Table of Contents

[Synopsis 1](#_Toc188964551)

[Components 2](#_Toc188964552)

[Features 2](#_Toc188964553)

[User Flow 3](#_Toc188964554)

[Technical Details 3](#_Toc188964555)

[Task Sheet 3](#_Toc188964556)

[Problem Definition 4](#_Toc188964557)

[Objective 4](#_Toc188964558)

[Input 5](#_Toc188964559)

[Output 5](#_Toc188964560)

[Constraints 5](#_Toc188964561)

[Steps 5](#_Toc188964562)

[Assumptions 5](#_Toc188964563)

[Project Analysis 5](#_Toc188964564)

[Project Structure 5](#_Toc188964565)

[Checkbook.java Class 6](#_Toc188964566)

[CheckbookGUI.java Class 6](#_Toc188964567)

[LoginGUI.java Class 7](#_Toc188964568)

[Design 7](#_Toc188964569)

[Flowchart 7](#_Toc188964570)

[User's Guide 8](#_Toc188964571)

[Introduction 8](#_Toc188964572)

[Getting Started 8](#_Toc188964573)

[Using the Checkbook 9](#_Toc188964574)

[Developer's Guide 9](#_Toc188964575)

[Introduction 9](#_Toc188964576)

[Class Overview 9](#_Toc188964577)

[Setting Up the Project 10](#_Toc188964578)

# Synopsis

**Overview**

The Check Book Application is a desktop application developed in Java using the Swing framework. It enables users to manage their checkbook transactions, such as deposits and withdrawals, through a graphical user interface. The application includes a login mechanism for authentication.

## Components

1. **LoginGUI.java:**
   * The entry point of the application.
   * Provides a user interface for logging in with a username and password.
   * If authenticated successfully, transitions to the main Checkbook management interface.
2. **Checkbook.java:**
   * The core class handling the business logic of the application.
   * Manages the current balance and a list of transactions.
   * Methods include:
     + deposit(double amount): Adds the specified amount to the balance.
     + withdraw(double amount): Subtracts the specified amount from the balance, provided there are sufficient funds.
     + getBalance(): Returns the current balance.
     + getTransactions(): Returns a list of all transactions.
3. **CheckBookGUI.java:**
   * The main user interface for managing checkbook transactions.
   * Allows users to perform deposits and withdrawals.
   * Displays the transaction history and current balance.
   * Consists of various Swing components to provide interactive features.

## Features

* **Login Screen:**
  + User authentication with predefined credentials.
    - Username: “user”.
    - Password: “1234”.
  + Transition to the main application upon successful login.
* **Checkbook Management:**
  + Deposit money into the checkbook.
  + Withdraw money from the checkbook, with checks for sufficient funds.
  + View transaction history and current balance.
  + Simple and intuitive graphical interface.

## User Flow

1. **Launch the Application:**
   * The application starts with the LoginGUI.java class, displaying the login screen.
2. **Login:**
   * Users enter their username and password.
   * Upon successful authentication, the main Checkbook interface (CheckBookGUI.java) is displayed.
3. **Perform Transactions:**
   * Users can choose to deposit or withdraw money.
   * Enter the amount and submit the transaction.
   * The transaction history and balance are updated accordingly.
4. **Exit:**
   * Users can exit the application using the exit button provided in the interface.

## Technical Details

* **Programming Language:** Java
* **Framework:** Swing
* **Classes:**
  + LoginGUI.java: Handles user authentication.
  + Checkbook.java: Manages balance and transactions.
  + CheckBookGUI.java: Provides the main user interface for transactions.

# Task Sheet

**Week 1: Planning and Setup**

* Task 1: Define project objectives.

: Write down the main goals of the project.

* Task 2: Outline functional and non-functional requirements

: List the features and quality attributes the application should have

* Task 4: Set up development environment

: Install necessary software (Java / J2EE / .NET / C /)

: Create a new project.

* Task 5: Design application structure

: Define the main components and their interactions

* Task 6: Create initial project structure

: Set up packages and create placeholder classes for main components

* Task 7: Implement basic console input/output

: Write code to handle user input and display output

**Week 2: Transaction Handling**

* Task 1: Implement the Transaction class

: Define properties amount as type (deposit or withdrawal)

: Implement methods for creating and displaying transactions

* Task 2: Implement transaction recording functionality

: Write code to add transactions to a list or array

* Task 3: Test transaction handling

: Create sample transactions and ensure they are recorded correctly

**Week 3: Balance Calculation**

* Task 1: Implement balance calculation logic

: Write code to update the balance after each transaction

* Task 2: Integrate balance calculation with transaction recording

: Ensure the balance updates correctly when transactions are added

* Task 3: Test balance calculation

: Add multiple transactions and verify the balance is correct

**Week 4: User Interface and Interaction**

* Task 1: Improve console interface

: Add prompts and clear instructions for users

* Task 2: Implement user input validation

: Ensure valid data entry for transactions

* Task 3: Test user interface and interaction

: Simulate user interactions and ensure smooth operation

**Week 5: Testing and Refinement**

* Task 1: Conduct comprehensive testing

: Test all features and functionalities

* Task 2: Fix bugs and refine code

: Address any issues found during testing

* Task 3: Finalize documentation

: Write user guide and developer documentation

# Problem Definition

## Objective

Create a check book balancing program that reads a series of transactions from input and maintains a running balance. The program will process each transaction, which consists of a type ("withdraw" or "deposit") followed by an amount, and print the updated balance after each transaction. The program will exit upon reaching the end of the input.

## Input

The input consists of multiple lines in pairs:

1. The first line of each pair specifies the type of transaction, which can be either "withdraw" or "deposit".
2. The second line of each pair specifies the amount for the transaction, represented as a floating-point number.

The input continues until the end-of-file (EOF) is reached.

## Output

The program should output the balance after each transaction in the following format:

balance: X.XX

where X.XX is the updated balance to two decimal places.

## Constraints

1. The initial balance is 0.00.
2. Each transaction will be processed in the order they appear.
3. The program should handle invalid input gracefully.
4. The program should terminate cleanly upon reaching end-of-file.

## Steps

1. Initialize the balance to 0.00.
2. Read the input transaction type ("withdraw" or "deposit").
3. Read the transaction amount.
4. Update the balance based on the transaction type.
5. Print the updated balance formatted to two decimal places.
6. Repeat steps 2-5 until end-of-file is reached.

## Assumptions

1. The input is provided in pairs of lines, where the first line is always the transaction type and the second line is the amount.
2. The transaction type will always be valid ("withdraw" or "deposit").
3. The amount will always be a valid floating-point number.

# Project Analysis

## Project Structure

The project is structured into three main components:

1. **Checkbook.java Class**: Manages the check book operations such as deposit, withdraw, and transaction history.
2. **CheckbookGUI.java Class**: Provides a graphical user interface for the check book operations.
3. **LoginGUI.java Class**: Implements a simple login interface to secure access to the check book application.

### Checkbook.java Class

**Purpose**:

* The Checkbook.java class handles the core functionality of a check book, including depositing money, withdrawing money, and maintaining a transaction history.

**Key Methods**:

* **deposit(double amount)**: Adds a specified amount to the balance and records the transaction.
* **withdraw(double amount)**: Deducts a specified amount from the balance if sufficient funds are available; otherwise, records a failed transaction.
* **getBalance()**: Returns the current balance.
* **getTransactions()**: Returns the list of transactions.

**Strengths**:

* The class is simple and focused, adhering to the single responsibility principle.
* It maintains a clear record of transactions, which is useful for auditing and debugging.

### CheckbookGUI.java Class

**Purpose**:

* The CheckbookGUI.java class provides a graphical interface for interacting with the Checkbook.java class. It allows users to deposit and withdraw money and view the transaction history.

**Key Components**:

* **JTextArea**: Displays transaction history and messages to the user.
* **JTextField**: Allows user input for transaction amounts.
* **JRadioButton**: Lets users select between deposit and withdraw actions.
* **JButton**: Various buttons for initiating transactions, starting new transactions, and exiting the application.
* **JLabel**: Displays the current balance.

**Strengths**:

* User-friendly interface with clear options for deposit and withdraw actions.
* Immediate feedback for user actions via messages and balance updates.
* Use of Button Group for mutually exclusive radio buttons, ensuring only one action is selected at a time.

### LoginGUI.java Class

**Purpose**:

* The LoginGUI.java class provides a simple login interface to restrict access to the check book application.

**Key Components**:

* **JTextField**: For entering the username.
* **JPasswordField**: For entering the password.
* **JButton**: For submitting the login credentials.

**Strengths**:

* Basic authentication is implemented to restrict access to the application.
* Clear separation between the login screen and the main application interface.

**Overall Project Analysis**

* The project demonstrates good use of object-oriented principles, such as encapsulation and separation of concerns.
* The GUI components are well-integrated with the business logic, providing a functional and interactive user experience.
* Basic validation and user feedback mechanisms are in place, contributing to the usability of the application.

# Design

## Flowchart

No amount provided – Break loop

Initialize Account with balance 0.0

Prompt user for instructions

Create Scanner object for reading user input

Print updated balance

Deposit the amount

Invalid transaction type

Read the type of transaction

Withdraw the amount

Parse the amount from the input

Close the scanner

Check if amount is provided for the transaction

Perform the transaction based on type

# User's Guide

## Introduction

The Check Book application is a simple program that allows users to manage their checkbook through a graphical user interface. Users can log in, deposit money, withdraw money, and view their transaction history and balance.

## Getting Started

1. **Launch the Application:**
   * Run the Checkbook.jar file to start the application.
2. **Login:**
   * Enter your username and password. The default username is “user” and the password is “1234”.
   * Click the "Login" button to proceed to the Checkbook management screen.

## Using the Checkbook

1. **New Transaction:**
   * Click the "New Transaction" button to reset the input fields.
   * Select either the "Deposit" or "Withdraw" option.
2. **Enter Amount:**
   * Use the numeric keypad or type directly into the text field to enter the transaction amount.
3. **Submit Transaction:**
   * Click the "Transact" button to complete the transaction.
   * A message will confirm the transaction and update your balance and transaction history.
4. **View Balance and Transactions:**
   * The balance is displayed at the bottom left of the window.
   * The transaction history is displayed in the text area in the center of the window.
5. **Exit:**
   * Click the "Exit" button to close the application.

# Developer's Guide

## Introduction

The Check Book application is built using Java Swing for the graphical user interface. It consists of three main classes: LoginGUI.java, Checkbook.java, and CheckBookGUI.java.

## Class Overview

1. **LoginGUI.java:**
   * This class provides the login interface.
   * It authenticates the user and launches the CheckBookGU.java if the credentials are correct.
2. **Checkbook.java:**
   * This class handles the business logic of the application.
   * It maintains the balance and transaction history.
   * Methods:
     + deposit(double amount): Adds the specified amount to the balance.
     + withdraw(double amount): Subtracts the specified amount from the balance if sufficient funds are available.
     + getBalance(): Returns the current balance.
     + getTransactions(): Returns the list of transactions.
3. **CheckBookGUI.java:**
   * This class provides the main interface for managing the checkbook.
   * It interacts with the Checkbook class to process transactions and update the UI.
   * Components:
     + JTextArea transaction Area: Displays transaction history.
     + JTextField amountField: Input field for transaction amounts.
     + JLabel balanceLabel: Displays the current balance.
     + JRadioButton depositButton: Option for deposit transactions.
     + JRadioButton withdrawButton: Option for withdrawal transactions.
     + JButton newTransactionButton: Resets input fields.
     + JButton transactButton: Processes the transaction.
     + JButton exitButton: Exits the application.

## Setting Up the Project

1. **Create the Project:**
   * Use an IDE like Netbeans or Eclipse.
   * Create a new Java project and add the provided Java files (LoginGUI.java, Checkbook.java, and CheckBookGUI.java) to the project.
2. **Compile and Run:**
   * Ensure all dependencies for Swing are included (most IDEs include these by default).
   * Compile the project and run the LoginGUI.java class to start the application.