QAP 1 Nicholas Power SD12

# 1. Clean Code Practices

This project follows several clean code practices. Here are three key examples:

* • Single Responsibility Principle (SRP):

Each class in the application is focused on a single task. For example, the `Account` class only manages account-level data (balance, history, holder info), while the `Bank` class is responsible for managing all accounts.

A computer screen shot of a program code

AI-generated content may be incorrect.

* • Descriptive Naming and Method Clarity:

Functions such as `checkPassword()`, `getTransactionHistory()`, and `createAccount()` are clearly named, making the code easy to read and understand without guessing what each method does.

A screen shot of a computer program

AI-generated content may be incorrect.

* • Validation and Logging:

The code includes built-in validation for all sensitive operations, such as preventing overdrafts or negative deposits. Example: `if (amount <= 0) throw new IllegalArgumentException("Amount must be positive.");`

Additionally, transaction actions are logged to memory for clarity and future retrieval.

A screen shot of a computer code

AI-generated content may be incorrect.

# 2. Project Overview and Test Cases

The Simple Banking Application is a command-line banking simulator written in Java. Users can create a new bank account using a name and password. Each account is assigned a unique 3-digit number. After logging in, users can deposit, withdraw, transfer money, view balance, and see their transaction history.

The application consists of the following classes:

- Account: Represents user data and handles balance updates and transaction logs.  
- Bank: Stores all accounts and manages account creation, lookup, and transfers.  
- BankService: Provides a middle layer of logic between the CLI and Bank.  
- Main: The interactive CLI that controls session flow and user input.

Test cases (in `BankTest.java`) include:

- Account creation and proper account number formatting  
- Login success and failure (wrong password / missing account)  
- Depositing and withdrawing money  
- Preventing overdrafts  
- Transferring funds and confirming history

# 3. Dependencies

This project uses the following dependencies, all managed by Maven and downloaded from Maven Central Repository:

- `junit-jupiter`: The JUnit 5 test framework

- `maven-surefire-plugin`: Required to run tests during Maven's `test` phase

- `java.util.logging`: Used for internal logging of events (built-in to Java)

# 4. Challenges

• Time was a challenge for this one as I mistakenly thought it was due June 2nd at 11:59.