Project Based Evaluation

Project Report

Semester-IV (Batch-2023)

Java Gamble

Supervised By:

Mr. Robin

Submitted By:

Harman Singla,	2310990513 (G7)
Harry Jindal,	2310990514 (G7)
Mahesh Dadwal,	2310990562 (G7)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

The Java Gambling Game – "Double the Money" is a Java-based console application designed to provide an engaging and educational experience that combines betting mechanics with programming knowledge. The game revolves around a multiple-choice quiz system where players answer Java and DSA-related questions. A correct answer doubles the player's bet amount, while an incorrect response results in a loss. This project applies Object-Oriented Programming (OOP) principles to ensure clean structure, reusability, and maintainability.

Each player is identified with attributes such as name, age, gender, and their gameplay involves managing balances through deposit and withdrawal options. Player data is persistently stored using file handling, allowing sessions to be resumed.

The system supports the following core functionalities:

1. **Register New User**
2. **Play Game**
3. **Deposit Money**
4. **Withdraw Money**
5. **View Users**
6. **Sort by Name**
7. **Sort by Balance**
8. **Sort by Age**
9. **Save and Exit**

Technical Highlights:

- **OOP Concepts**: Encapsulation, modular class design
- **User Interaction**: Console-based input/output for intuitive gameplay
- **Custom Sorting**: Users can be sorted by name, balance, or age
- **File Handling**: Save and load user data in a persistent format
- **Educational Gameplay**: MCQs focused on Java and DSA topic

TABLE OF CONTENTS

S NO.	CONTENTS	SIGNATURE
1.)	Introduction	
2.)	Problem Definition & Requirements	
3.)	Proposed Design	
4.)	Performance & Limitations	
5.)	Results	
6.)	References	

1. Introduction

In today’s world, gambling games are an exciting and engaging way for people to test their knowledge while having fun. The **"Double the Money"** game is a Java-based command-line application designed to simulate a betting game with a twist. It allows players to bet their virtual money based on their knowledge of Java and Data Structures & Algorithms (DSA). Players can increase their balance by answering multiple-choice questions correctly, doubling their bet, or losing it on incorrect answers.

This project aims to provide a fun, interactive way to apply core programming concepts while offering a simple gambling experience. The game includes features like user registration, depositing and withdrawing money, and sorting users by various attributes such as name, balance, or age. Players can keep track of their progress and interact with the system using a simple console-based interface.

A key feature of this system is its ability to manage player data, such as **name**, **age**, and **balance**, and store this data persistently in a file. Additionally, users can enjoy sorting functionalities to view users based on various criteria, and the system ensures all player data is saved for later sessions.

This project demonstrates fundamental Java concepts such as **Object-Oriented Programming (OOP)**, **file handling**, and **sorting algorithms**, while also offering an enjoyable and educational gaming experience. It serves as a foundation for future improvements, such as adding a graphical user interface (GUI), implementing advanced game mechanics, or expanding the database system to store larger amounts of player data.

1.1Background

Gambling games have always been a popular form of entertainment, offering excitement and engagement through a combination of chance and knowledge. In the world of programming, implementing such games is an excellent way to apply concepts of object-oriented programming (OOP), file handling, and data management. While many commercial gambling games are available, few offer a simple, educational, and customizable platform for understanding programming fundamentals.

This project was developed using Java, with a focus on core programming concepts such as OOP, file handling, sorting algorithms, and user data management. The game is designed to allow users to bet their virtual money on multiple-choice questions related to Java and Data Structures & Algorithms (DSA). Player data, including balance and user information, is stored in a human-readable .txt file, making it easy to track progress and share data across sessions. The system's command-line interface is user-friendly and serves as a great starting point for further enhancements, such as a graphical user interface (GUI) or a web-based version.

1.2. Objectives

The main objectives of the Java Gambling Game are:

- To implement an interactive and educational betting game using Java.
- To provide a simple interface for registering users, placing bets, and tracking progress.
- To allow users to deposit and withdraw virtual money for betting.
- To sort users based on different criteria such as name, balance, and age.
- To store and retrieve player data in a human-readable .txt file for persistence.
- To use Java data structures (e.g., ArrayLists, Sorting algorithms) to enhance user experience and performance.
- To strengthen understanding of core Java concepts such as OOP, file handling, collections, and sorting

1.3 Significance

The Java Gambling Game serves as a valuable tool for learning Java programming while providing a fun and engaging experience. The significance of this project can be summarized as follows:

- **Educational Value**: The project provides hands-on experience in implementing Java-based gambling mechanics, enhancing understanding of OOP principles, file handling, and data manipulation techniques.
- **Engagement and Motivation**: By integrating multiple-choice questions on Java and DSA topics, the game combines entertainment with learning, encouraging users to deepen their knowledge while having fun.
- **Customizability and Expandability**: The game is built with flexibility in mind. The simple command-line structure makes it easy to expand, whether by adding new betting mechanics, improving user interaction with a GUI, or integrating a database to manage user information.
- **Readable and Persistent Data Storage**: Player data, including balance and personal details, is stored in an easily accessible .txt file. This transparency ensures that players can track their progress across sessions and makes the data portable across different platforms.
- **Promotes Good Practices**: The project encourages the use of good programming practices, such as clean code design, file handling for persistence, and the application of sorting and data structure concepts. These habits are beneficial not only in gaming but also in software development and project management.

2. Problem Defination and Requirements

2.1 Problem Statement

In today's fast-paced world, gambling games that combine entertainment with skill-based elements are in high demand. However, many gambling platforms are complex, often relying on advanced algorithms or external dependencies, making them difficult for beginners to implement or customize. Additionally, many users enjoy betting games but lack an easy-to-understand, interactive platform where they can test their knowledge while engaging in a fun, risk-free environment.

Traditional betting games often focus solely on chance, but incorporating educational elements such as quizzes on programming concepts can add value to the experience. However, many platforms overlook this combination of learning and entertainment. There is a need for a simple, command-line-based gambling game that allows users to bet on multiple-choice questions related to Java programming and Data Structures & Algorithms (DSA), while also managing their virtual balance and personal information.

The Java Gambling Game aims to address this gap by offering an easy-to-use and educational betting game. It allows users to register, deposit, withdraw, and place bets on questions, all while tracking their balance and progress. This project will provide a fun yet educational experience, helping users reinforce programming concepts through a gaming platform. Furthermore, it ensures that all user data, including balance and personal details, is stored in a human-readable format, making it accessible for future sessions and analysis.

2.2 Requirements

The Java Gambling Game is a terminal-based Java application that allows users to register, deposit money, place bets on multiple-choice questions, and track their progress. To build and operate this application effectively, several functional and non-functional requirements must be met.

2.2.1 Software Requirements:

- **Programming Language:** Java (JDK 8 or above)
- **IDE**: IntelliJ IDEA or any Java-supported IDE
- **Operating System**: Windows / Linux / macOS
- **Libraries/Packages**:
  - java.util (for collections like LinkedList, PriorityQueue)
  - java.io (for file handling)
  - java.time (for handling deadlines using LocalDate)
- **Text Editor**: Any (VS Code, etc. — for viewing .txt file output)
- **Compiler**: Java Compiler (javac)

2.2.2 Hardware Requirements:

- **Processor**: Dual Core or higher
- **RAM**: Minimum 2 GB (4 GB or more recommended)
- **Hard Disk**: Minimum 100 MB of free space
- **Display**: 1024x768 resolution or higher
- **Input Devices**: Keyboard (for console input)

3. Proposed Design

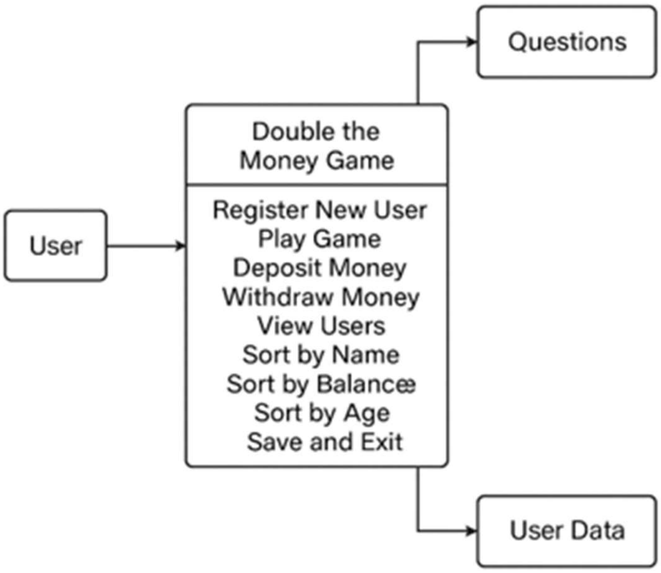


Fig 3 Proposed Design

The proposed design of the Java Gamble system consists of the following key modules:

1. **User Registration Module:**
   Allows new players to register by entering details like name, age, and gender. A starting balance is also assigned during registration.
2. **Game Module:**
   This is the core feature where users place bets and answer multiple-choice questions. Correct answers double the bet amount, while wrong answers result in loss.
3. **Deposit Module:**
   Enables users to add money to their account to continue playing if they run low on balance.

4. **Withdraw Module:**
   Allows users to withdraw money from their account, provided the balance is sufficient.
5. **User Management Module:**
   Displays all registered users along with their details (name, age, gender, balance).
6. **Sorting Module:**
   Provides sorting functionality to sort users by name, balance, or age for easier analysis and viewing.
7. **Data Persistence Module**:
   Ensures all user data and game history are saved to a file (users.txt) so that it is retained between sessions.

4. Performance & Limitations

Performance

- Efficient User Management:
  The Java Gamble system is designed to handle a reasonable number of users efficiently. Registration, sorting by name, balance, and age, and viewing users are performed quickly, ensuring that the system can scale for a moderate user base.
- Fast Game Execution:
  The betting game and question answering feature run with minimal delay, providing a smooth experience for users. Correct answers double the bet amount promptly, while incorrect answers process without noticeable lag.
- Real-Time Balance Updates:
  The deposit and withdrawal features work in real-time, updating the user's balance instantly after each transaction, ensuring immediate reflection of changes.
- Data Persistence:
  User data is saved and loaded effectively from the file (users.txt), ensuring that game progress and user details are preserved across sessions.

Limitations

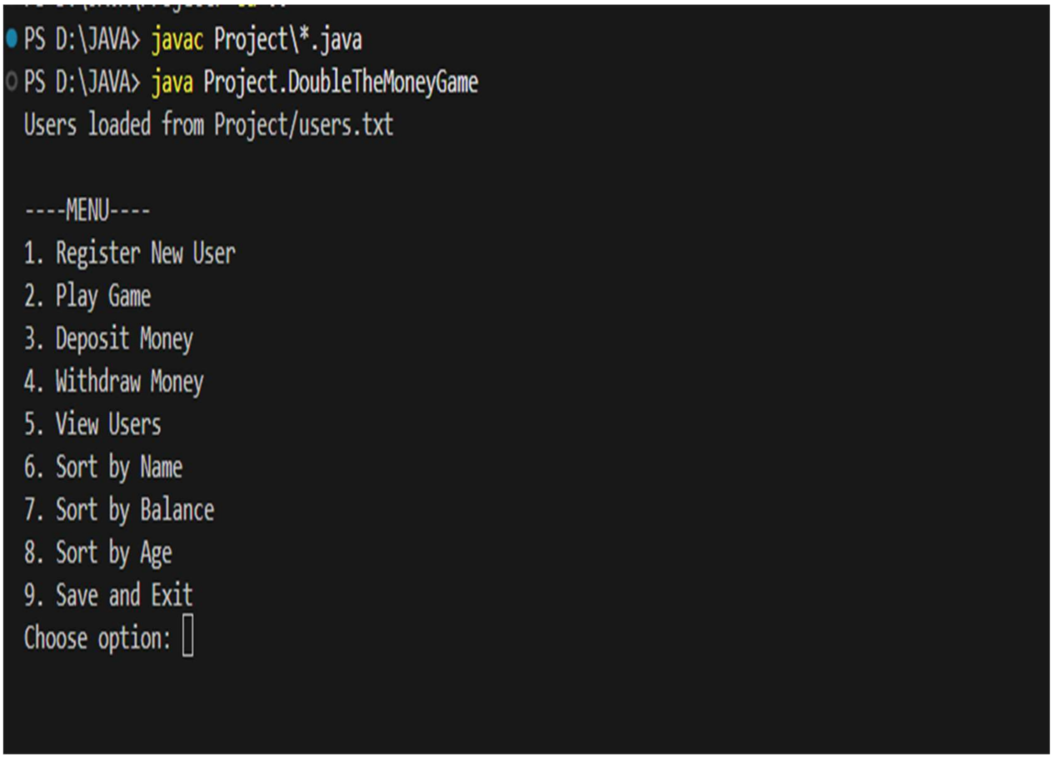
- Limited Scalability:
  As the project relies on storing user data in a text file (users.txt), it may not scale well with large numbers of users. Handling a large dataset could cause performance degradation, as text files are not ideal for large-scale database management.
- No Advanced Data Security:
  Since the system does not include encryption or secure authentication methods, user data (such as balances and personal information) may not be fully protected from unauthorized access.
- Limited Game Complexity:
  The game logic is relatively simple and could be expanded with additional features such

as varying question difficulties, multiple betting rounds, or more complex user interactions to enhance the gameplay experience.

- Single-Threaded Application:
  The system operates in a single-threaded mode, meaning it can handle one user interaction at a time. This limits the system’s ability to perform multiple operations simultaneously or handle multiple users in a more complex real-world scenario.
- Lack of Error Handling:
  Basic error handling exists, but the system does not cover all possible edge cases. For example, it might crash if the user inputs non-numeric values or exceeds predefined limits.

5. Result

5.1 Application Launch and Main Menu

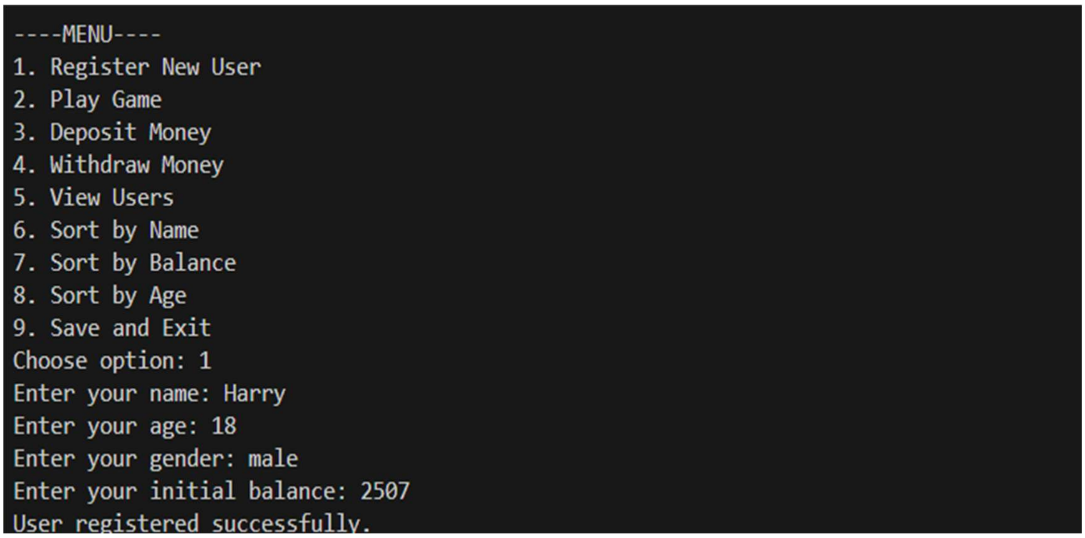


```
PS D:\JAVA> javac Project\*.java
PS D:\JAVA> java Project.DoubleTheMoneyGame
Users loaded from Project/users.txt

----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option:
```

Fig. 5.1 Application Main Menu displayed in Terminal

The user is presented with a main menu that allows them to register, play the game, deposit money, withdraw money, view user details, and sort users based on various criteria like name, balance, and age. The option to save and exit is also provided.

5.2 Registering a New User

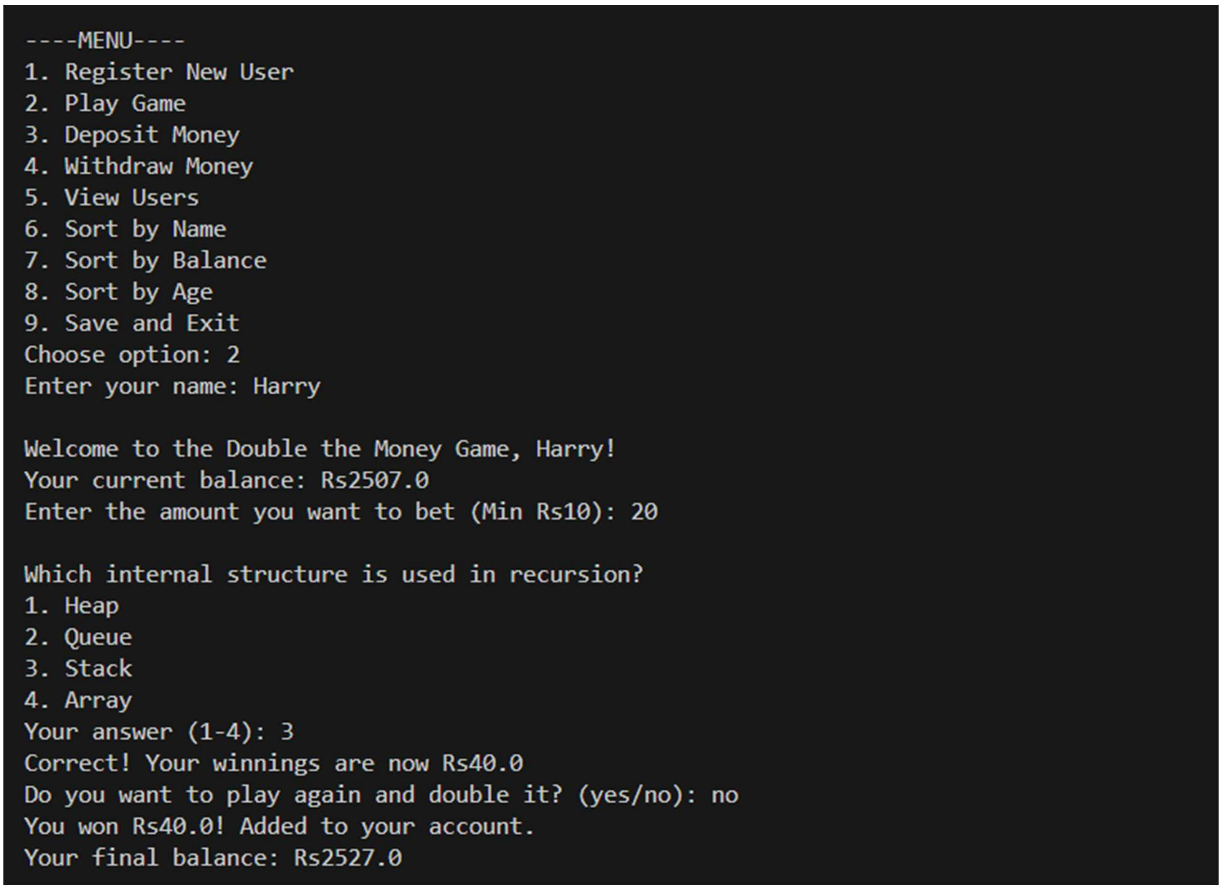


```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 1
Enter your name: Harry
Enter your age: 18
Enter your gender: male
Enter your initial balance: 2507
User registered successfully.
```

Fig. 5.2 Register New User Screen

The system allows users to register by providing their name, age, gender, and an initial balance. Once registered, the user can proceed to play the game using the same credentials.

5.3 Playing the Game

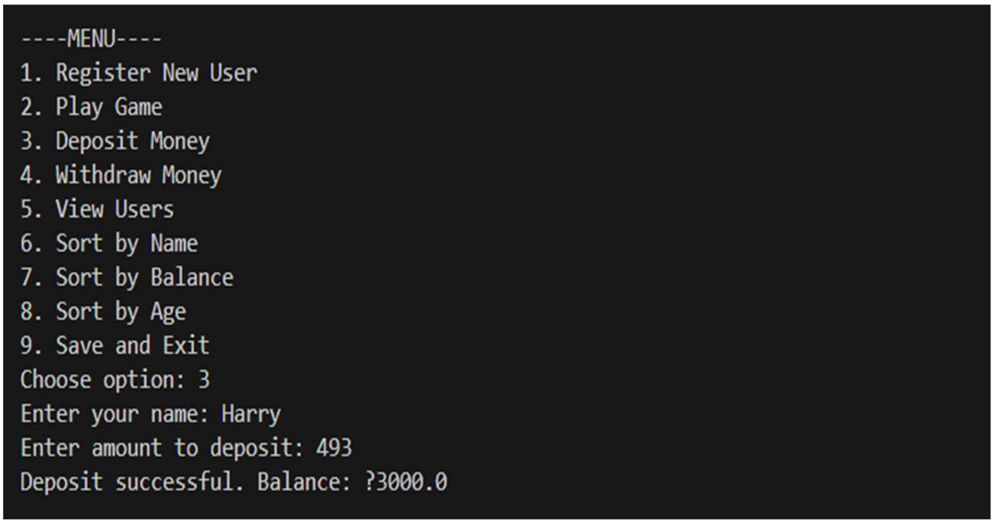


```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 2
Enter your name: Harry

Welcome to the Double the Money Game, Harry!
Your current balance: Rs2507.0
Enter the amount you want to bet (Min Rs10): 20

Which internal structure is used in recursion?
1. Heap
2. Queue
3. Stack
4. Array
Your answer (1-4): 3
Correct! Your winnings are now Rs40.0
Do you want to play again and double it? (yes/no): no
You won Rs40.0! Added to your account.
Your final balance: Rs2527.0
```

Fig. 5.3 User Playing the Game

Registered users can play the "Double the Money" game, where they answer multiple-choice questions to increase their bet. If they win, their bet doubles; if they lose, the bet is lost.

5.4 Deposit Money

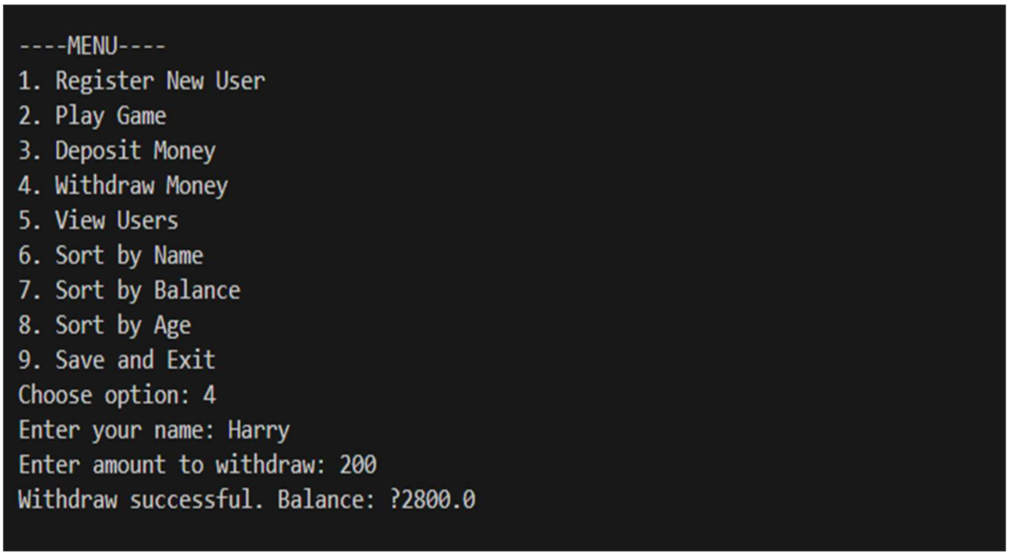


```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 3
Enter your name: Harry
Enter amount to deposit: 493
Deposit successful. Balance: ?3000.0
```

Fig. 5.4 User Depositing Money

Users can deposit money into their account, ensuring their balance increases.

5.5 Withdraw Money

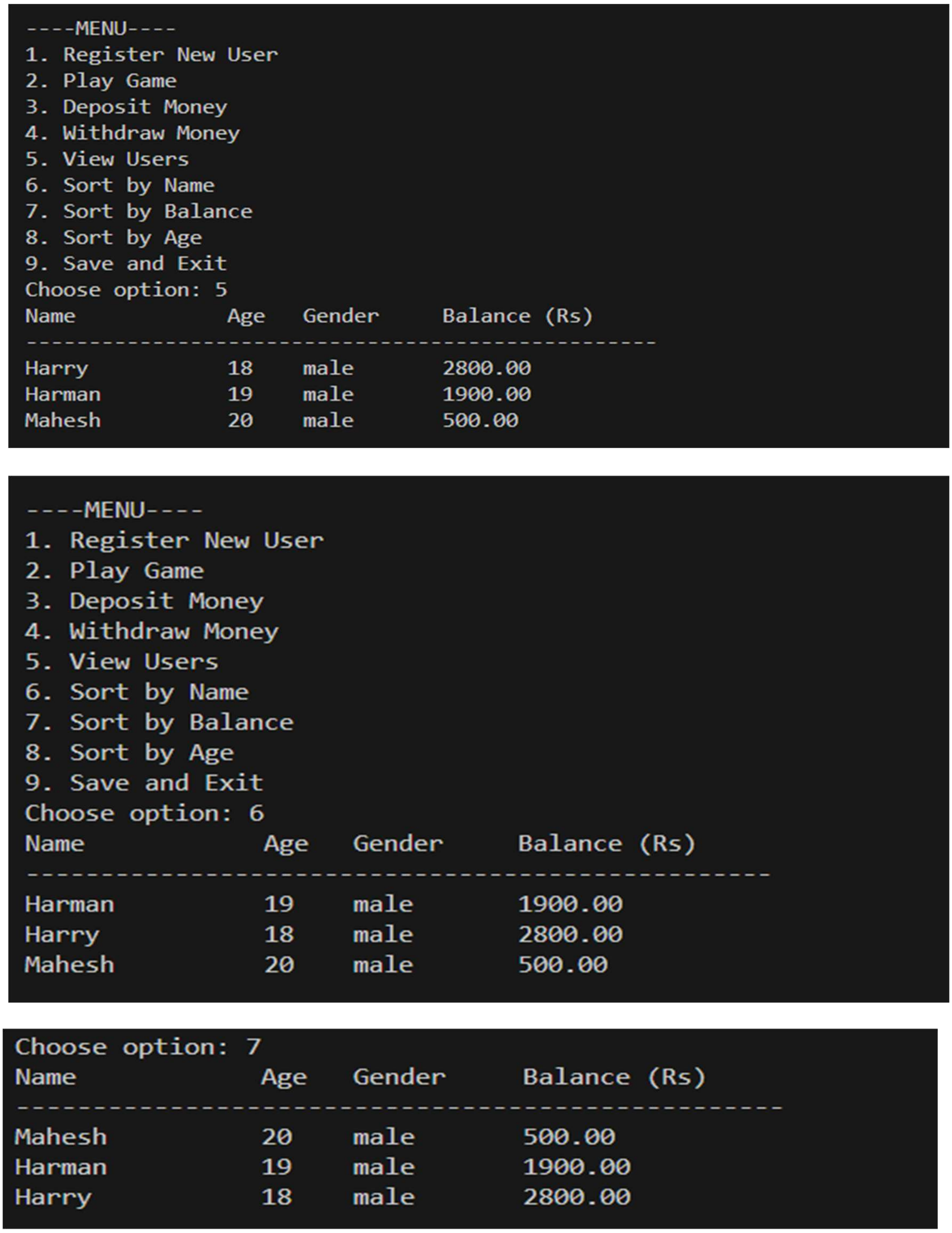


```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 4
Enter your name: Harry
Enter amount to withdraw: 200
Withdraw successful. Balance: ?2800.0
```

Fig. 5.5 User Withdrawing Money

Users can withdraw money from their account, given that they have sufficient balance to cover the withdrawal.

5.6 Displaying Users and Sorting



```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 5
Name            Age   Gender     Balance (Rs)
--------------------------------------------------
Harry           18    male       2800.00
Harman          19    male       1900.00
Mahesh          20    male       500.00
```

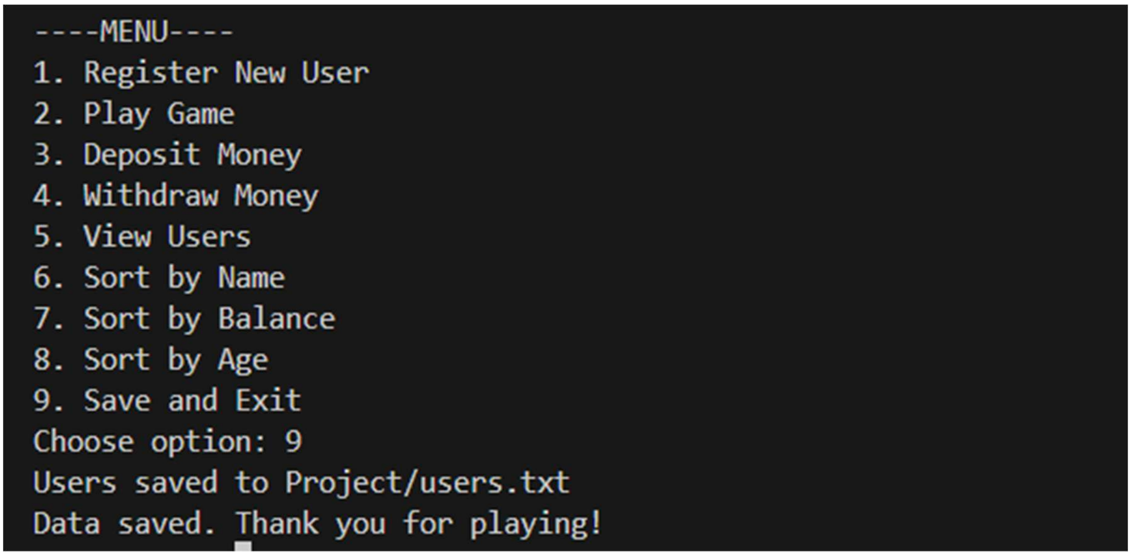
```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 6
Name            Age   Gender     Balance (Rs)
--------------------------------------------------
Harman          19    male       1900.00
Harry           18    male       2800.00
Mahesh          20    male       500.00
```

```
Choose option: 7
Name            Age   Gender     Balance (Rs)
--------------------------------------------------
Mahesh          20    male       500.00
Harman          19    male       1900.00
Harry           18    male       2800.00
```

Fig. 5.6 Top 3 Priority Tasks

The system allows the display of all registered users with details such as name, age, gender, and account balance. Users can also sort the list by name, balance, or age.

5.7 Save and exit



```
----MENU----
1. Register New User
2. Play Game
3. Deposit Money
4. Withdraw Money
5. View Users
6. Sort by Name
7. Sort by Balance
8. Sort by Age
9. Save and Exit
Choose option: 9
Users saved to Project/users.txt
Data saved. Thank you for playing!
```

Fig. 5.7 Saving and exit

User data, including account details and balances, is saved in a file for persistence. This ensures the data remains intact between application runs.

User data is stored in a file (users.txt), ensuring that user information such as account balances and gameplay history is retained for future sessions.

6. References

1. **Java Documentation**
   - **Oracle, "The Java™ Tutorials," available at: https://docs.oracle.com/javase/tutorial/**
2. **File Handling in Java**
   - **W3Schools, "Java File I/O," available at: https://www.w3schools.com/java/java_files.asp**
3. **OOP Concepts in Java**
   - **GeeksforGeeks, "Object-Oriented Programming in Java," available at: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/**
4. **Sorting Algorithms**
   - **GeeksforGeeks, "Sorting Algorithms," available at: https://www.geeksforgeeks.org/sorting-algorithms/**
5. **Java Collections Framework**
   - **Oracle, "Collections Framework," available at: https://docs.oracle.com/javase/8/docs/technotes/guides/collections/**
6. **File I/O Operations**
   - **Baeldung, "Java File I/O," available at: https://www.baeldung.com/java/java-file-io**
7. **User Input Validation in Java**
   - **Stack Overflow, "How to Validate User Input in Java," available at: https://stackoverflow.com/questions/22766198/how-to-validate-user-input-in-java**
8. **Java Data Structures and Algorithms**
   - **"Data Structures and Algorithms in Java" by Robert Lafore, 4th Edition, Pearson Education**