

**King Abdulaziz University**

**Department of Information Technology**

**Faculty of Computing and Information Technology**

**CPIT-252**

**Design Patterns**

**Group Project**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Section** |
| Albaraa Ali Alghamdi | 1847268 | IT1 |
| Tariq Abdulghani Alzahrani | 2040411 | IT1 |
| Sultan Ayidh Alkharmani | 2037278 | IT1 |

**1. Introduction**

The ATM system is a simulation of traditional banking ATM operations that provides the users with security to perform their transactions. which includes checking balances, withdrawing cash, depositing funds, and transferring money between accounts. The development focuses on reliability, maintainability and usability using different design patterns.

**2. Problem Definition**

Major issues addressed by the ATM system in banking operations:

* **User Authentication**: Secure user identity verification to ensure no unauthorized access.
* **Session Management:** Ensuring transaction validity by maintaining session states.
* **Transaction Integrity:** Handling transaction processing securely and ensuring error recovery through effective rollback mechanisms.

**3. Suggested Solution**

In leading the system implementation, a Model-View-Controller (MVC) is implemented alongside other design patterns.

* **Model-View-Controller (MVC):** MVC is an architectural pattern to separate the application logic from the business logic. it improves decoupling and modularity.
* **Facade Pattern:** Provides a simple interface for users on one end while dealing with complex subsystems at the other end using unified interfaces.
* **Singleton Pattern:** Utilizes a single instance for controlling session management throughout an application which bars multiple accesses preventing duplication sessions.
* **Command Pattern:** Encapsulates requests into objects, allowing for flexible operations and provides decoupling.
* **Observer Pattern:** Implements automatic updates of the view when the model changes, ensuring real-time feedback to the user.

**4. Main Functionalities**

* **User Authentication**: Users authenticate by providing a card number and PIN, securing access to their accounts.
* **Check Balance**: Authenticated users can check their account balances.
* **Deposit Money**: Users can deposit funds into their accounts, with the system verifying the deposited amount.
* **Withdraw Money**: Allows cash withdrawals from accounts, subject to available funds.
* **Transfer Money**: Enables users to transfer funds between accounts.
* **Session Management**: Manages user sessions, ensuring that transactions are conducted within active sessions.

**5. Class Diagram**

A diagram of a computer code

Description automatically generated with medium confidence

**6. Code: Classes and Interfaces**

**Classes**

**ATMController:**

**Description:** Handles user interactions by taking in commands and modifying the view.

**Role:** Acts as the Controller in MVC and handles results from command executions.

**Pattern:** MVC, Command.

**AtmView:**

**Description:** Manages all user interactions and displays outputs.

**Role:** Serves as the View in MVC, handling all outputs directly to the user.

**Pattern:** MVC, Observer

**ATMOperationsFacade:**

**Description:** Facilitates communication between the controller and complex subsystems like authentication and transaction services.

**Role:** Abstracts complex operations into simpler method calls.

**Pattern:** Facade

**ATMSessionSingleton:**

**Description:** Manages the session state, ensuring a single active session.

**Role:** Uses Singleton to manage session states across the application.

**Pattern:** Singleton

**AccountService, AuthService, TransactionService:**

**Description:** Handle account data management, user authentication, and transaction processing respectively.

**Role:** Part of the Model in MVC, these services manage the business logic.

**Pattern:** MVC, Facade

**AccountModel:**

**Description:** Represents account data and notifies observers of changes.

**Role:** Part of the Model in MVC, this class is observed by views for updates.

**Pattern:** MVC, Observer

**Observable:**

**Description:** Base class that allows objects to be observed.

**Role:** Manages observer list and notification mechanisms.

**Pattern:** Observer

**Commands:**

**AuthenticateCommand, CheckBalanceCommand, DepositCommand, WithdrawCommand, TransferCommand**

**Description:** Encapsulate specific operations like authentication, balance checking, deposits, withdrawals, and transfers.

**Role:** Implements the Command pattern to execute operations decoupled from the user interface.

**Pattern:** Command

**Interfaces**

**Command**

**Description:** Defines an executable action such as authentication or a transaction.

**Role:** Facilitates implementation of the Command pattern, enabling flexible execution of operations.

**Pattern:** Command

**Observer**

**Description:** Interface for objects that should be notified of changes in observable objects.

**Role:** Ensures that implementing classes update based on changes in the observed objects.

**Pattern:** Observer

**7. Expected Results**

**User Authentication:** Secure login with card and PIN.

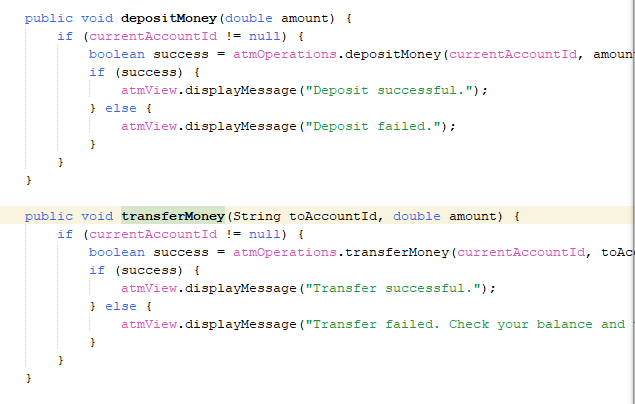
**Session Activation:** Starts a session upon successful login.

**Transaction Processing:** Execute and confirm balance checks, deposits, withdrawals, and transfers.

**Session Termination:** Ends the session, requiring re-authentication for further transactions.

Appendix:A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

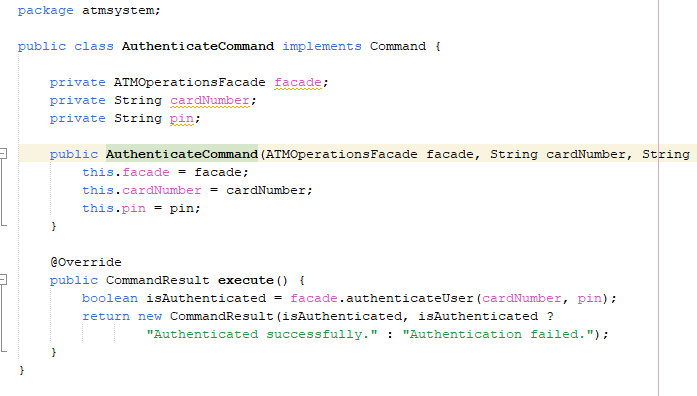
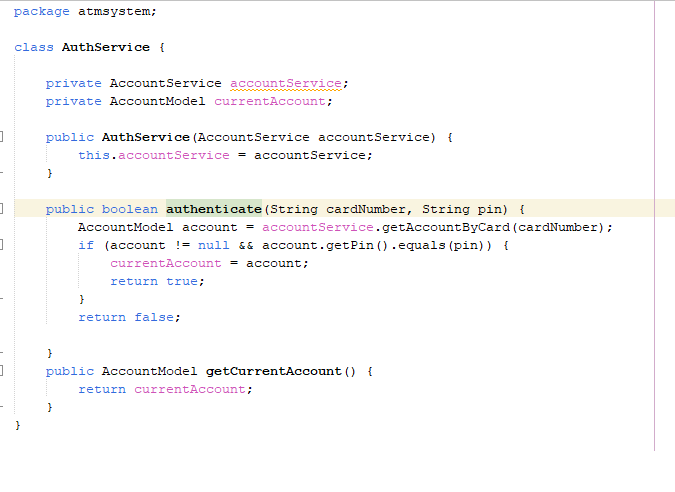
Description automatically generated

A screenshot of a computer program

Description automatically generatedA computer code with text

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated