ATM Management System with GUI

1. Project Overview

This project is a **Java-based ATM Management System** that integrates **core banking operations**, **database connectivity (MySQL via JDBC)**, and a **Graphical User Interface (GUI)** using **Swing**. The system allows users to securely log in and perform basic ATM operations such as viewing balance, depositing money, withdrawing money, and account management.

2. Key Features

- User authentication using **Customer Number and PIN**
- GUI-based interaction using **Java Swing**
- Backend database integration using **MySQL + JDBC**
- Modular design using **DAO (Data Access Object)** pattern
- Secure handling of user credentials
- Supports CRUD operations on bank accounts

3. Technologies Used

- **Programming Language:** Java
- **GUI:** Java Swing (JFrame, JButton, JTextField, JPasswordField)
- **Database:** MySQL
- **Connectivity:** JDBC (MySQL Connector/J)
- **IDE:** Eclipse / IntelliJ IDEA

4. Project Structure

ATM_WITH_GUI

|

├── ATM.java → Main class (application entry point)

├── ATM_GUI.java → Handles GUI screens and events

├── OptionMenu.java → Business logic and menu operations

├── Account.java → Account model (encapsulation)

├── AccountDAO.java → Database operations (DAO layer)

├── DBConnection.java → JDBC connection handling

└── mysql-connector.jar → JDBC Driver

5. Functional Flow

1. Application starts from ATM.java
2. Login screen appears (Customer No + PIN)
3. Credentials are validated via AccountDAO
4. On success, ATM operations screen is displayed
5. User can:
    - Check balance
    - Deposit money
    - Withdraw money
    - Exit safely

6. Object-Oriented Programming (OOP) Concepts Applied

6.1 Encapsulation

- Account-related data such as **customer number, PIN, and balance** is kept private inside the Account class.
- Access to data is provided through **getters and setters**, ensuring controlled modification.
- This protects sensitive banking data from unauthorized access.

6.2 Abstraction

- Database operations are abstracted inside the AccountDAO class.
- The GUI and main logic do not know **how** data is stored or fetched, only **what** operations are available.
- JDBC complexity is hidden from the rest of the application.

6.3 Inheritance

- The OptionMenu class extends the Account class.
- This allows reuse of account-related properties and methods without rewriting code.
- Helps in reducing redundancy and improving maintainability.

6.4 Polymorphism (Conceptual)

- Methods like login validation and transaction handling can be extended or overridden for future enhancements.
- The design allows flexible behavior without changing existing code.

7. Exception Handling Implementation

7.1 SQL Exception Handling

- Database-related errors are handled using try-catch blocks.
- Common exceptions handled:
    - SQLException
    - SQLNonTransientConnectionException
- Prevents application crash during database connection or query failures.

7.2 Class Loading Exception

- ClassNotFoundException is handled while loading the MySQL JDBC driver.
- Ensures the application reports driver issues clearly.

7.3 Input Validation Exceptions

- Invalid user inputs (wrong PIN, invalid customer number) are handled safely.
- Prevents program termination due to incorrect input formats.

7.4 Graceful Error Messages

- Instead of showing raw error traces, user-friendly messages are displayed.
- Improves usability and debugging experience.

8. Learning Outcomes

- Hands-on experience with **Java GUI development**
- Understanding of **JDBC and MySQL integration**
- Practical implementation of **OOP concepts**
- Experience in building a real-world desktop application

9. Conclusion

This ATM Management System demonstrates a complete **end-to-end Java desktop application** with GUI and database support. It is suitable for academic projects and helps in understanding how frontend, backend logic, and database layers work together in real applications.