JUNIA Grande école d'ingénieurs

2024-2025

Java Banking System
Project Report

Réalisé par :

- LEKHBIOUI Hamza
- Mohammed Amin Moussaif

1. Overview

This project is a Java-based banking system application with a graphical user interface (GUI) built using Swing components. The project consists of three main files: main.java, SimpleGUI.java, and users.json. The project aims to provide a simple user-friendly interface for managing user information and transactions. Below, we provide a detailed explanation of each component and its functionality.

2. File Overview

2.1 main.java

The main.java file contains the main entry point of the application. It initializes the SimpleGUI class to launch the user interface. Here's the key content of this file:

```
import javax.swing.SwingUtilities;

public class main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new SimpleGUI());
    }
}
```

Purpose: The main function is responsible for starting the application. It utilizes SwingUtilities.invokeLater to ensure that the GUI is created and updated on the Event Dispatch Thread (EDT), which is recommended for Swing applications.
new SimpleGUI(): This creates an instance of the SimpleGUI class, which is where the main application logic and interface are defined.

2.2 SimpleGUI.java

The SimpleGUI.java file contains all the methods and interface elements required to operate the banking application. Key components include:

2.2.1 Imports

The file imports various Java packages, including:

java.awt.*: For defining GUI components and layout.

com.google.gson.Gson: To work with JSON data, enabling data serialization and deserialization.

java.io.*: To handle file input and output, which is critical for saving and loading user data.

java.security.*: For cryptographic operations, potentially used to secure sensitive user data.

java.time.*: For handling date and time, including LocalDate and Period for managing user transactions and dates.

javax.swing.*: For building the user interface using Swing components.

2.2.2 Variables and Components

User Interface Elements: The GUI components are created using classes such as JFrame, JButton, JTextField, and other Swing components.

User Data Handling: The use of Gson indicates that user data, such as account information and transactions, is read from and saved to JSON files.

3. Application Flow

Startup: The application starts by running main.java, which initializes the SimpleGUI class.
User Interface: The SimpleGUI class constructs the graphical interface. Users can interact with buttons, text fields, and other components to perform actions like viewing their account balance, making deposits or withdrawals, and viewing transaction history.

Data Management: User details are saved in users.json, and every operation performed by the user (such as making a transaction) is updated in this file.

4. Key Methods in SimpleGUI.java

loadUserData(): Reads user data from the users.json file and populates the user interface accordingly. This method ensures that users have access to up-to-date account information each time they launch the application.

saveUserData(): Saves user changes, such as new transactions or updated account details, to the JSON file. This ensures that data persists across sessions.

createTransaction(): Creates a new transaction for the user. This method takes the transaction details (amount, type, date) and appends them to the user's list of transactions in the users.json file.

hashPassword(): Uses MessageDigest to hash user passwords before storing them, adding a layer of security to protect user credentials.

generateAccountNumber(): Generates a unique account number for new users to ensure uniqueness and proper tracking of individual accounts.

validateUserInput(): Ensures that user input is correct and complete before proceeding with any action, such as transactions or account creation.

calculateInterest(): Computes interest based on the user's balance, helping users understand potential growth of their savings over time.

loadTransactions(): Loads all transactions from the users.json file and displays them in the user interface. This method ensures that users can view their full transaction history each time the application is launched.

loadUsers(): Loads all user data from the users.json file and returns it as a list of User objects. This method is used to initialize the application with existing users, ensuring continuity across sessions.

hashPassword(String password): Takes a plain text password and returns its hashed representation using MessageDigest. This method is essential for ensuring that user credentials are stored securely.

findUser(String username): Searches for a user by their username in the list of loaded users and returns the corresponding User object if found. This method helps in managing user-specific actions, such as authentication and account management.

5. User Interface Design

The user interface is designed using Swing components. The main features include:

Account Summary Panel: Displays the current balance and account details.
Transaction Form: A form allowing users to input transaction details, such as the amount and type (e.g., deposit or withdrawal).

Action Buttons: Buttons to confirm actions like saving data, initiating a transaction, or closing the application.
The interface aims to be intuitive, enabling users with minimal technical knowledge to manage their banking operations smoothly.

6. Data Handling and Security

The DATABASE_FILE variable points to "data/database.json", which serves as the location where all user information, including account details and transactions, are stored. This centralization of data allows for consistent access and easy management of user records.

Conclusion

This Java-based banking system project is structured to provide a simple and effective user interface for managing banking transactions. The use of Swing for the interface and JSON for data management makes the application easy to extend and maintain. However, for a production environment, further considerations around data security and user authentication would be critical.

The project demonstrates key Java concepts, such as GUI programming with Swing, file handling, and JSON data management, making it a comprehensive exercise in building a practical application.