Java Programming

DATE 20 July – 20 August
COURSE TITLE: Internship

STUDENT'S NAME: ShehrYar Ahmed Khan
Student Id : CA/JU1/34691

INTRODUCTION

As part of my internship with **Code Alpha**, I completed four hands-on Java projects to strengthen my programming skills and apply theoretical concepts practically.

Student Grade Tracker: Built a GUI-based system to input grades, calculate averages, and generate student performance reports using Java Swing and Array Lists.

Stock Trading Platform: Simulated a basic stock market environment with buy/sell functionality and portfolio tracking using OOP.

AI Chatbot: Created a Java chatbot using rule-based logic and simple NLP to respond to common queries interactively.

Hotel Reservation System: Developed a booking system with room categorization, reservation management, and data persistence using file I/O.

These projects enhanced my understanding of real-world application development, GUI design, and core Java principles.

Objectives

Task 1: Student Grade Tracker

Objective: To develop a GUI-based system that manages student grades and generates performance summaries using Java and Array Lists.

Task 2: Stock Trading Platform

Objective: To simulate a basic stock trading environment with buy/sell functionality and portfolio tracking using Object-Oriented Programming.

Task 3: AI Chatbot

Objective: To build an interactive Java-based chatbot using simple NLP and rule-based responses for real-time user communication.

Task 4: Hotel Reservation System

Objective: To create a reservation system that manages hotel room bookings, cancellations, and room categories with file-based data storage.

Task 1 :

Objective:

To develop a Java-based application that allows teachers to input and manage student grades, calculate average, highest, and lowest scores, and display a summary report using a clean GUI interface with ArrayList-based data management.

Tools & Technologies Used:

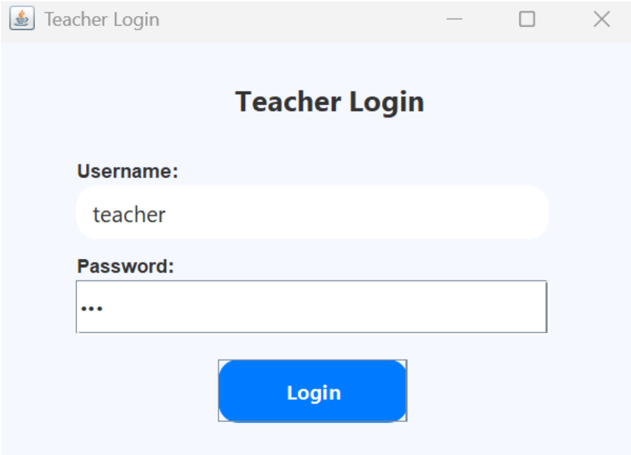
- Object-Oriented Programming (OOP)
- Event-Driven GUI (Swing)
- File Handling with BufferedReader and BufferedWriter
- Data analysis using loops and conditions
- Modular architecture (separate methods for logic)

Features Implemented:

- Teacher login with credential validation
- Grade entry interface (student name, score)
- Automatic calculation of:
  - Average score
  - Highest score
  - Lowest score
- Summary report displayed in a new GUI window
- All activities (login, data entry) logged in activity_log.txt

Conclusion:

This task enhanced my understanding of GUI design in Java, event handling, and file-based data management. It mimicked real-world systems where teachers manage academic records, making it an insightful and practical exercise in software development.

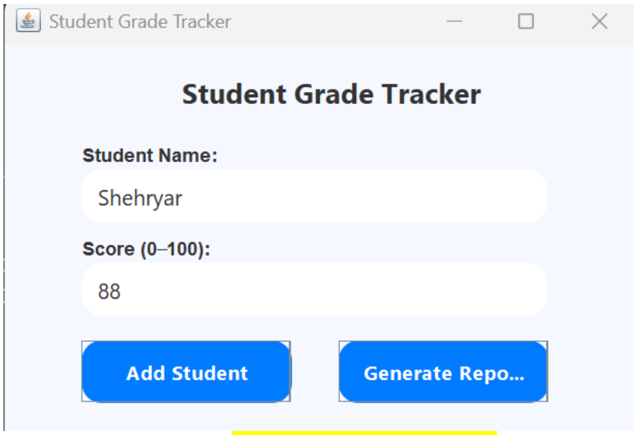


Teacher Login

Username: teacher

Password: •••

Login

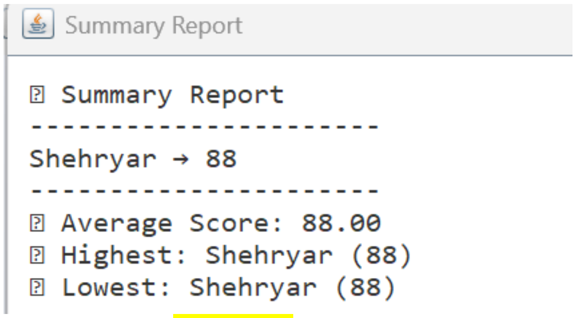


Student Grade Tracker

Student Name: Shehryar

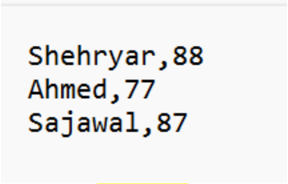
Score (0–100): 88

Submitting Grade



```
⍰ Summary Report
----------------------
Shehryar → 88
----------------------
⍰ Average Score: 88.00
⍰ Highest: Shehryar (88)
⍰ Lowest: Shehryar (88)
```

Summary



```
Shehryar,88
Ahmed,77
Sajawal,87
```

.txt file

Task 2 :

Title: Stock Trading Platform

Objective:

To design a GUI-based Java application that simulates a simple stock trading environment, allowing users to buy, sell, and track shares, manage a portfolio, and simulate market dynamics using Object-Oriented Programming principles.

Tools & Technologies Used:

Language: Java (JDK 17+)
GUI Framework: Java Swing
IDE: VS Code
Concepts Applied:
Object-Oriented Programming
GUI Development (Swing)
Event Handling
Collections (HashMap)
Business Logic & State Management

Features Implemented:

Real-Time Market Simulation:

- Randomly generated stock prices between 6–10 for symbols like AAPL, GOOGLE, TESLA, etc.
- Market refresh functionality to simulate real-world fluctuation.

User Portfolio & Balance Management:

- Initial balance of **$10,000**.
- Displays live portfolio status with owned stocks and quantities.
- Real-time balance updates after every transaction.

Buy Logic:

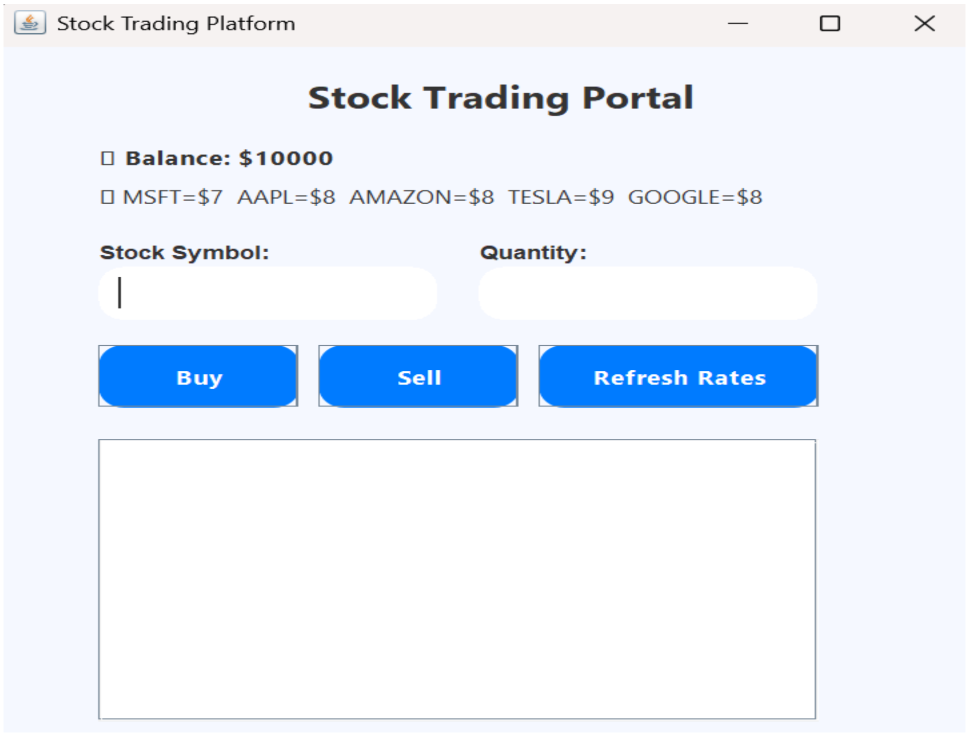
- Users can buy any quantity of available stocks.
- If shares ≤ 30 → **3x profit** is added.
- If shares > 30 → **2x loss** is deducted.

Sell Logic:

- User can sell previously owned shares.
- Stocks are sold at the current market price.
- Balance updates accordingly.

Custom Styled UI:

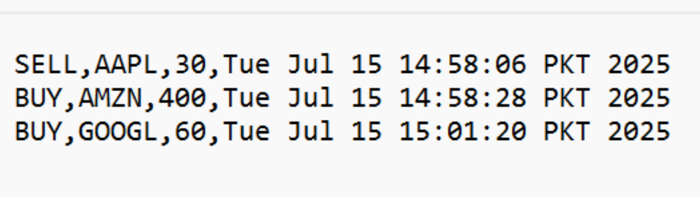
- Rounded buttons & text fields for a clean, modern look.
- Compact, intuitive layout for ease of interaction.



□ **Balance: $10000**

□ MSFT=$7 AAPL=$8 AMAZON=$8 TESLA=$9 GOOGLE=$8

Interface



```
SELL,AAPL,30,Tue Jul 15 14:58:06 PKT 2025
BUY,AMZN,400,Tue Jul 15 14:58:28 PKT 2025
BUY,GOOGL,60,Tue Jul 15 15:01:20 PKT 2025
```

Activity History

Task 3 :

Title: Java-based AI Chatbot with Interactive GUI

Objective:

To develop a rule-based chatbot in Java using Swing, capable of real-time user interaction through a modern GUI and respond to frequently asked questions using keyword detection.

Tools & Technologies Used:

- Java (JDK 8+)
- Java Swing (GUI toolkit)
- OOP (Object-Oriented Programming principles)

Features Implemented:

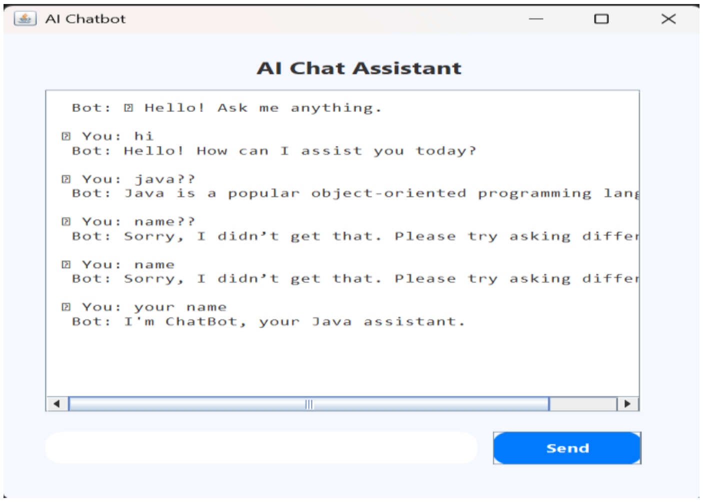
Feature	**Description**
Interactive Chat Interface	GUI with styled input box, send button, and chat history display
Resizable Window	Chat window can be maximized/minimized by the user
Keyword Matching Logic	Chatbot uses basic NLP keyword checking to return responses
Modern Styling	Rounded buttons and text fields for a professional interface
No External Dependencies	Fully functional with pure Java — no libraries or databases required.

Core Functional Flow:

1. **User Input:**
   - User types a message into the input field and clicks the **Send** button.
2. **Keyword-Based Response:**
   - Program checks if the user input contains any predefined keywords (e.g., "hello", "java", "grade").
   - If a match is found, it returns a suitable predefined response.
   - If no match is found, it replies: *"Sorry, I didn't get that."*
3. **Chat Display:**
   - Messages from both user and bot are displayed in the chat area.

Sample Supported Keywords & Responses:

Keyword	**Bot Response**
hello, hi	"Hello! How can I assist you today?"
java	"Java is a popular object-oriented programming language."
your name	"I'm ChatBot, your Java assistant."
grade	"I can help you manage student grades and performance."
bye	"Goodbye! See you soon."
thank	"You're welcome!"



```
 Bot: ⍰ Hello! Ask me anything.

⍰ You: hi
 Bot: Hello! How can I assist you today?

⍰ You: java??
 Bot: Java is a popular object-oriented programming lang

⍰ You: name??
 Bot: Sorry, I didn't get that. Please try asking differ

⍰ You: name
 Bot: Sorry, I didn't get that. Please try asking differ

⍰ You: your name
 Bot: I'm ChatBot, your Java assistant.
```

Interface and Test Output

Task 4 :

Title :

Hotel Reservation System

Objective

The purpose of this task is to build a user-friendly, file-based hotel reservation system using Java and Swing that allows users to book, cancel, and manage hotel room reservations with categorized room types. The system simulates payment calculation and provides a visual overview of all bookings. The project applies core principles of Object-Oriented Programming (OOP) and GUI design.

Key Features & Functional Modules

#	Module/Feature	Description
1	**Room Categorization**	Supports 3 categories: • **Standard** ($100/night) • **Deluxe** ($200/night) • **Suite** ($350/night)
2	**Reservation Interface**	Users enter their **name**, choose **room type**, and input **number of nights**.
3	**Payment Simulation**	Calculates and displays total cost before confirming the booking.
4	**Booking Confirmation**	Upon submission, booking details are stored in `reservations.txt` and a success message is shown.
5	**Cancellation System**	Allows users to cancel their reservation by entering their name; system searches and deletes the matching record from the file.
6	**Booking Viewer**	Displays all stored bookings in a formatted window via a scrollable text area.
7	**Data Persistence**	All reservations are saved in plain text file (`reservations.txt`) using Java File I/O.
8	**Modern Graphical Interface**	UI features rounded buttons, fields, fonts, and a resizable frame for improved UX.

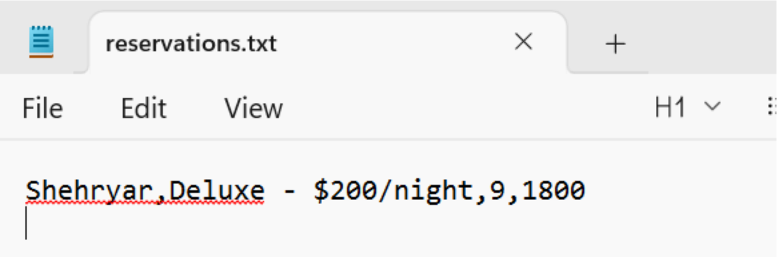
Technologies & Tools Used

- Programming Language: Java
- GUI Framework: Swing (JFrame, JPanel, JLabel, JTextField, JComboBox, JButton)
- File Management: BufferedWriter, BufferedReader, FileWriter, FileReader
- Design Principles: OOP (Encapsulation, Classes, Event Handling)
- UX Styling: Custom paint methods for buttons/text fields for rounded, modern appearance
- Storage File: reservations.txt.

File Format & Structure

All reservations are saved as plain text in a comma-separated format.

File Name: reservations.txt



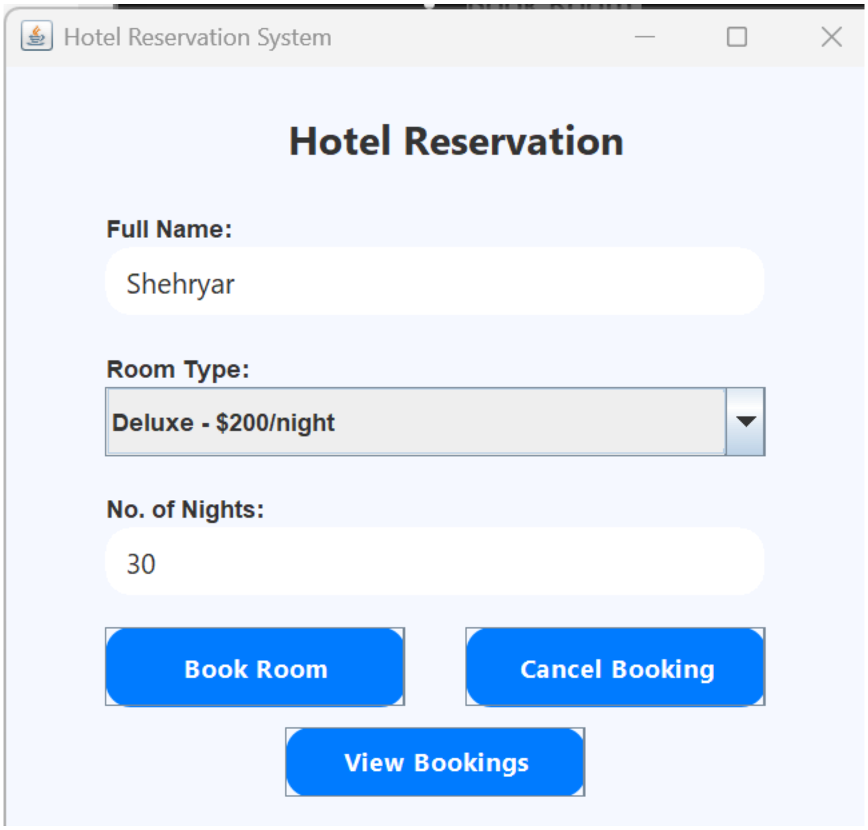
```
Shehryar,Deluxe - $200/night,9,1800
```

User Interface Highlights

- Main window titled **Hotel Reservation System**
- Three primary buttons:
  - **Book Room**
  - **Cancel Reservation**
  - **View All Bookings**
- Input fields for:
  - Full Name
  - Room Category (dropdown menu)
  - Number of Nights

All components are styled consistently with previous tasks using:

- Rounded buttons (JButton)
- Styled text fields (JTextField)
- Scrollable reports in JDialog



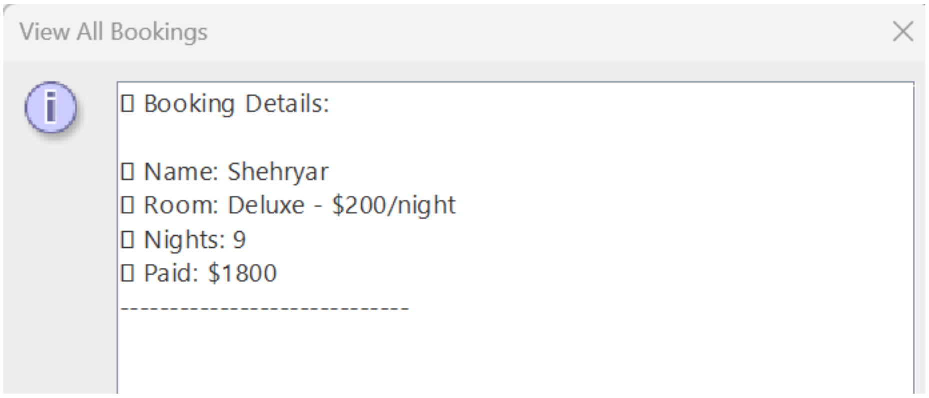
Hotel Reservation

Full Name: Shehryar

Room Type: Deluxe - $200/night

No. of Nights: 30

Reservation System



```
□ Booking Details:

□ Name: Shehryar
□ Room: Deluxe - $200/night
□ Nights: 9
□ Paid: $1800
-----------------------------
```

Booking History