```
    SEARCH visitorList for visitor
        IF visitor found
            CALCULATE reward points
            SHOW reward points
        ELSE
            SHOW "Visitor ID not found"
END METHOD


cancelProduct()
METHOD CancelProduct
    TAKE visitor ID, artwork name, cancellation reason from input
    IF any field empty, SHOW error and EXIT

    SEARCH visitorList for visitor
        IF visitor found
            CALL cancelProduct method (depends on visitor type)
            SHOW cancellation message
        ELSE
            SHOW "Visitor ID not found"
```

```
END FUNCTION


 generateBill()
FUNCTION generateBill
    READ visitor ID from input
    IF empty or invalid, SHOW error and EXIT


    SEARCH visitorList for visitor
        IF visitor found
            IF visitor has not bought, SHOW message and EXIT
            BUILD bill string (visitor info, artwork, discount, final price)
            DISPLAY bill in GUI
            EXPORT bill to .txt file
        ELSE
            SHOW "Visitor ID not found"
END FUNCTION
Save()
FUNCTION save
    IF visitorList is empty THEN
        SHOW message "No visitor data to save!"
```

```
        EXIT FUNCTION
    END IF

    CREATE file "VisitorDetails.txt"

    TRY
        OPEN file for writing
        WRITE header line (ID, Name, Gender, Contact, Date, Ticket Type, Cost)        FOR EACH visitor IN visitorList
            WRITE visitor details in formatted line
        END FOR

        SHOW message "Visitor details saved successfully!"
    CATCH any exception
        SHOW error message
    END TRY
END FUNCTION

Read()

FUNCTION read
    CREATE file "VisitorDetails.txt"
```

```
    IF file does not exist OR file is empty THEN
        SHOW message "No visitor data added yet!"
        EXIT FUNCTION
    END IF

    TRY
        OPEN file for reading
        CREATE text area for display
WHILE NOT end of file
            READ line from file
            APPEND line to text area
        END WHILE
        SHOW text area in scrollable message dialog
        PRINT "Visitor data loaded successfully"
    CATCH any exception
        SHOW error message
        PRINT error message
    END TRY
END FUNCTION
```

5.Method Description

Class Name	Method Name	Description
ArtGalleryVisitor	getVisitorId()	Returns the unique ID assigned to the visitor. Used to identify each visitor in the system.
	getFullName()	Returns the visitor's full name as entered during registration.
	getGender()	Returns the gender of the visitor.
	getContact Number()	Returns the contact number of the visitor.
	getRegistration Date()	Returns the registration date of the visitor in the system
	getTicketType()	Returns the type of ticket purchased (Standard or Elite).
	getTicketCost()	Returns the cost of the ticket purchased by the visitor
	isBought()	Checks if the visitor has already bought an artwork. Returns true if purchased.
	setBought(boolean)	Sets the purchase status for the visitor. Used to prevent multiple purchases of the same artwork without logging a visit.
	logVisit()	Records a visit for the visitor by incrementing the visitCount. Marks the visitor as active for future purchase actions
	getArtworkName()	Returns the name of the artwork purchased by the visitor
	getArtworkPrice()	Returns the price of the purchased artwork
	getFinalPrice()	Returns the final price of purchased artwork after discount.
	getCancellation Reason()	Returns the reason provided by the visitor for cancelling a purchase.
	buyProduct(String, double)	Abstract method. Must be implemented in subclasses to handle artwork purchase, including setting the artwork name, price, discount, reward points, and purchase status.
	calculateDiscount()	Abstract method. Must be implemented in subclasses to calculate discount based on visitor type.
	calculateReward Point()	Abstract method. Must be implemented in subclasses to calculate reward points based on purchase value.
	cancelProduct(String, String)	Abstract method. Must be implemented in subclasses to cancel a purchase, deduct reward points, and update visitor status.

	generateBill()	Abstract method. Must be implemented in subclasses to print a detailed invoice including artwork, price, discount, and final cost.
	display()	Displays all visitor information, including personal details, ticket info, visit count, purchase status, discounts, final price, and reward points.

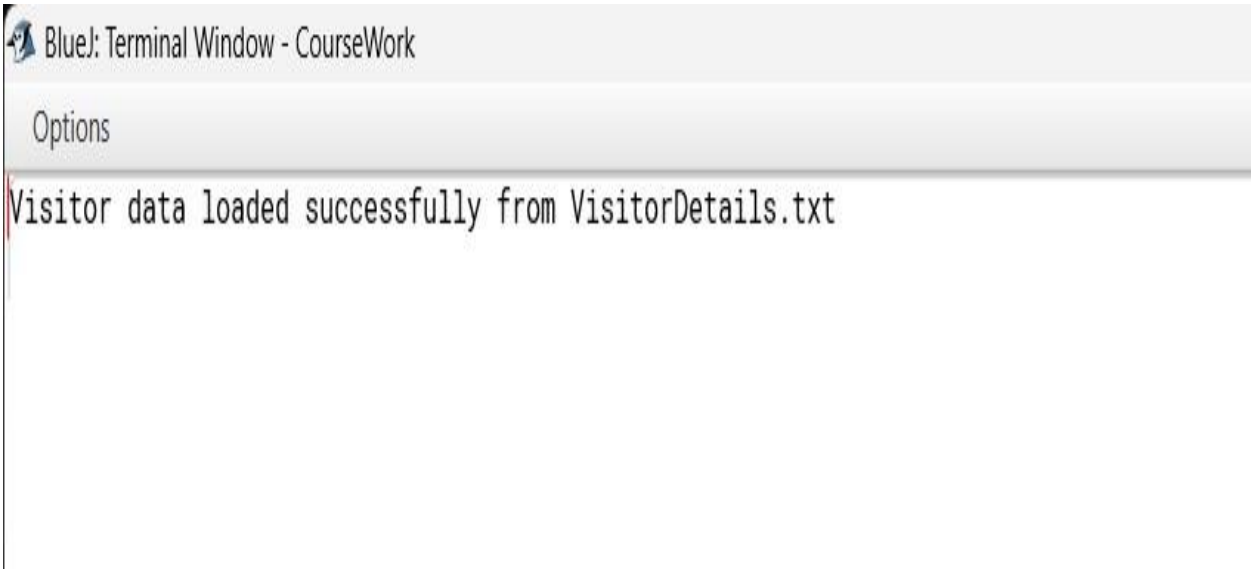
StandardVisitor	getUpgradeDiscount()	Returns whether the visitor is eligible for a discount upgrade after a certain number of visits
	getVisitLimit()	Returns the maximum number of visits allowed before account termination or restriction.
	getDiscountPercent()	Returns the current discount percentage applied to the visitor's purchase.
	buyProduct(String, double)	Handles artwork purchase for standard visitors. Checks if visitor has logged a visit, prevents multiple purchases without a new visit, applies discount, and calculates reward points.
	checkDiscountUpgrade()	Checks if the visitor's visit count qualifies them for a discount upgrade and updates the discount percentage
	calculateDiscount()	Calculates discount and final price for standard visitors based on eligibility and ticket type.
	calculateRewardPoint()	Calculates reward points earned based on the final price after discount.
	generateBill()	Prints a detailed bill for the standard visitor, showing artwork purchased, original price, discount applied, final price, and reward points earned.
	cancelProduct(String ArtworkName, String CancelReason)	Cancels a purchase for a standard visitor. Updates reward points, resets purchase status, records cancellation reason, and checks if visitor exceeds visit limits to terminate the account if needed.
	display()	Displays all standard visitor details, including discount eligibility, visit counts, ticket info, purchased artwork, and financial details

EliteVisitor	getAssignedPersonalArtAdvisor()	Returns true if a personal art advisor has been assigned to the visitor.
	getExclusiveEventAccess()	Returns true if the visitor has access to exclusive events, granted when advisor is assigned.
	assignPersonalArtAdvisor()	Assigns a personal art advisor to the visitor if reward points exceed 5000. Updates visitor status accordingly.
	checkExclusiveEventAccess()	Grants or denies access to exclusive events based on advisor assignment.
	buyProduct(String, double)	Handles artwork purchase for elite visitors. Applies higher discount, updates reward points, sets purchase status, and generates billing info.
	calculateDiscount()	Calculates a 40% discount for elite visitors and sets the final price accordingly.
	calculateRewardPoint()	Calculates reward points for elite visitors at a rate of 10 points per rupee spent on discounted price.
	generateBill()	Prints a detailed bill including discount, reward points, and final purchase price for elite visitors.
	cancelProduct(String ArtworkName, StringCancelReason)	Cancels a purchased artwork with a 5% cancellation fee, updates reward points, resets purchase status, records cancellation reason, and terminates account if visit limits are exceeded.
	display()	Displays all elite visitor details including personal art advisor assignment, exclusive event access, ticket info, purchased artwork, discount, final price, and reward points

6.Testing

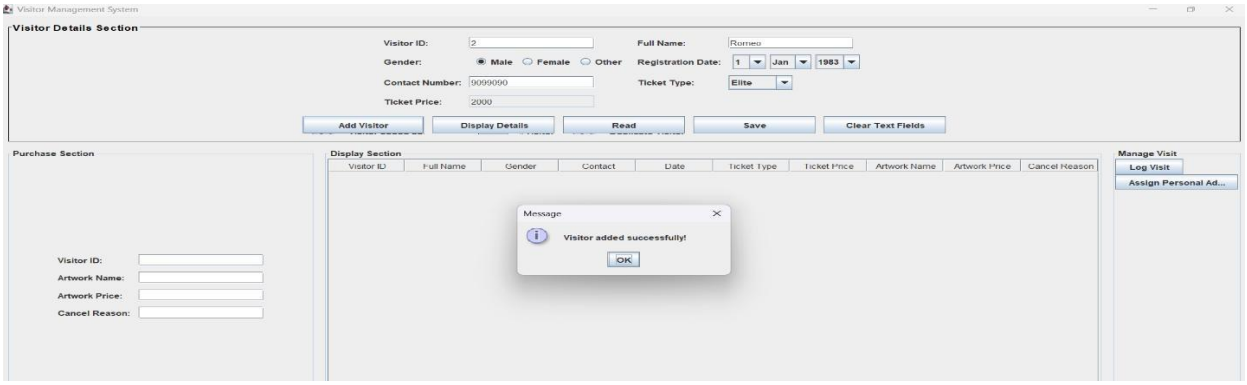
Test 1: Compile and run the program using command prompt/terminal

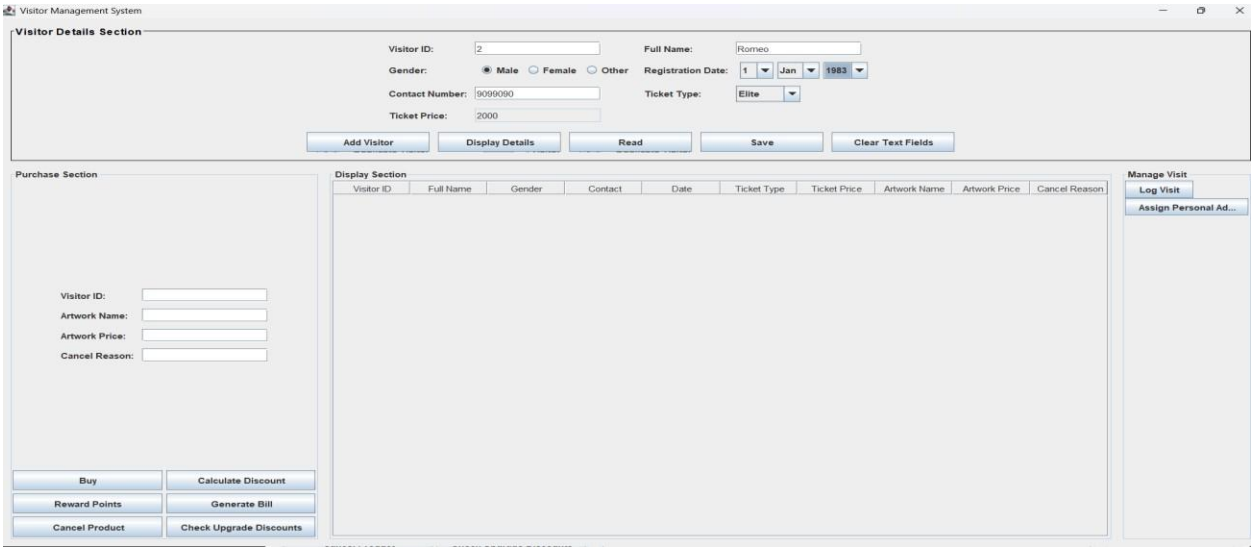
Objective	To run Java Program.
Action	Open in terminal and run Program
Expected Output	It runs without errors.
Actual Output	Program compiles successfully
Conclusion	It appears on terminal message

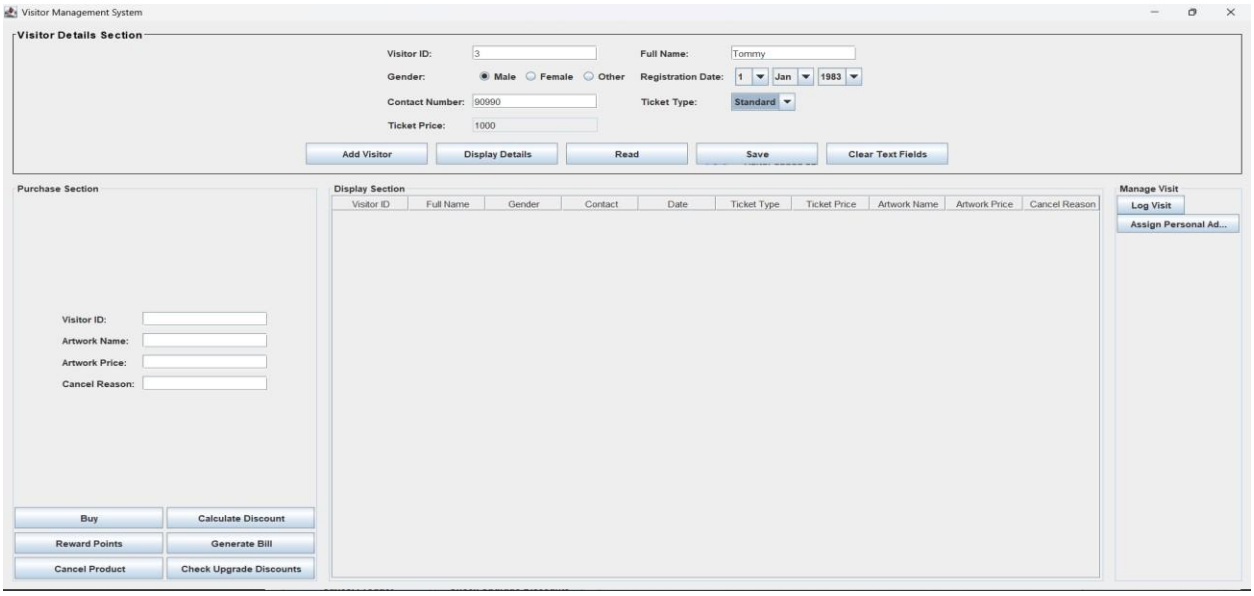


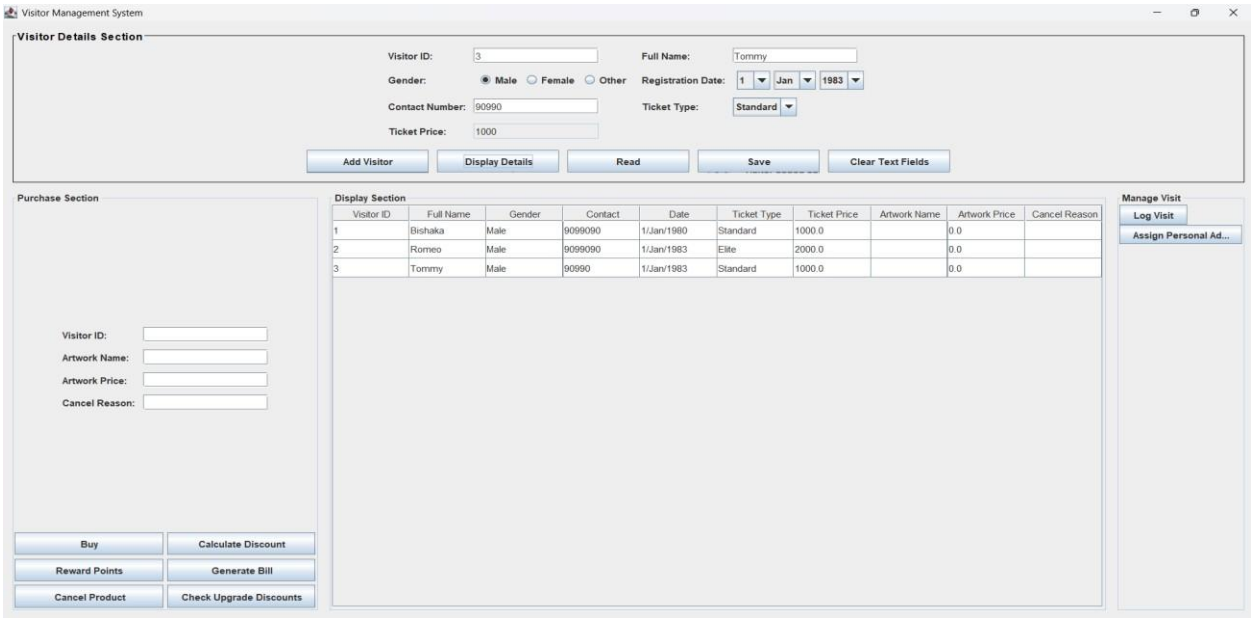
Test 2: Adding Standard and Elite visitors respectively.

Objective	To verify that the program correctly adds both standard and Elite visitors and display them in visitor List.
Action	Open and add visitor details in both visitors.
Expected Result	Both visitors are successfully added and display.
Actual Result	Successfully added both visitor along with different unique id.
Conclusion	The program handles the addition of both standard and elite visitors and display


Visitor Management System
Visitor Details Section
Visitor ID: 2
Full Name: Romeo
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 9099090
Ticket Type: Elite
Ticket Price: 2000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID:
Artwork Name:
Artwork Price:
Cancel Reason:
Display Section
Visitor ID
Full Name
Gender
Contact
Date
Ticket Type
Ticket Price
Artwork Name
Artwork Price
Cancel Reason
Manage Visit
Log Visit
Assign Personal Ad...
Message
Visitor added successfully!
OK

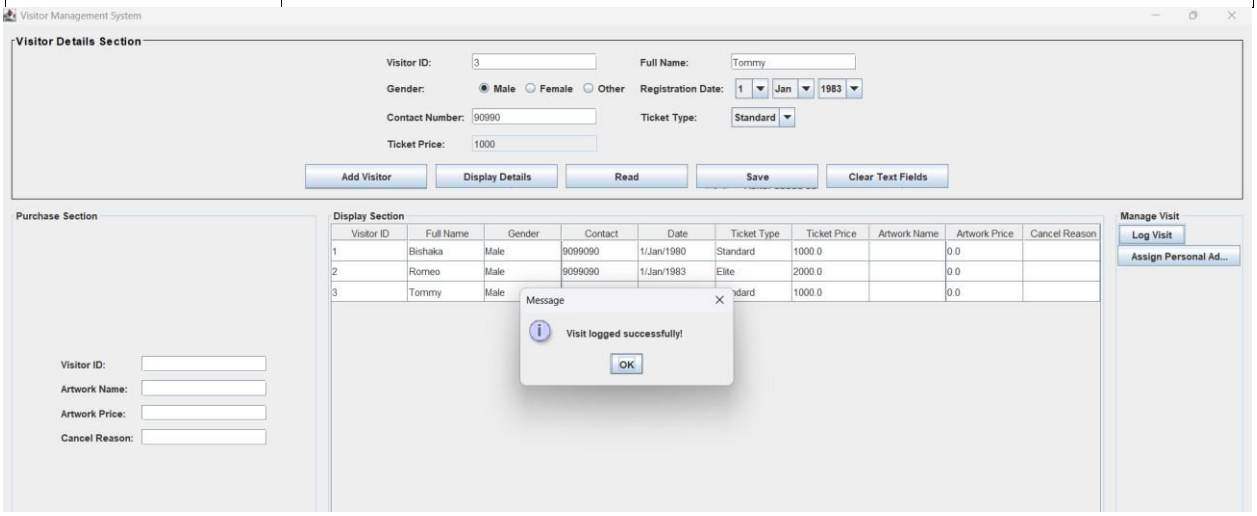

Visitor Management System
Visitor Details Section
Visitor ID: 2
Full Name: Romeo
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 9099090
Ticket Type: Elite
Ticket Price: 2000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID:
Artwork Name:
Artwork Price:
Cancel Reason:
Buy
Calculate Discount
Reward Points
Generate Bill
Cancel Product
Check Upgrade Discounts
Display Section
Visitor ID
Full Name
Gender
Contact
Date
Ticket Type
Ticket Price
Artwork Name
Artwork Price
Cancel Reason
Manage Visit
Log Visit
Assign Personal Ad...

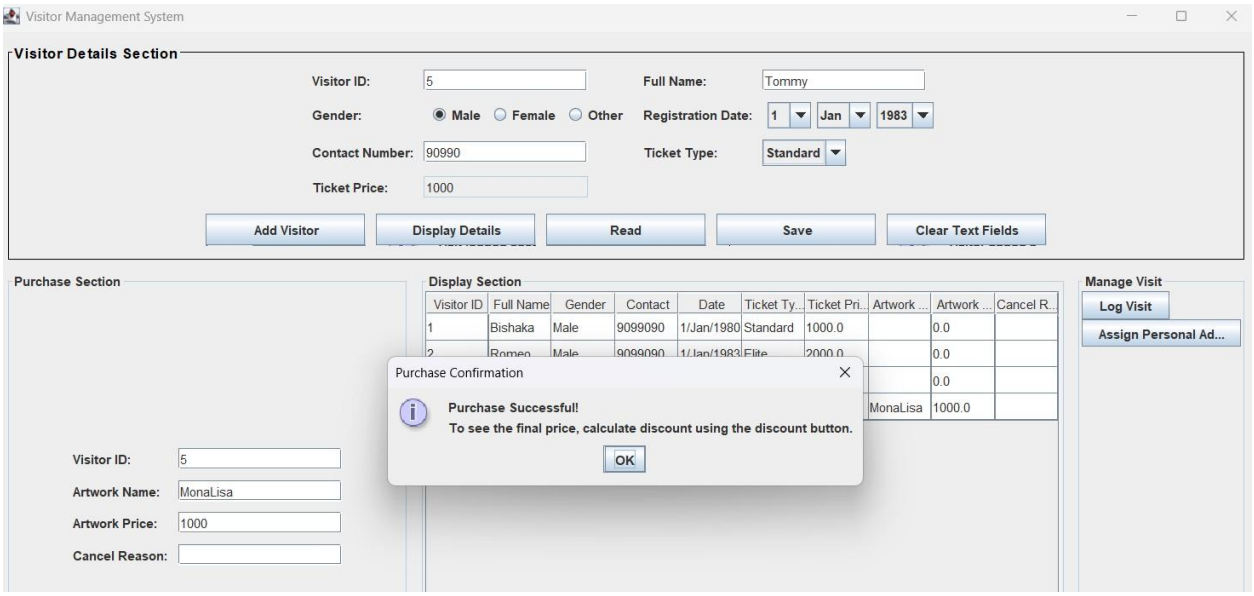

Visitor Management System
Visitor Details Section
Visitor ID: 3
Full Name: Tommy
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 90990
Ticket Type: Standard
Ticket Price: 1000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID:
Artwork Name:
Artwork Price:
Cancel Reason:
Buy
Calculate Discount
Reward Points
Generate Bill
Cancel Product
Check Upgrade Discounts
Display Section
Visitor ID
Full Name
Gender
Contact
Date
Ticket Type
Ticket Price
Artwork Name
Artwork Price
Cancel Reason
Manage Visit
Log Visit
Assign Personal Ad...

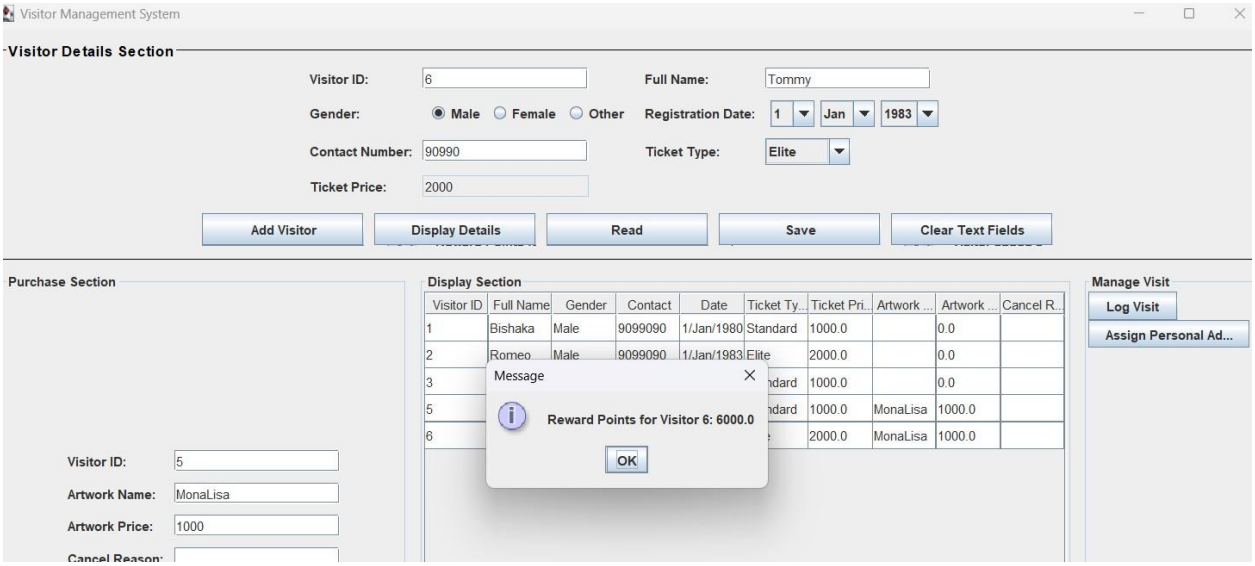


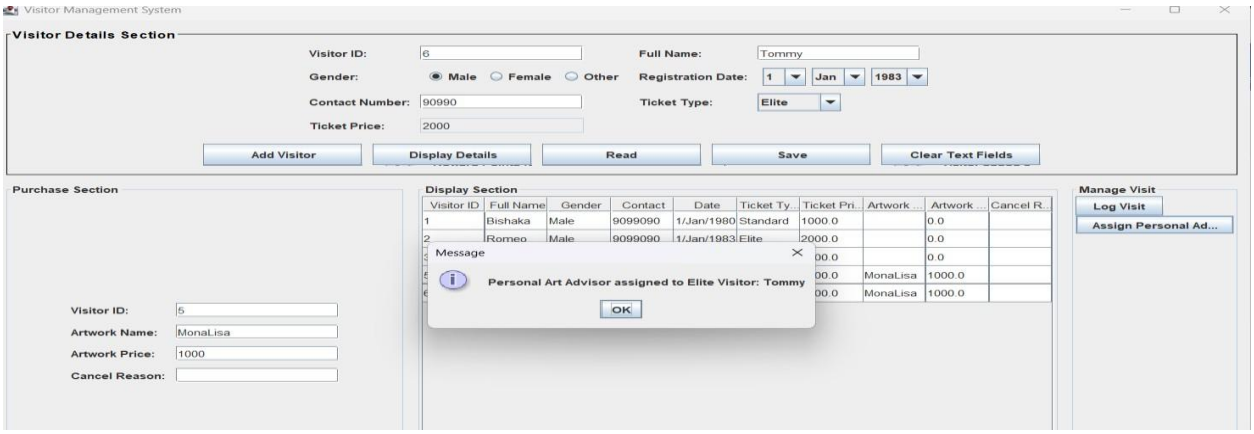
Test 3: Log Visit, Buy Product, Cancel Product, Assign Personal Art Advisor.

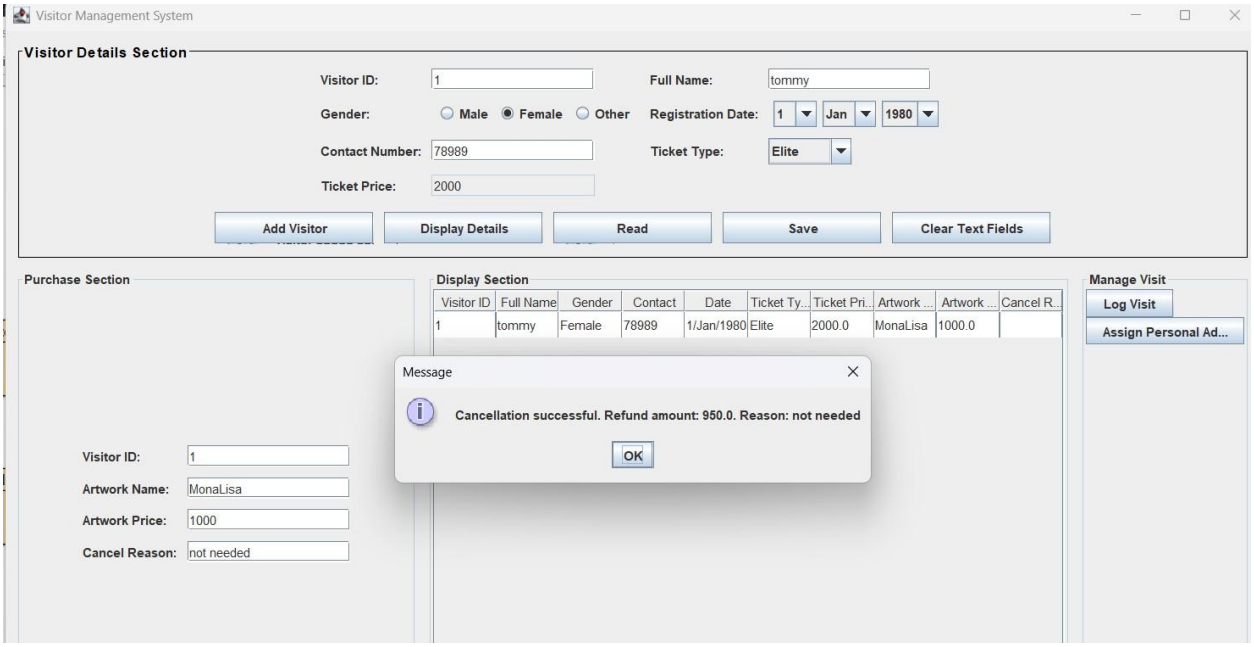
Objective	To verify the program runs correctly logs visits, make purchases , cancels products and assigns personal art advisors.
Action	Select visitor, log visit, buy product, cancel it, assign advisor.
Expected Output	All actions processed successfully with correct updates.
Actual Output	Actions completed successfully, messages displayed, data updated.
Conclusion	Visitor actions work as expected.




Visitor Management System
Visitor Details Section
Visitor ID: 5
Full Name: Tommy
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 90990
Ticket Type: Standard
Ticket Price: 1000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID: 5
Artwork Name: MonaLisa
Artwork Price: 1000
Cancel Reason:
Display Section
Visitor ID | Full Name | Gender | Contact | Date | Ticket Ty... | Ticket Pri... | Artwork ... | Artwork ... | Cancel R...
1 | Bishaka | Male | 9099090 | 1/Jan/1980 | Standard | 1000.0 | | 0.0 |
2 | Romeo | Male | 9099090 | 1/Jan/1983 | Elite | 2000.0 | | 0.0 |
0.0
MonaLisa | 1000.0
Purchase Confirmation
Purchase Successful!
To see the final price, calculate discount using the discount button.
OK
Manage Visit
Log Visit
Assign Personal Ad...

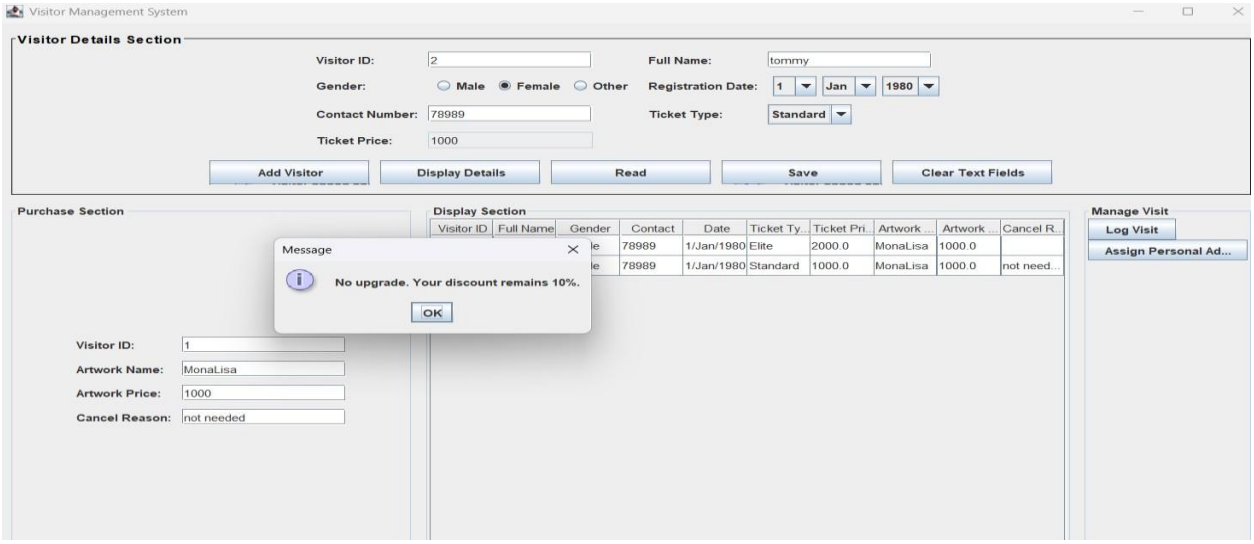

Visitor Management System
Visitor Details Section
Visitor ID: 6
Full Name: Tommy
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 90990
Ticket Type: Elite
Ticket Price: 2000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID: 5
Artwork Name: MonaLisa
Artwork Price: 1000
Cancel Reason:
Display Section
Visitor ID | Full Name | Gender | Contact | Date | Ticket Ty... | Ticket Pri... | Artwork ... | Artwork ... | Cancel R...
1 | Bishaka | Male | 9099090 | 1/Jan/1980 | Standard | 1000.0 | | 0.0 |
2 | Romeo | Male | 9099090 | 1/Jan/1983 | Elite | 2000.0 | | 0.0 |
3 | 1000.0 | | 0.0 |
5 | 1000.0 | MonaLisa | 1000.0 |
6 | 2000.0 | MonaLisa | 1000.0 |
Message
Reward Points for Visitor 6: 6000.0
OK
Manage Visit
Log Visit
Assign Personal Ad...


Visitor Management System
Visitor Details Section
Visitor ID: 6
Full Name: Tommy
Gender: Male Female Other
Registration Date: 1 Jan 1983
Contact Number: 90990
Ticket Type: Elite
Ticket Price: 2000
Add Visitor
Display Details
Read
Save
Clear Text Fields
Purchase Section
Visitor ID: 5
Artwork Name: MonaLisa
Artwork Price: 1000
Cancel Reason:
Display Section
Visitor ID | Full Name | Gender | Contact | Date | Ticket Ty... | Ticket Pri... | Artwork ... | Artwork ... | Cancel R...
1 | Bishaka | Male | 9099090 | 1/Jan/1980 | Standard | 1000.0 | | 0.0 |
2 | Romeo | Male | 9099090 | 1/Jan/1983 | Elite | 2000.0 | | 0.0 |
0.0
MonaLisa | 1000.0
MonaLisa | 1000.0
Message
Personal Art Advisor assigned to Elite Visitor: Tommy
OK
Manage Visit
Log Visit
Assign Personal Ad...



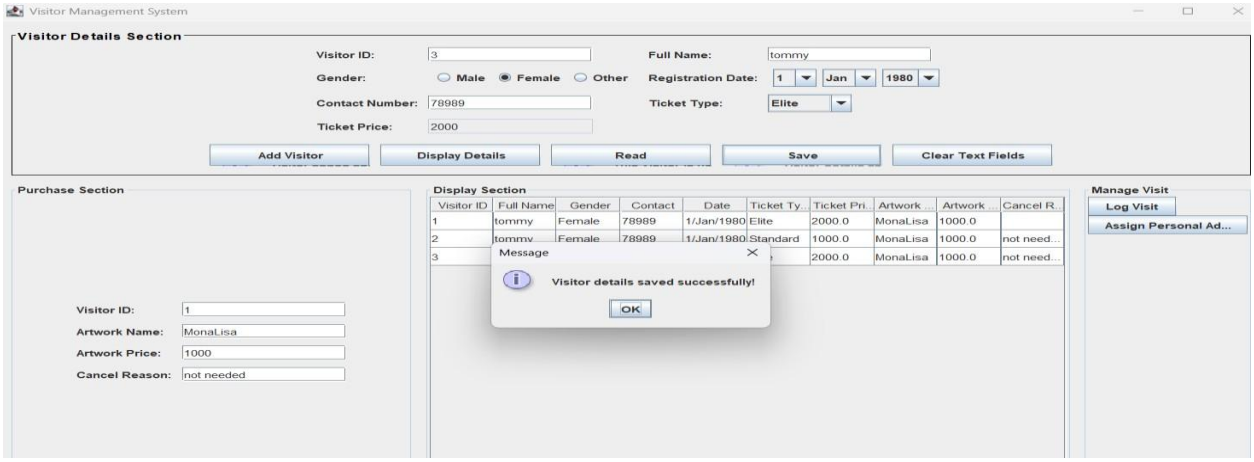
Test 4: Check Discount & Reward Points

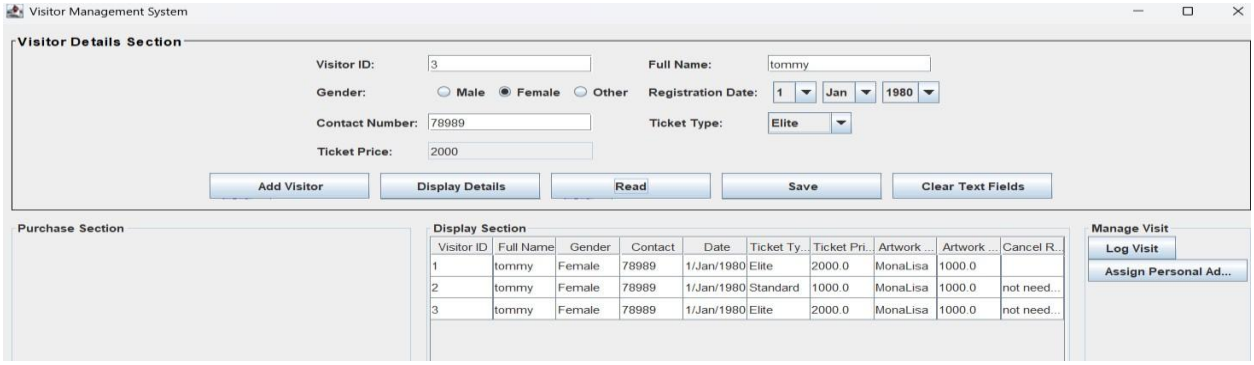
Objective	To verify discount eligibility , rewards points and calculate discount.
Action	Select Elite Visitor, check discount, buy product, check reward points.
Expected Output	Discount applied correctly and also reward point is calculate.
Actual Output	It calculates correctly.
Conclusion	Discount and reward points feature works correctly.

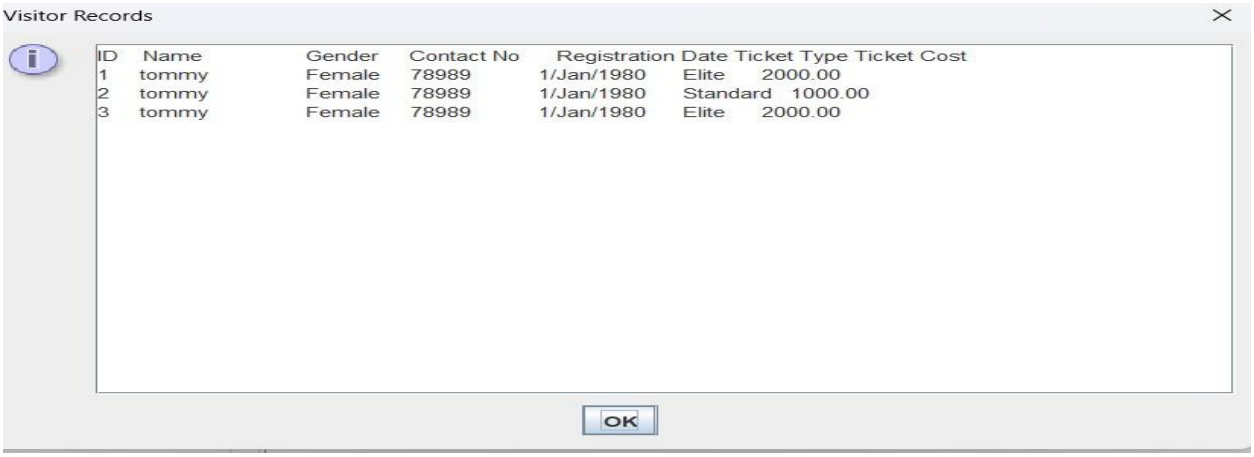


Test 5: Save and Load Data from File

Objective	To save and read data from visitor data.
Action	Perform actions save to file, close program and reload data
Expected Output	Data saved and read correctly with all updates.
Actual Output	All data loaded successfully and matches saved data.
Conclusion	It works as expected with data persistence.

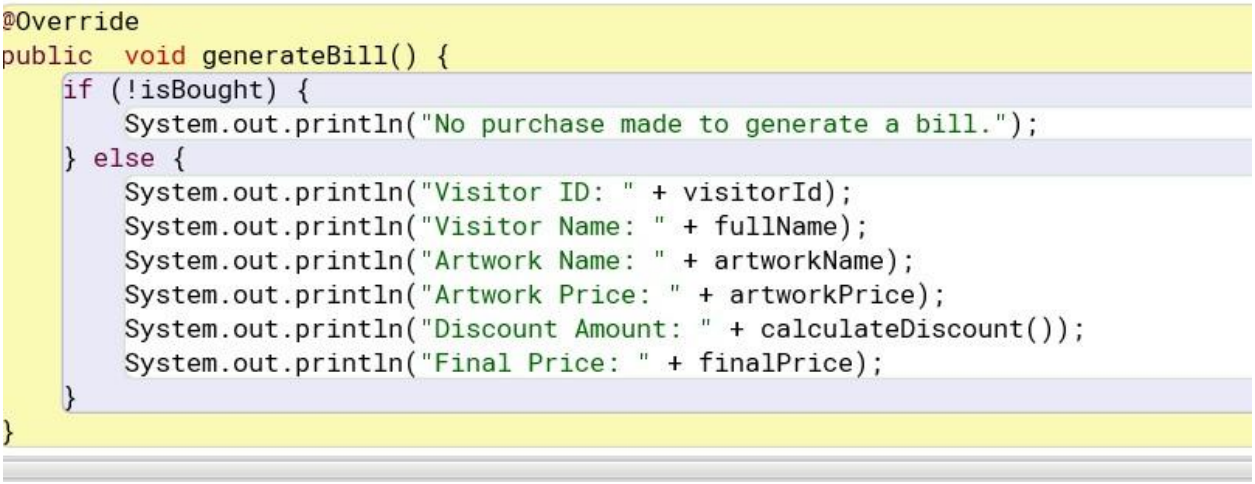






7.Error Detection and Correction

Error No:	Type of Error	What happened	Error message Evidence	correction
1	Syntax Error	Missed semicolon at the end of a statement in StandardVisitor class.	error: ';' expected	Added the missing semicolon at the end of the statement.



Error No:	Type of Error	What Happened	Error Message	Correction
2	Semantic error	Used wrong data type while parsing visitor ID from text field (double instead of int).	error: incompatible types: possible lost conversion from double to int	Changedparsing to Integer.parseInt()to match the int data type of visitor ID

Error No:	Type of Erro	Description/what happened	Error message/Evidence	Correction
3	Logical Error	Allowed purchase without checking if the visitor had logged a visit. The program displayed "Purchase Successful" even if visit was not logged.	program ran but gave wrong result (incorrect purchase allowed)	(!visitor.hasLoggedVisit()) to prevent purchase if visit not logged.

```
            }

            double discount = visitor.calculateDiscount();
            double finalPrice = visitor.getFinalPrice(); // getter in parent class

            JOptionPane.showMessageDialog(null,
                "Discount Amount: " + discount + "\nFinal Price: " + totalPrice);
            break;
        }
    }

    if (!found) {
        JOptionPane.showMessageDialog(null, "Visitor ID not found.");
    }
```

```
            }

            double discount = visitor.calculateDiscount();
            double finalPrice = visitor.getFinalPrice(); // getter in parent class

            JOptionPane.showMessageDialog(null,
                "Discount Amount: " + discount + "\nFinal Price: " + finalPrice);
            break;
        }
    }
```

8.Conclusion

Evaluation of Work:

The Art Gallery Visitor Management System was successfully implemented with all the required functionalities, including implementing logic to the buttons like addVistors, logvVisitors, calculateDiscount etc. I think the program meets the project requirements and performs as expected most of the scenario if there is any mistake ,errors comes from system I would be responsible to fix the issues as much as I can.

Reflection and learnings:

Throughout the assignment , I learned how to implement the object-oriented programming (oop) concepts such as inheritance, abstraction, polymorphism and abstract classes in java. I also have gain practical experienced in handling the GUI components with Swing, managing files for data, storage and debugging different types of errors. This coursework improved my understanding of logic design and data management in real-life scenario.

Challenges I Faced:

I ran into a few challenges while working on this project. One was handling errors when reading from and writing to files, especially making sure the program didn't crash if a file didn't exist or was empty. Another challenge was getting the calculations for reward points, discounts, and final prices to work correctly. I also had some difficulty making sure the GUI buttons behaved as expected—updating visitor information properly and preventing duplicate purchases.

How I Overcame These Difficulties:

I managed to solve these problems in a few ways. For file-related issues, I used trycatch blocks to handle errors safely. To fix calculation mistakes, I carefully traced each step and tested the program with different inputs until the results were correct. For the GUI issues, I wrote separate methods for each button's functionality and tested them individually. I also referred to Java documentation and online tutorials for Swing and file handling, which helped me understand and implement the features correctly

9.References

Balsamiq. (n.d.). Uses o balsamiq. Retrieved August 30, 2025, from http://www.balsamiq.com/

BlueJ. (n.d.). BuleJ. Retrieved August 30, 2025, from https://en.wikipedia.org/wiki/BlueJ

Draw.io. (n.d.). Draw.Io. Retrieved August 30, 2025, from https://www.diagrams.net/

Java/Oracle. (n.d.). what is java. Retrieved August 3, 2025, from https://www.java.com/

Microsoft. (n.d.). Miscrosoft word. Retrieved August 30, 2025, from https://www.microsoft.com/en-us/microsoft-365

SynergisticIT. (n.d.). why is java programming language so popular? Retrieved August 30, 2025, from https://www.synergisticit.com/why-java-is-popular/ wikipedia. (n.d.). Balsamiq uses. Retrieved August 30, 2025, from https://en.wikipedia.org

10.Appendix

ArtGalleryGUI

```
package Coursework; import javax.swing.*;
import javax.swing.border.*; import
javax.swing.table.DefaultTableModel; import
java.awt.*; import java.util.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.io.PrintWriter; import
java.io.BufferedWriter; import
java.io.FileWriter; import java.io.*;


public class ArtGalleryGUI extends JFrame {

// for Visitor storage

private ArrayList<ArtGalleryVisitor> visitorList = new ArrayList<>();


// Text fields

private JTextField visitorIdText, fullNameText, contactText, ticketPriceText;     private
JTextField artworkNameText, artworkPriceText, cancelReasonText;


    // Radio buttons for gender

    private JRadioButton maleRadio, femaleRadio, otherRadio;
private ButtonGroup genderGroup;


    // Combo boxes

    private JComboBox<String> dayCombo, monthCombo, yearCombo;
private JComboBox<String> ticketPriceCombo;
    //log visit

    private boolean isActive = false;//it becomes true when visit is logged


    // Buttons
```

```
    private JButton btnAddVisitor, btnLogVisit, btnBuy, btnCalculateDiscount,
btnCalculateRewardsPoint, btnGenerateBill, btnDisplayDetails,
    btnClearFields, btnCancelProduct, btnCheckUpgrade, btnAssignAdvisor,btnRead,btnSave;
private DefaultTableModel tableModel;


    // Table or TextArea to display visitor details or bill
private JTextArea displayArea;     public
ArtGalleryGUI() {
        setTitle("Visitor Management System");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new BorderLayout());         setSize(1200, 800);
setLocationRelativeTo(null);         setResizable(true);


        // Visitor Details Section
visitorIdText = new JTextField(15);
fullNameText = new JTextField(15);
contactText= new JTextField(15);
ticketPriceText = new JTextField(15);


        ticketPriceCombo = new JComboBox<>(new String[]{"Standard", "Elite"});
JTextField registrationDateField = new JTextField(15);


        maleRadio = new JRadioButton("Male");
femaleRadio = new JRadioButton("Female");
otherRadio = new JRadioButton("Other"); genderGroup =
new ButtonGroup();         genderGroup.add(maleRadio);
genderGroup.add(femaleRadio);
genderGroup.add(otherRadio);


        JPanel mainPanel = new JPanel(new BorderLayout(10,10));
mainPanel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));


        // Visitor Panel
```

```
        JPanel visitorPanel = new JPanel(new BorderLayout());
visitorPanel.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createLineBorder(Color.BLACK), "Visitor Details Section",
                TitledBorder.LEFT,
TitledBorder.TOP,
                new Font("SansSerif", Font.BOLD, 14),
new Color(0, 0, 0)
            ));


        JPanel formPanel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5,5,5,5);         gbc.anchor =
GridBagConstraints.WEST;


        gbc.gridx=0;
gbc.gridy=0;
        formPanel.add(new JLabel("Visitor ID:"), gbc);
gbc.gridx=1;
        formPanel.add(visitorIdText, gbc);


        gbc.gridx=2; formPanel.add(new JLabel("Full
        Name:"), gbc);         gbc.gridx=3;
        formPanel.add(fullNameText, gbc);


        gbc.gridx=0; gbc.gridy=1;
        formPanel.add(new JLabel("Gender:"), gbc);
        JPanel genderPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 5,0));
genderPanel.add(maleRadio);         genderPanel.add(femaleRadio);
genderPanel.add(otherRadio);
        gbc.gridx=1; formPanel.add(genderPanel, gbc);


        gbc.gridx=2; formPanel.add(new JLabel("Registration Date:"), gbc);
gbc.gridx=3;         gbc.gridy=1;
```

```
        // Create combo boxes for date
String[] days = new String[31];         for
(int i = 1; i <= 31; i++)             days[i-1]
= String.valueOf(i);
        dayCombo = new JComboBox<>(days);


        String[] months = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
monthCombo = new JComboBox<>(months);


        String[] years = new String[50];
for (int i = 0; i < 50; i++)
            years[i] = String.valueOf(1980 + i);
yearCombo = new JComboBox<>(years);


        // Add day, month, year to a small panel

        JPanel datePanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 5,0));
datePanel.add(dayCombo); datePanel.add(monthCombo);

        datePanel.add(yearCombo);
formPanel.add(datePanel, gbc);


        gbc.gridx=0; gbc.gridy=2;

        formPanel.add(new JLabel("Contact Number:"), gbc);
gbc.gridx=1; formPanel.add(contactText, gbc);         gbc.gridx=2;
formPanel.add(new JLabel("Ticket Type:"), gbc);
gbc.gridx=3;
        formPanel.add(ticketPriceCombo,gbc);



        gbc.gridx=0; gbc.gridy=3;

        formPanel.add(new JLabel("Ticket Price:"), gbc);
gbc.gridx=1;
        ticketPriceText.setEditable(false); // make it non-editable
formPanel.add(ticketPriceText, gbc);
        // Add ActionListener to update ticket price automatically
ticketPriceCombo.addActionListener(new ActionListener()
```

```
        {
            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedType = (String) ticketPriceCombo.getSelectedItem();
if (selectedType.equals("Standard")) {
ticketPriceText.setText("1000");                } else if
(selectedType.equals("Elite")) {                    ticketPriceText.setText("2000");
                } else {
                    ticketPriceText.setText("");
                }
            }
        });


    visitorPanel.add(formPanel, BorderLayout.CENTER);


    JPanel visitorButtons = new JPanel(new FlowLayout(FlowLayout.CENTER,10,10));
    btnAddVisitor = new JButton("Add Visitor");
    btnAddVisitor.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
addVisitor();
            }
        });
    visitorButtons.add(btnAddVisitor);


    btnDisplayDetails = new JButton("Display Details");
btnDisplayDetails.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
displayDetails();
```

```
            }
        });
        visitorButtons.add(btnDisplayDetails);

        btnRead = new JButton("Read");
btnRead.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
read();
            }
        });
        visitorButtons.add(btnRead);

        btnSave= new JButton("Save");
        btnSave.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
                save();
            }
        });
        visitorButtons.add(btnSave);

        btnClearFields = new JButton("Clear Text Fields");
btnClearFields.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
clearFields();
            }
        });
```

```
        visitorButtons.add(btnClearFields);


        Dimension btnSize = new Dimension(150,30);
btnAddVisitor.setPreferredSize(btnSize);
btnDisplayDetails.setPreferredSize(btnSize);
btnRead.setPreferredSize(btnSize);        btnSave.setPreferredSize(btnSize);
btnClearFields.setPreferredSize(btnSize);


        visitorButtons.add(btnAddVisitor);
visitorButtons.add(btnDisplayDetails);
visitorButtons.add(btnRead);        visitorButtons.add(btnSave);
visitorButtons.add(btnClearFields);
visitorPanel.add(visitorButtons, BorderLayout.SOUTH);
mainPanel.add(visitorPanel, BorderLayout.NORTH);


        // ------------------- Purchase Section -------------------

        JPanel purchasePanel = new JPanel(new BorderLayout());
purchasePanel.setBorder(BorderFactory.createTitledBorder("Purchase Section"));


        JPanel purchaseForm = new JPanel(new GridBagLayout());

        GridBagConstraints Gbc = new GridBagConstraints();

        Gbc.insets = new Insets(5,5,5,5);

        Gbc.anchor = GridBagConstraints.WEST;


        JTextField purchaseVisitorIdField = new JTextField(15);
artworkNameText = new JTextField(15);          artworkPriceText
= new JTextField(15);          cancelReasonText = new
JTextField(15);


        Gbc.gridx=0;
Gbc.gridy=0;
        purchaseForm.add(new JLabel("Visitor ID:"), Gbc);
Gbc.gridx=1;
```

```
        purchaseForm.add(purchaseVisitorIdField, Gbc);


        Gbc.gridx=0;
Gbc.gridy=1;
        purchaseForm.add(new JLabel("Artwork Name:"), Gbc);
Gbc.gridx=1;
        purchaseForm.add(artworkNameText, Gbc);


        Gbc.gridx=0;
Gbc.gridy=2;
        purchaseForm.add(new JLabel("Artwork Price:"), Gbc);
Gbc.gridx=1;
        purchaseForm.add(artworkPriceText, Gbc);


        Gbc.gridx=0;
Gbc.gridy=3;
purchaseForm.add(new
JLabel("Cancel Reason:"),
Gbc);
        Gbc.gridx=1;
        purchaseForm.add(cancelReasonText, Gbc);


        purchasePanel.add(purchaseForm, BorderLayout.CENTER);


        JPanel purchaseButtons = new JPanel(new GridLayout(3,2,5,5));
btnBuy = new JButton("Buy");         btnBuy.addActionListener(new
ActionListener()
            {
                public void actionPerformed(ActionEvent e){
buy();
                }
            });
```

```
        purchaseButtons.add(btnBuy);


        btnCalculateDiscount=new JButton("Calculate Discount");
btnCalculateDiscount.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e){
calculateDiscount();
                }
            });
        purchaseButtons.add(btnCalculateDiscount);


        btnCalculateRewardsPoint=new JButton("Reward Points");
btnCalculateRewardsPoint.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e){
calculateRewardPoints();
                }
            });
        purchaseButtons.add(btnCalculateRewardsPoint);


        btnGenerateBill=new JButton("Generate Bill");
        btnGenerateBill.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e){
generateBill();
                }
            });
        purchaseButtons.add(btnGenerateBill);
```

```
        btnCancelProduct=new JButton("Cancel Product");
btnCancelProduct.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e){
cancelProduct();
            }
        });
        purchaseButtons.add(btnCancelProduct);
btnCheckUpgrade = new JButton("Check Upgrade Discounts");
btnCheckUpgrade.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e){
checkUpgrade();
            }
        });
        purchaseButtons.add(btnCheckUpgrade);
purchasePanel.add(purchaseButtons, BorderLayout.SOUTH);


        mainPanel.add(purchasePanel, BorderLayout.WEST);


        // Manage Visitor Log
        JPanel managePanel = new JPanel();        managePanel.setLayout(new
BoxLayout(managePanel, BoxLayout.Y_AXIS));
managePanel.setBorder(BorderFactory.createTitledBorder("Manage Visit"));
        // Create buttons
        Dimension manageBtnSize = new Dimension(200, 40);
btnLogVisit = new JButton("Log Visit");
btnLogVisit.setPreferredSize(manageBtnSize);
btnLogVisit.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent f){
logVisitor();
            }
```

```
        });

    managePanel.add(btnLogVisit);


    btnAssignAdvisor = new JButton("Assign Personal Advisory");
btnAssignAdvisor.setPreferredSize(manageBtnSize);
btnAssignAdvisor.addActionListener(new ActionListener()
        {

            public void actionPerformed(ActionEvent e){
assignPersonalArtAdvisor();
            }

        });

    managePanel.add(btnAssignAdvisor);



    // Set same preferred size for all buttons
//Dimension btnSize = new Dimension(200, 40);
btnLogVisit.setPreferredSize(btnSize);
btnAssignAdvisor.setPreferredSize(btnSize);
btnCalculateDiscount.setPreferredSize(btnSize);        //
Add buttons directly managePanel.add(btnLogVisit);
    managePanel.add(btnAssignAdvisor);

    // managePanel.add(btnCalculateDiscount);


    // Add this panel to the main panel (right side)
mainPanel.add(managePanel, BorderLayout.EAST);
    // ------------------- Display Section -------------------

    String[] columns = {"Visitor ID", "Full Name", "Gender", "Contact","Date","Ticket Type",


            "Ticket Price","Artwork Name","Artwork Price","Cancel Reason"};
tableModel = new DefaultTableModel(columns,0);        JTable
displayTable = new JTable(tableModel);
displayTable.setRowHeight(25);
```

```
        JScrollPane scrollPane = new JScrollPane(displayTable);
scrollPane.setPreferredSize(new Dimension(800,200));


        JPanel displayPanel = new JPanel(new BorderLayout());

        displayPanel.setBorder(BorderFactory.createTitledBorder("Display Section"));
displayPanel.add(scrollPane, BorderLayout.CENTER);


        mainPanel.add(displayPanel, BorderLayout.CENTER);



        add(mainPanel);
setVisible(true);
    }


    public static void main(String[] args) {
new ArtGalleryGUI();
    }


    private void addVisitor() {

        try {

            // --- VISITOR ID ---

            String visitorIdStr = visitorIdText.getText().trim();
if (visitorIdStr.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Visitor ID cannot be empty.");
return;
            }

            int visitorId;
try {
                visitorId = Integer.parseInt(visitorIdStr);
if (visitorId < 0) {
                    JOptionPane.showMessageDialog(this, "Visitor ID cannot be negative.");
return;
                }

            } catch (NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(this, "Visitor ID must be a valid number.");
return;
        }


        String fullName = fullNameText.getText().trim();
if (fullName.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Full Name cannot be empty.");
return;
        }

        if (!fullName.matches("[a-zA-Z\\s]+")) {  // only letters and spaces allowed

            JOptionPane.showMessageDialog(this, "Full Name cannot contain numbers or symbols.");
return;
        }


        String contactNumber = contactText.getText().trim();
if (contactNumber.isEmpty()) {
JOptionPane.showMessageDialog(this, "Contact Number
cannot be empty.");

            return;

        }

        if (contactNumber.length() > 10) {

            JOptionPane.showMessageDialog(this, "Contact Number cannot exceed 10 digits.");
return;
        }
try {
            long num = Long.parseLong(contactNumber);
if (num < 0) {
                JOptionPane.showMessageDialog(this, "Contact Number cannot be negative.");
return;
            }

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(this, "Contact Number must be numeric only.");
return;
```

```
        }


        // Date part unchanged
        String day = (String) dayCombo.getSelectedItem();
        String month = (String) monthCombo.getSelectedItem();
        String year = (String) yearCombo.getSelectedItem();
        String registrationDate = day + "/" + month + "/" + year;


        String gender = "";             if
(maleRadio.isSelected()) {
gender = "Male";
        } else if (femaleRadio.isSelected()) {
gender = "Female";
        } else if (otherRadio.isSelected()) {
gender = "Other";
        } else {
JOptionPane.showMessageDialog(this, "Please select a gender.");
            return;
        }


        String ticketType = (String) ticketPriceCombo.getSelectedItem();
String ticketCostStr = ticketPriceText.getText().trim();             if
(ticketCostStr.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Ticket Cost cannot be empty.");
return;
        }
        double ticketCost;
try {
            ticketCost = Double.parseDouble(ticketCostStr);
        } catch(NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(this,"Ticket Cost must be a valid number.");
return;
        }


        boolean isExist = false;            for
(ArtGalleryVisitor v : visitorList) {
if (v.getVisitorId() == visitorId) {
                JOptionPane.showMessageDialog(this, "Duplicate Visitor ID");
isExist = true;                    break;
            }

        }


        if (!isExist) {

            ArtGalleryVisitor visitor;

            if (ticketType.equalsIgnoreCase("Standard")) {

                visitor = new StandardVisitor(visitorId, fullName, gender, contactNumber,
registrationDate, ticketCost, ticketType);

            } else {

                visitor = new EliteVisitor(visitorId, fullName, gender, contactNumber,
registrationDate, ticketCost, ticketType);
            }


            String artworkName = artworkNameText.getText().trim();
visitor.setArtworkName(artworkName);


            String artworkPriceStr = artworkPriceText.getText().trim();
double artworkPrice = 0;                if (!artworkPriceStr.isEmpty()) {
try {
                    artworkPrice = Double.parseDouble(artworkPriceStr);

                } catch (NumberFormatException e) {

                    JOptionPane.showMessageDialog(this, "Artwork Price must be a valid number.");
return;
```

```
                }
            }
            visitor.setArtworkPrice(artworkPrice);

            visitor.setCancellationReason(cancelReasonText.getText().trim());

            visitorList.add(visitor);
            JOptionPane.showMessageDialog(this, "Visitor added successfully!");
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
    }
}

private void logVisitor(){
    try {
        int visitorId = Integer.parseInt(visitorIdText.getText());
boolean found = false;

        for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
visitor.logVisit();  // increment visit count
isActive=true;
                JOptionPane.showMessageDialog(null, "Visit logged successfully!");
found = true;                    break;
            }
        }
```

```
            if (!found) {
                JOptionPane.showMessageDialog(null, "Visitor ID not found.");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter a valid Visitor ID.");
        }
    }

    private void displayDetails(){
try{

            tableModel.setRowCount(0);
            // Display ALL current visitors (including new ones)
for (ArtGalleryVisitor visitor : visitorList) {
if(visitor!=null){
                    tableModel.addRow(new Object[] {
visitor.getVisitorId(),                         visitor.getFullName(),
visitor.getGender(),
visitor.getContactNumber(),
visitor.getRegistrationDate(),
visitor.getTicketType(),
visitor.getTicketCost(),
visitor.getArtworkName(),
visitor.getArtworkPrice(),
visitor.getCancellationReason(),
                        });
                }

            }

        }catch(Exception none){
```

```
            JOptionPane.showMessageDialog(this,"There is no data to show");


        }
    }


    private void clearFields(){
visitorIdText.setText("");
fullNameText.setText("");
contactText.setText("");
genderGroup.clearSelection();
ticketPriceCombo.setSelectedIndex(0);
ticketPriceText.setText("");
artworkNameText.setText("");
artworkPriceText.setText("");
cancelReasonText.setText("");
dayCombo.setSelectedIndex(0);
monthCombo.setSelectedIndex(0);
yearCombo.setSelectedIndex(0);
visitorIdText.setText("");
    }


    private void buy() {
        try {
            // --- Get Visitor ID ---
            String visitorIdStr = visitorIdText.getText().trim();
if (visitorIdStr.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Visitor ID cannot be empty.");
return;
            }
            int visitorId = Integer.parseInt(visitorIdStr);
if (visitorId < 0) {
                JOptionPane.showMessageDialog(this, "Visitor ID cannot be negative.");
return;
```

```
        }


        // --- Check if visitor is active ---
if (!isActive) {
            JOptionPane.showMessageDialog(this, "Please log in first to make a purchase.");
return;
        }


        // --- Get Artwork Details ---

        String artworkName = artworkNameText.getText().trim();
if (artworkName.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Artwork name cannot be empty");
return;
        }


        String artworkPriceStr = artworkPriceText.getText().trim();
if (artworkPriceStr.isEmpty()) {
JOptionPane.showMessageDialog(this, "Artwork price cannot be
empty.");              return;
        }

        double artworkPrice = Double.parseDouble(artworkPriceStr);


        // --- Find Visitor and make purchase ---
boolean found = false;
        for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
found=true;
                // --- Check if visitor has logged visit ---
if (!visitor.hasLoggedVisit()){ // using your visit flag
            JOptionPane.showMessageDialog(this, "You must log your visit before making a purchase!");
return; // stop purchase if not visited
        }
```

```
                visitor.buyProduct(artworkName, artworkPrice);  // handles isBought & buyCount
visitor.setBought(true);
                JOptionPane.showMessageDialog(this,
                    "Purchase Successful!\n" +
                    "To see the final price, calculate discount using the discount button.",
                    "Purchase Confirmation",
                    JOptionPane.INFORMATION_MESSAGE);
found = true;                    break;
                }
            }


            if (!found) {
                JOptionPane.showMessageDialog(this, "Visitor ID not found in records!");
            }
            displayDetails();//it refreshes table after purchase
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid number format in Visitor ID or Artwork Price.");
} catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }


    private void assignPersonalArtAdvisor() {
try {
            // --- VISITOR ID INPUT ---
            String visitorIdStr = visitorIdText.getText().trim();
if (visitorIdStr.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Visitor ID cannot be empty.");
return;
            }
```

```
            int visitorId = Integer.parseInt(visitorIdStr);


            // --- SEARCH IN VISITOR LIST ---
boolean found = false;
            for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
found = true;


                    // --- CHECK IF ELITE VISITOR ---
if (visitor instanceof EliteVisitor) {
                        EliteVisitor eliteVisitor = (EliteVisitor) visitor; // casting
eliteVisitor.assignPersonalArtAdvisor(); // method call
                        JOptionPane.showMessageDialog(this, "Personal Art Advisor assigned to Elite Visitor: "

                            + eliteVisitor.getFullName());

                    } else {

                        JOptionPane.showMessageDialog(this, "This visitor is not an Elite Visitor. Advisor cannot be
assigned.");

                    }
break;
                }

            }


            if (!found) {

                JOptionPane.showMessageDialog(this, "Visitor ID not found in the system.");

            }


        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(this, "Visitor ID must be a number.");

        } catch (Exception e) {
```

```
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void checkUpgrade() {
        try {
            int visitorId = Integer.parseInt(visitorIdText.getText().trim());
boolean found = false;

            for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
found = true;

                    // Check if a purchase has been made
if (!visitor.isBought()) {
                        JOptionPane.showMessageDialog(null,
                            "Purchase required first to check for upgrade.");
break;
                    }

                    // Only StandardVisitors can upgrade
                    if (visitor instanceof StandardVisitor standardVisitor) {
boolean upgraded = standardVisitor.checkDiscountUpgrade();
if (upgraded) {
                        JOptionPane.showMessageDialog(null,
                            "Congratulations! You are eligible for upgraded discount (15%).");
                    } else {
                        JOptionPane.showMessageDialog(null,
                            "No upgrade. Your discount remains 10%.");
                    }
```

```
                } else {
                    JOptionPane.showMessageDialog(null,
                        "Only Standard Visitors can check for discount upgrade.");
                }
break;
            }
        }

        if (!found) {
            JOptionPane.showMessageDialog(null, "Visitor ID not found.");
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Please enter a valid Visitor ID.");
    }
}

private void calculateDiscount() {
    try {
        int visitorId = Integer.parseInt(visitorIdText.getText().trim());
boolean found = false;

        for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
found = true;

                if (!visitor.isBought()) {
                    JOptionPane.showMessageDialog(null,
                        "Purchase required first to calculate discount.");
break;
```

```
                }


                double discount = visitor.calculateDiscount();
                double finalPrice = visitor.getFinalPrice(); // getter in parent class


                JOptionPane.showMessageDialog(null,
                    "Discount Amount: " + discount + "\nFinal Price: " + finalPrice);
break;
            }
        }
        if (!found) {
            JOptionPane.showMessageDialog(null, "Visitor ID not found.");
        }


    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Invalid Visitor ID.");
    }
  }


  private void calculateRewardPoints() {
try {
        String visitorIdStr = visitorIdText.getText().trim();
        if (visitorIdStr.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Visitor ID cannot be empty.");
return;
        }


        int visitorId = Integer.parseInt(visitorIdStr);
```

```
        // Search for visitor in visitorList
for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
                double points = visitor.calculateRewardPoint(); // Polymorphism here when ticket type separates
both child classes overrides calculate point ()

                JOptionPane.showMessageDialog(this,

                   "Reward Points for Visitor " + visitorId + ": " + points);
return;
              }

           }


           // If not found

           JOptionPane.showMessageDialog(this, "Visitor ID not found!");


       } catch (NumberFormatException ex) {

           JOptionPane.showMessageDialog(this, "Invalid Visitor ID. Please enter a number.");

       }

   }


   private void cancelProduct() {

       try {

           // --- Read input values ---

           String visitorIdStr = visitorIdText.getText().trim();

           String artworkName = artworkNameText.getText().trim();

           String cancellationReason = cancelReasonText.getText().trim();


           if (visitorIdStr.isEmpty() || artworkName.isEmpty() || cancellationReason.isEmpty()) {
JOptionPane.showMessageDialog(this, "All fields must be filled.");                return;
           }
```

```
            int visitorId = Integer.parseInt(visitorIdStr);


            // --- Search visitor in visitorList ---
boolean found = false;
            for (ArtGalleryVisitor visitor : visitorList) {
if (visitor.getVisitorId() == visitorId) {
found = true;


                    // --- Call cancelProduct depending on visitor type ---
String message;
                    if (visitor instanceof EliteVisitor) {
                        message = ((EliteVisitor) visitor).cancelProduct(artworkName, cancellationReason);
                    } else if (visitor instanceof StandardVisitor) {
                        message = ((StandardVisitor) visitor).cancelProduct(artworkName, cancellationReason);
                    } else {
                        message = "Cancel Product.";
                    }


                    // Show result to user
                    JOptionPane.showMessageDialog(this, message);
break;
                }
            }


            if (!found) {
                JOptionPane.showMessageDialog(this, "Visitor ID not found.");
            }


        } catch (NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(this, "Visitor ID must be a number.");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    // Method to handle bill generation
private void generateBill() {
        String visitorIdStr = visitorIdText.getText().trim();
if (visitorIdStr.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Visitor ID cannot be empty.");
return;
        }
        int visitorId;
try {
            visitorId = Integer.parseInt(visitorIdStr);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Visitor ID must be a number.");
return;
        }

        // Search for the visitor
ArtGalleryVisitor visitor = null;         for
(ArtGalleryVisitor v : visitorList) {
if (v.getVisitorId() == visitorId) {
visitor = v;                 break;
            }
        }
        if (visitor == null) {
            JOptionPane.showMessageDialog(this, "Visitor ID not found.");
return;
        }
```

```
        if (!visitor.isBought()) {
            JOptionPane.showMessageDialog(this, "No purchase made by this visitor.");
return;
        }

        // Build bill string

        String bill =
            "===== ART GALLERY BILL =====\n" +
            "Visitor ID: " + visitor.getVisitorId() + "\n" +
            "Visitor Name: " + visitor.getFullName() + "\n" +
            "Artwork Name: " + visitor.getArtworkName() + "\n" +
            "Artwork Price: " + visitor.getArtworkPrice() + "\n" +
            "Discount Amount: " + visitor.calculateDiscount() + "\n" +
            "Final Price: " + visitor.getFinalPrice() + "\n" +
            "============================\n";

        // Display in GUI
        JTextArea billArea = new JTextArea(bill.toString());
billArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(billArea);
scrollPane.setPreferredSize(new Dimension(400, 200));
        JOptionPane.showMessageDialog(this, scrollPane, "Bill Details",
JOptionPane.INFORMATION_MESSAGE);

        // Export to .txt file
        try (PrintWriter out = new PrintWriter(visitor.getFullName() + "_Bill.txt")) {
out.println(bill.toString());
            JOptionPane.showMessageDialog(this, "Bill exported as " + visitor.getFullName() + "_Bill.txt");
} catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, "Error exporting bill: " + ex.getMessage());
        }
    }



private void save() {     if
(visitorList.isEmpty()) {
        JOptionPane.showMessageDialog(this, "No visitor data to save!");
return;
    }

    // Use absolute path
    File file = new File("C:\\Users\\eijke\\First Semester\\CourseWork\\Coursework\\VisitorDetails.txt");

    try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
        // Header
        bw.write(String.format("%-5s %-20s %-10s %-15s %-15s %-10s %-10s%n",
                "ID", "Name", "Gender", "Contact No", "Registration Date", "Ticket Type", "Ticket Cost"));

        // Visitor data
        for (ArtGalleryVisitor visitor : visitorList) {
            bw.write(String.format("%-5d %-20s %-10s %-15s %-15s %-10s %-10.2f%n",
visitor.getVisitorId(),                    visitor.getFullName(),
visitor.getGender(),                    visitor.getContactNumber(),
visitor.getRegistrationDate(),
                    visitor.getTicketType(),
visitor.getTicketCost()));
        }

        JOptionPane.showMessageDialog(this, "Visitor details saved successfully!");
```

```
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error saving file: " + e.getMessage());
    }
}
private void read() {
    File file = new File("C:\\Users\\eijke\\First Semester\\CourseWork\\Coursework\\VisitorDetails.txt");

    // Check if file exists and is not empty
if (!file.exists() || file.length() == 0) {
        JOptionPane.showMessageDialog(this, "No visitor data added yet!");
return;
    }

    // Read and display file content
    try (BufferedReader br = new BufferedReader(new FileReader(file))) {
JTextArea displayArea = new JTextArea();        displayArea.setEditable(false);

        String line;
        while ((line = br.readLine()) != null) {
displayArea.append(line + "\n");
        }

        JScrollPane scrollPane = new JScrollPane(displayArea);
scrollPane.setPreferredSize(new Dimension(600, 300));
JOptionPane.showMessageDialog(this, scrollPane, "Visitor Records",
JOptionPane.INFORMATION_MESSAGE);

        // Print message to terminal
        System.out.println("Visitor data loaded successfully from VisitorDetails.txt");
```

```
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error reading file: " + e.getMessage());
System.out.println("Error reading file: " + e.getMessage());
    }
}
}
```

Elite visitor

```
package Coursework;


public class EliteVisitor extends ArtGalleryVisitor{


    // Unique attributes for EliteVisitor     private
boolean assignedPersonalArtAdvisor;     private
boolean exclusiveEventAccess;


    // Constructor
    public EliteVisitor(int visitorId, String fullName, String gender, String contactNumber,
String registrationDate, double ticketCost, String ticketType) {         super(visitorId,
fullName, gender, contactNumber, registrationDate, ticketCost,          ticketType);
        this.assignedPersonalArtAdvisor = false;
this.exclusiveEventAccess = false;
    }


    // Accessor methods
    public boolean getAssignedPersonalArtAdvisor() {
return assignedPersonalArtAdvisor;
    }
```

```
    public boolean getExclusiveEventAccess() {
return exclusiveEventAccess;
    }


    // Method to assign a personal art advisor
public boolean assignPersonalArtAdvisor() {     if
(rewardPoints > 5000) {
this.assignedPersonalArtAdvisor = true;
    }
    return this.assignedPersonalArtAdvisor;
    }


    // Method to check exclusive event access
public boolean checkExclusiveEventAccess() {     if
(assignedPersonalArtAdvisor) {
this.exclusiveEventAccess = true;
    }
    return this.exclusiveEventAccess;
    }


    // Overriding buyProduct method
    @Override
    public String buyProduct(String artworkName, double artworkPrice) {
if (!isActive) {
            return "Please log in before making a purchase.";
        }
        if (this.artworkName == null || !this.artworkName.equals(artworkName)) {
this.artworkName = artworkName;
            this.artworkPrice = artworkPrice;
this.isBought = true;             this.buyCount++;
            return "Purchase successful for artwork: " + artworkName;
```

```
        }
        return "This artwork has already been purchased.";
    }

    // Method to calculate discount
    @Override
     public double calculateDiscount() {
    double discountAmount = artworkPrice * 0.40; // Calculate 40% discount     double
finalPrice = artworkPrice - discountAmount; // Calculate final price after discount     return
finalPrice; // Return the discounted price
    }
    // Method to calculate reward points
    @Override
    public double calculateRewardPoint() {
if (isBought) {
        double finalPrice = calculateDiscount();//Get discounted price from calculateDiscount()
rewardPoints += finalPrice * 10;//add reward points(10 points per rupee)
        }
        return rewardPoints;
    }
    // Overriding generateBill method
    @Override
    public void generateBill() {
if (!isBought) {
            System.out.println("No purchase to generate bill.");
        } else {
            double discountAmount = artworkPrice * 0.40;
double finalPrice = artworkPrice - discountAmount;
System.out.println("Visitor ID: " + visitorId);
            System.out.println("Visitor Name: " + fullName);
```

```
            System.out.println("Artwork Name: " + artworkName);
            System.out.println("Artwork Price: " + artworkPrice);
            System.out.println("Discount Amount: " + discountAmount);
            System.out.println("Final Price: " + finalPrice);
        }
    }

    // Method to terminate visitor
private void terminateVisitor() {
isActive = false;
        assignedPersonalArtAdvisor = false;
exclusiveEventAccess = false;
visitCount = 0;         cancelCount = 0;
rewardPoints = 0;
    }

    // Method to cancel product
    @Override
    public String cancelProduct(String artworkName, String cancellationReason) {
    // Check if cancellation limit reached
if (cancelCount >= 3) {
        terminateVisitor();  // Terminate account
        return "Your account has been terminated due to excessive cancellations.";
    }

    // Check if any product has been purchased
if (buyCount == 0) {         return "No product to
cancel.";
    }
```

```
    // Check if artwork name matches
    if (this.artworkName != null && this.artworkName.equals(artworkName)) {
        // Calculate refundable amount after 5% cancellation fee
double refundableAmount = artworkPrice - (artworkPrice * 0.05);

        // Deduct reward points based on discounted final price         double
finalPrice = calculateDiscount(); // reuse discount calculation
rewardPoints -= finalPrice * 10;

        // Reset purchase info
this.artworkName = null;         isBought
= false;

        // Update cancellation and purchase counters
cancelCount++;         buyCount--;
        return "Cancellation successful. Refund amount: " + refundableAmount +
". Reason: " + cancellationReason;
    }

    // If artwork names don't match
return "Incorrect artwork name.";
    }
    // Overriding display method
    @Override     public
void display() {
super.display();
        System.out.println("Assigned Personal Art Advisor: " + assignedPersonalArtAdvisor);
System.out.println("Exclusive Event Access: " + exclusiveEventAccess);
    }
}
```

ArtGalleryVisitor

```
package Coursework; abstract
class ArtGalleryVisitor{     //
Protected attributes
    protected int visitorId, visitCount, cancelCount, buyCount;

    protected String fullName, gender, contactNumber, registrationDate, ticketType;
protected double ticketCost, rewardPoints, finalPrice, discountAmount, artworkPrice,
refundableAmount;
    protected String cancellationReason, artworkName;
protected boolean isActive, isBought;     protected
final int  cancelLimit;


    // Constants for ticket types and their costs     private static
final int STANDARD_TICKET_COST = 1000;     private
static final int ELITE_TICKET_COST = 2000;


    // Constructor

    public ArtGalleryVisitor(int visitorId, String fullName,String gender,String contactNumber,String registrationDate,

    double ticketCost,String ticketType) {
this.visitorId = visitorId;         this.fullName =
fullName;         this.contactNumber =
contactNumber;         this.gender = gender;
this.ticketType = ticketType;
this.registrationDate = registrationDate;
this.ticketCost = ticketCost;


        // Initialize default values
this.visitCount = 0;
this.rewardPoints = 0;         this.finalPrice
= 0;         this.discountAmount = 0;
this.refundableAmount = 0;
this.cancelCount = 0;         this.buyCount
= 0;         this.isActive = false;
this.isBought = false;
```

```
this.cancellationReason = "";
this.cancelLimit = 3; // Default value
    }


    // Accessor (getter) methods
public int getVisitorId(){
return visitorId;
    }


    public String getFullName(){
return fullName;
    }


    public String getGender(){
return gender;
    }


    public String getContactNumber(){
return contactNumber;
    }


    public String getRegistrationDate(){
return registrationDate;
    }


    public double getTicketCost(){
        return ticketCost;
    }


    public String getTicketType(){
return ticketType;
```

```
    }


    public boolean isBought(){
return isBought;
    }


    public void setFullName(String fullName){
this.fullName = fullName;
    }


    public void setGender(String gender){
this.gender = gender;
    }


    public void setContactNumber(String contactNumber){
this.contactNumber = contactNumber;
    }


    public void setBought(boolean bought) {
this.isBought = bought;
    }


    // Method to log visits
public void logVisit() {
visitCount++;         isActive
= true;
    }

    //for getter method     public
String getArtworkName() {
return artworkName;
    }
```

```
    public double getArtworkPrice() {
return artworkPrice;
    }


    public String getCancellationReason() {
return cancellationReason;
    }


    public double getFinalPrice() {
return finalPrice;
    }
    //for setter method
    public void setArtworkName(String artworkName) {
this.artworkName = artworkName;
    }


    public void setArtworkPrice(double artworkPrice) {
this.artworkPrice = artworkPrice;
    }


    public void setCancellationReason(String cancellationReason) {
this.cancellationReason = cancellationReason;
    }
    public boolean hasLoggedVisit(){
return visitCount>0;
}

    // Abstract methods     public abstract String buyProduct(String
artworkName, double artworkPrice);


    public abstract double calculateDiscount();
```

```
    public abstract  double calculateRewardPoint();

    public abstract String cancelProduct(String artworkName, String cancellationReason);

    public abstract void generateBill();

    // Method to display visitor details
public void display() {
        System.out.println("Visitor ID: " + visitorId);
        System.out.println("Full Name: " + fullName);
        System.out.println("Gender: " + gender);
        System.out.println("Contact Number: " + contactNumber);
        System.out.println("Registration Date: " + registrationDate);
        System.out.println("Ticket Cost: " + ticketCost);
        System.out.println("Ticket Type: " + ticketType);
        System.out.println("Visit Count: " + visitCount);
        System.out.println("Reward Points: " + rewardPoints);
        System.out.println("Cancel Count: " + cancelCount);
        System.out.println("Buy Count: " + buyCount);
        System.out.println("Is Active: " + isActive);
        System.out.println("Is Bought: " + isBought);
        System.out.println("Final Price: " + finalPrice);
        System.out.println("Discount Amount: " + discountAmount);
        System.out.println("Artwork Name: " + artworkName);
        System.out.println("Artwork Price: " + artworkPrice);
        System.out.println("Refundable Amount: " + refundableAmount);
        System.out.println("Cancellation Reason: " + cancellationReason);
```

```
    }
}
```

Standard visitor

```
package Coursework;


public class StandardVisitor extends ArtGalleryVisitor {

    // Additional attributes     private
boolean upgradeDiscount;     private
final int visitLimit;     private float
discountPercent;


    // Constructor

    public StandardVisitor(int visitorId, String fullName, String gender, String contactNumber,
String registrationDate, double ticketCost, String ticketType) {         super(visitorId, fullName,
gender, contactNumber, registrationDate, ticketCost,         ticketType);
        this.visitLimit = 5; // Set default visit limit

        this.discountPercent = 0.10f; // Default discount percent
this.upgradeDiscount= false; // Default eligibility
    }


    // Accessor methods

    public boolean getUpgradeDiscount() {
return upgradeDiscount;
    }


    public int getVisitLimit() {
return visitLimit;
    }


    public float getDiscountPercent() {

        return discountPercent;
```

```
    }

    // buyProduct method
@Override
    public String buyProduct(String artworkName, double artworkPrice) {
if (!isActive) {
        return "Please log in before making a purchase.";

    }

    if (this.artworkName != null && this.artworkName.equals(artworkName)) {
return "You have already purchased the artwork '" + artworkName + "'.";
    }

    this.artworkName = artworkName;
this.artworkPrice = artworkPrice;
this.isBought = true;     this.buyCount++;
    return "Artwork '" + artworkName + "' purchased successfully" + artworkPrice + ".";

    }

       public boolean checkDiscountUpgrade() {
if (visitCount >= visitLimit) {
upgradeDiscount = true;
discountPercent = 0.15f;
        } else {

            upgradeDiscount = false;
discountPercent = 0.10f;
        }

        return upgradeDiscount;

    }



    @Override

    public double calculateDiscount() {
if (!isBought) {
discountAmount = 0;
finalPrice=artworkPrice;
return discountAmount;
        }
```

```
        checkDiscountUpgrade();  // Call the method here

        discountAmount = artworkPrice * discountPercent;
finalPrice = artworkPrice - discountAmount;

        return discountAmount;
    }

    @Override
    public double calculateRewardPoint() {
if (!isBought) {
        // No product bought, so no new reward points earned
return rewardPoints;
    }

    // Calculate reward points: 5 points for each rupee spent on final price
rewardPoints += finalPrice * 5;

    return rewardPoints;
    }

    @Override     public  void
generateBill() {         if
(!isBought) {
            System.out.println("No purchase made to generate a bill.");
        } else {
            System.out.println("Visitor ID: " + visitorId);
            System.out.println("Visitor Name: " + fullName);
System.out.println("Artwork Name: " + artworkName);
```

```
            System.out.println("Artwork Price: " + artworkPrice);
            System.out.println("Discount Amount: " + calculateDiscount());
            System.out.println("Final Price: " + finalPrice);
        }
    }

    // terminateVisitor method
private void terminateVisitor() {
isActive = false;
upgradeDiscount = false;
visitCount = 0;         cancelCount
= 0;         rewardPoints = 0;
    }

    // cancelProduct method
    @Override
    public String cancelProduct(String artworkName, String cancellationReason) {
    // Check if cancellation limit is reached
if (cancelCount >= cancelLimit) {
terminateVisitor();
        return "Your account has been terminated due to excessive cancellations.";
    }

    // Check if any product has been purchased
if (!isBought) {
        return "No product to cancel.";
    }

    // Check if the given artwork name matches the purchased artwork     if
(this.artworkName != null && this.artworkName.equals(artworkName)) {
```

```
// Reset purchase details         this.artworkName = null;         this.isBought =
false;

        // Calculate refundable amount with a 10% cancellation fee
refundableAmount = artworkPrice - (artworkPrice * 0.10);

        // Deduct reward points earned for this purchase
rewardPoints -= finalPrice * 5;

        // Update cancellation count and buy count
cancelCount++;         buyCount--;

        // Record the cancellation reason
this.cancellationReason = cancellationReason;

        // Return success message
        return "Cancellation successful. Refundable Amount: " + refundableAmount;
    }

    // If artwork names do not match
return "Incorrect artwork name.";
    }
    // Display details method
    @Override     public
void display() {
        super.display(); // Call the parent class display method
        System.out.println("Eligible for Discount Upgrade: " + upgradeDiscount);
        System.out.println("Visit Limit: " + visitLimit);
        System.out.println("Discount Percent: " + discountPercent);
    }
```

```
}
```