ATM Interface

A Simulation of Banking Operations Using Java

PRESENTED BY

Logeshwari N

Tagore Engineering College

Enrollment Number : EBEON03251078045

Batch : 2024-14789

Introduction

- Objective: To develop a console-based ATM application that simulates basic banking operations.
- Purpose: Demonstrate the application of object-oriented programming (OOP) principles in Java.
- Features:
- User authentication via ID and PIN.
- Perform transactions: Deposit, Withdraw, Transfer.
- View transaction history.

Tools and Technologies Used

Programming Language: Java
Development Environment: Visual Studio Code (VS Code)
Java Development Kit (JDK): JDK 24
Extensions and Tools:
Java Extension Pack (includes Language Support, Debugger, Maven, etc.)
Java Language Pack (for UI localization)
Build Tools: Maven or Gradle (for project management and build automation)
Version Control: Git (optional, for source code management)

Project Architecture

The architecture of the ATM Interface project follows a modular, object-oriented structure. The system is designed to separate different functionalities into distinct classes, making the project easy to understand, manage, and scale. Each class performs a specific role in the ATM system.

A

Class Structure:
AccountHolder: Stores user credentials.
Account: Manages account balance.
BankTransaction: Records transaction details.
Bank: Handles core banking operations.
ATM: User interface for interaction.

B

Design Principles:
Encapsulation: Each class has specific responsibilities.
Modularity: Separation of concerns for maintainability

Class Descriptions

AccountHolder:

Attributes: userID, userPIN.
Methods: validateCredentials().

Account:

Attributes: balance.
Methods: deposit(), withdraw(), getBalance()

BankTransaction:

Attributes: transactionType, amount, date.
Methods: getTransactionDetails().

Bank:

Attributes: AccountHolder, Account, List of BankTransaction.
Methods: authenticate(), deposit(), withdraw(), transfer(), showTransactionHistory().

ATM:

Role: Main class to interact with the user and invoke Bank methods.

User Authentication

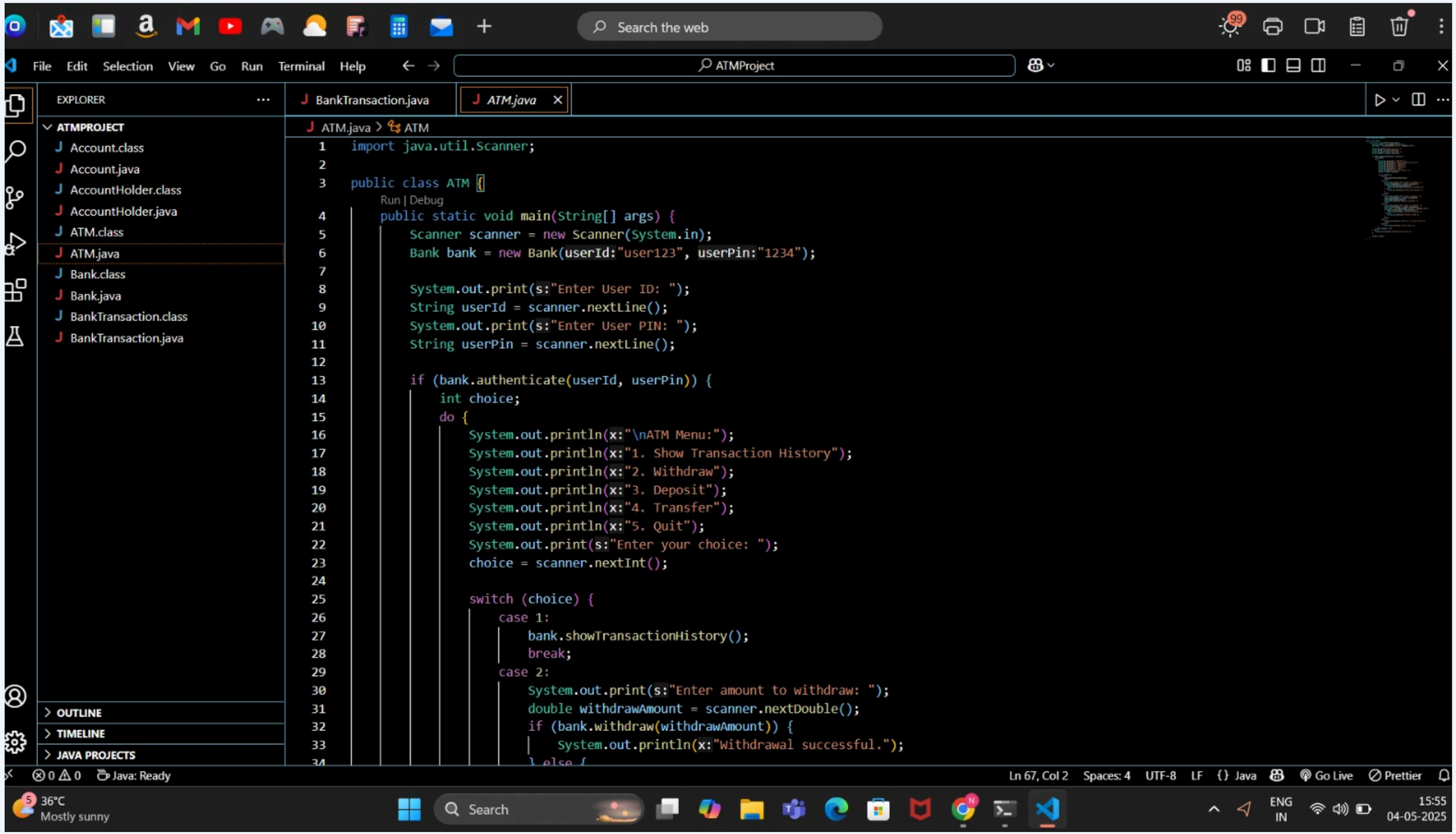
Process:

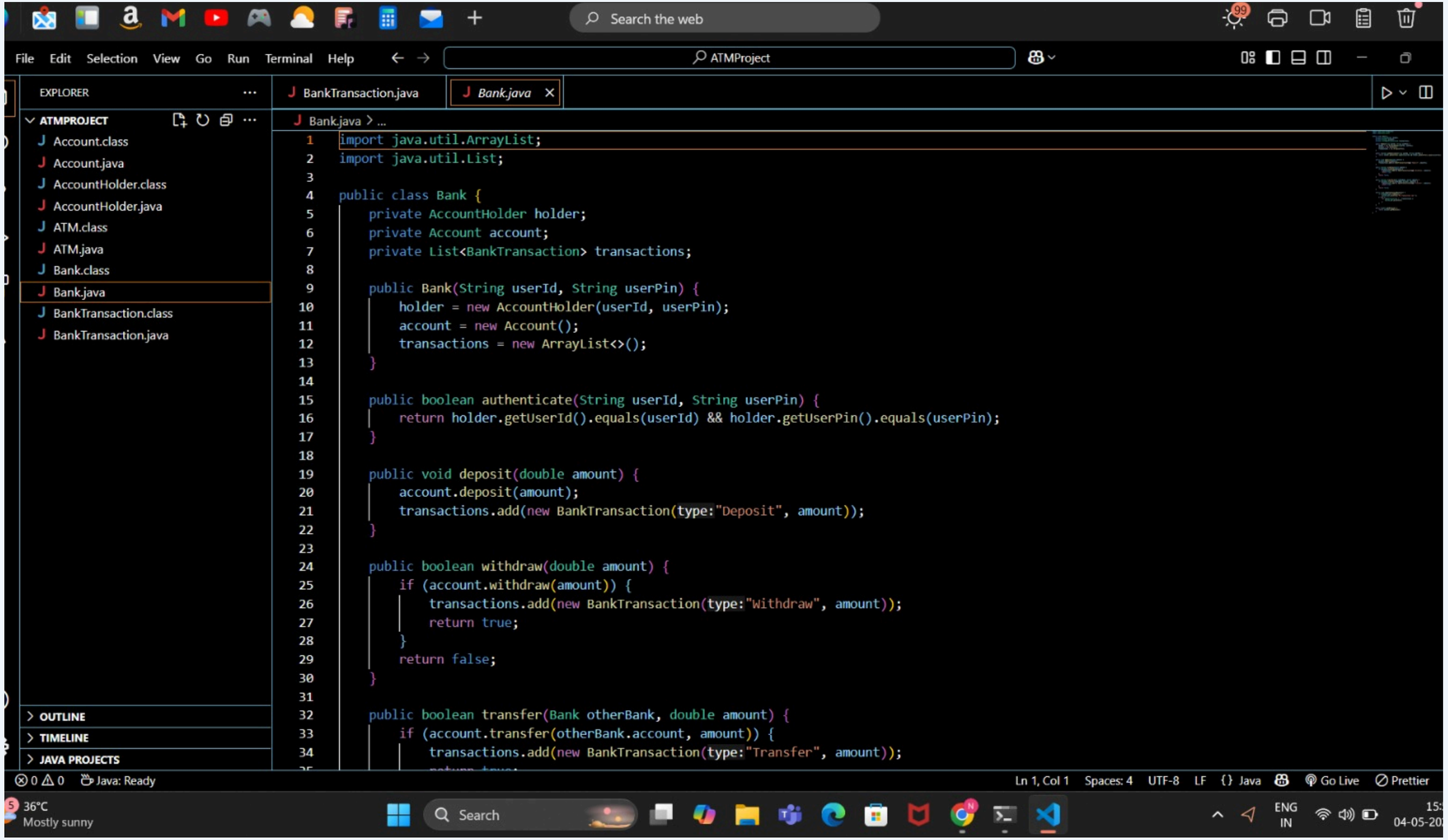
- Prompt user for ID and PIN.
- Validate credentials using AccountHolder class.
- Grant or deny access based on validation.

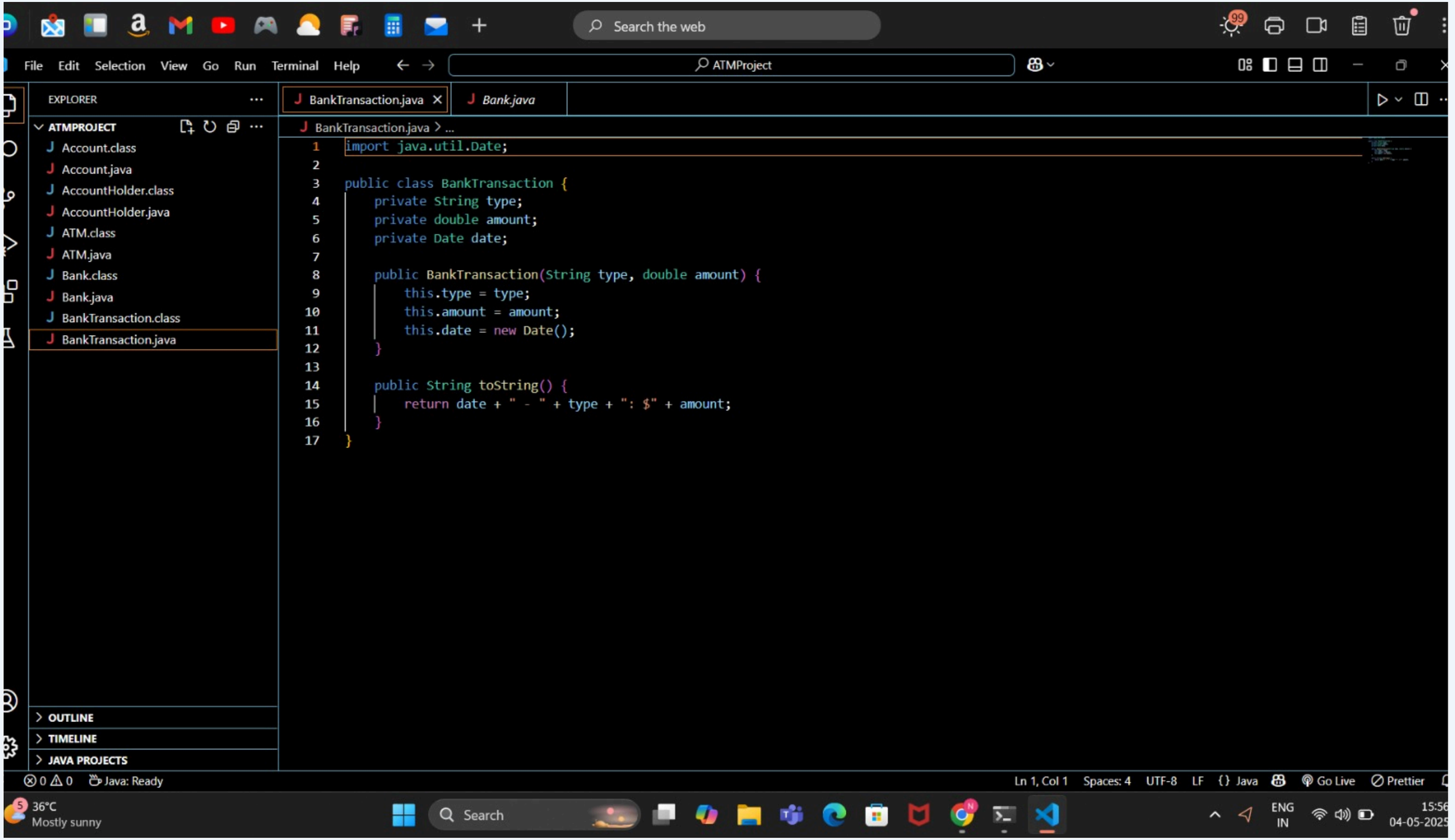
Security Consideratios:

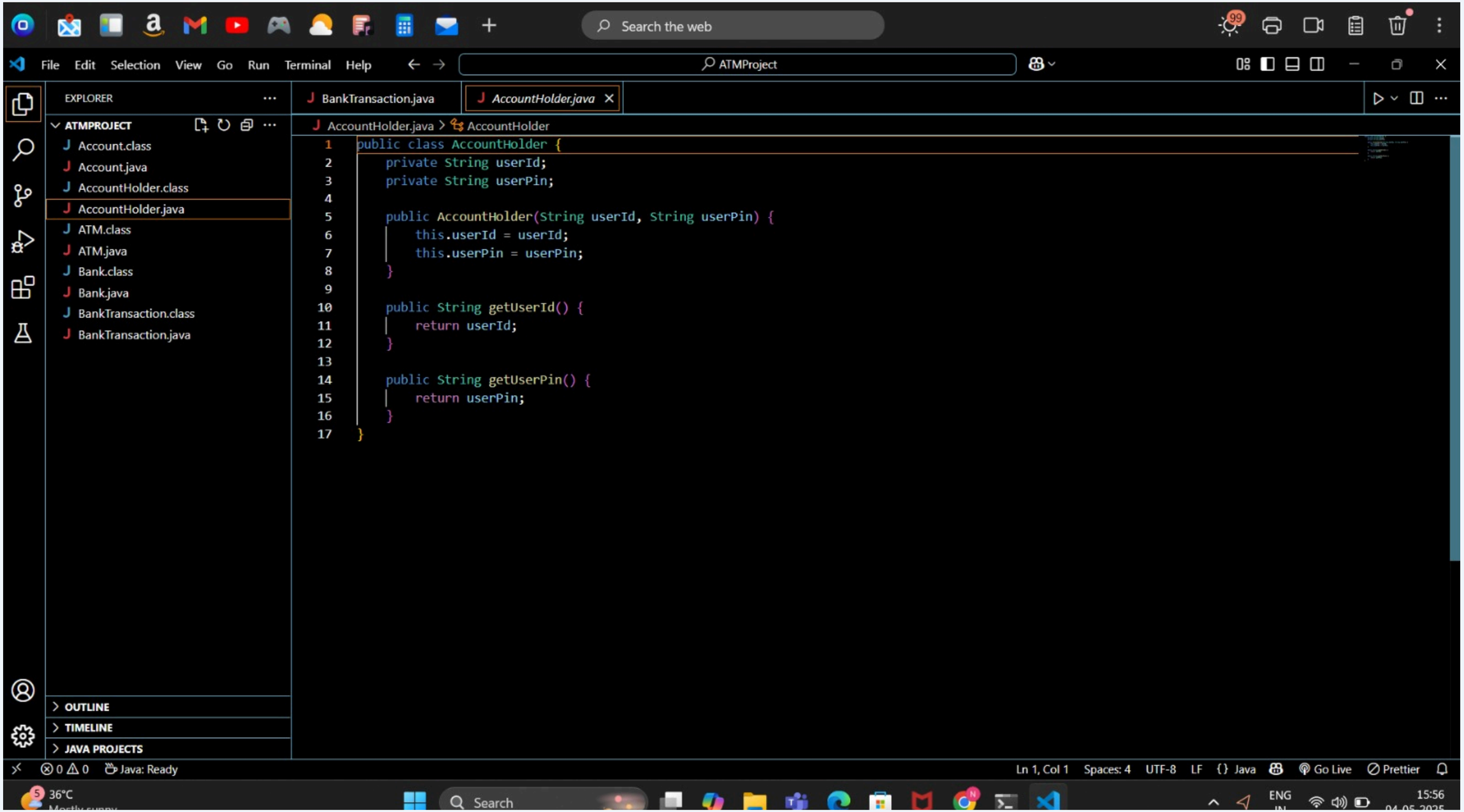
- Simple authentication for demonstration purposes.
- In real-world applications, implement secure authentication mechanisms.

CODING

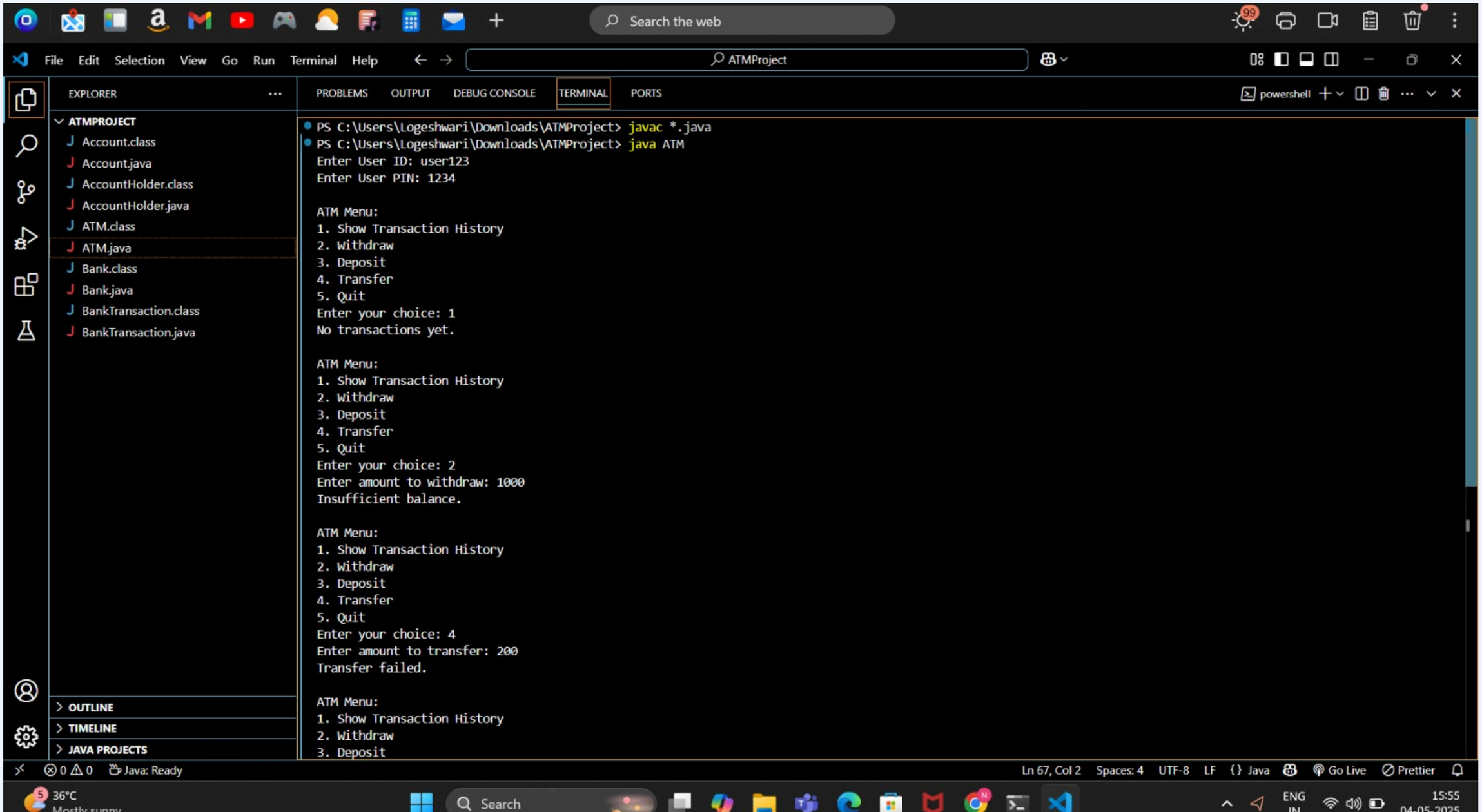

Search the web
File Edit Selection View Go Run Terminal Help
ATMProject
EXPLORER
ATMPROJECT
Account.class
Account.java
AccountHolder.class
AccountHolder.java
ATM.class
ATM.java
Bank.class
Bank.java
BankTransaction.class
BankTransaction.java
OUTLINE
TIMELINE
JAVA PROJECTS
BankTransaction.java
ATM.java
ATM.java > ATM
Run | Debug
import java.util.Scanner;
public class ATM {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
Bank bank = new Bank(userId:"user123", userPin:"1234");
System.out.print(s:"Enter User ID: ");
String userId = scanner.nextLine();
System.out.print(s:"Enter User PIN: ");
String userPin = scanner.nextLine();
if (bank.authenticate(userId, userPin)) {
int choice;
do {
System.out.println(x:"\nATM Menu:");
System.out.println(x:"1. Show Transaction History");
System.out.println(x:"2. Withdraw");
System.out.println(x:"3. Deposit");
System.out.println(x:"4. Transfer");
System.out.println(x:"5. Quit");
System.out.print(s:"Enter your choice: ");
choice = scanner.nextInt();
switch (choice) {
case 1:
bank.showTransactionHistory();
break;
case 2:
System.out.print(s:"Enter amount to withdraw: ");
double withdrawAmount = scanner.nextDouble();
if (bank.withdraw(withdrawAmount)) {
System.out.println(x:"Withdrawal successful.");
Java: Ready
Ln 67, Col 2 Spaces: 4 UTF-8 LF Java Go Live Prettier
36°C
Mostly sunny
Search
ENG
IN
15:55
04-05-2025


Search the web
File Edit Selection View Go Run Terminal Help
ATMProject
EXPLORER
ATMPROJECT
Account.class
Account.java
AccountHolder.class
AccountHolder.java
ATM.class
ATM.java
Bank.class
Bank.java
BankTransaction.class
BankTransaction.java
OUTLINE
TIMELINE
JAVA PROJECTS
BankTransaction.java
Bank.java
Bank.java > ...
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Bank {
5 private AccountHolder holder;
6 private Account account;
7 private List<BankTransaction> transactions;
8
9 public Bank(String userId, String userPin) {
10 holder = new AccountHolder(userId, userPin);
11 account = new Account();
12 transactions = new ArrayList<>();
13 }
14
15 public boolean authenticate(String userId, String userPin) {
16 return holder.getUserId().equals(userId) && holder.getUserPin().equals(userPin);
17 }
18
19 public void deposit(double amount) {
20 account.deposit(amount);
21 transactions.add(new BankTransaction(type:"Deposit", amount));
22 }
23
24 public boolean withdraw(double amount) {
25 if (account.withdraw(amount)) {
26 transactions.add(new BankTransaction(type:"Withdraw", amount));
27 return true;
28 }
29 return false;
30 }
31
32 public boolean transfer(Bank otherBank, double amount) {
33 if (account.transfer(otherBank.account, amount)) {
34 transactions.add(new BankTransaction(type:"Transfer", amount));
Java: Ready
Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Java Go Live Prettier
36°C
Mostly sunny
Search
ENG
IN


Search the web
File Edit Selection View Go Run Terminal Help
ATMProject
EXPLORER
BankTransaction.java
Bank.java
ATMPROJECT
Account.class
Account.java
AccountHolder.class
AccountHolder.java
ATM.class
ATM.java
Bank.class
Bank.java
BankTransaction.class
BankTransaction.java
OUTLINE
TIMELINE
JAVA PROJECTS
import java.util.Date;
public class BankTransaction {
private String type;
private double amount;
private Date date;
public BankTransaction(String type, double amount) {
this.type = type;
this.amount = amount;
this.date = new Date();
}
public String toString() {
return date + " - " + type + ": $" + amount;
}
}
Java: Ready
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Java
Go Live
Prettier
36°C
Mostly sunny
Search
ENG
IN
15:56
04-05-2025


Search the web
File Edit Selection View Go Run Terminal Help
ATMProject
EXPLORER
ATMPROJECT
Account.class
Account.java
AccountHolder.class
AccountHolder.java
ATM.class
ATM.java
Bank.class
Bank.java
BankTransaction.class
BankTransaction.java
BankTransaction.java
AccountHolder.java
AccountHolder.java > AccountHolder
public class AccountHolder {
private String userId;
private String userPin;
public AccountHolder(String userId, String userPin) {
this.userId = userId;
this.userPin = userPin;
}
public String getUserId() {
return userId;
}
public String getUserPin() {
return userPin;
}
}
OUTLINE
TIMELINE
JAVA PROJECTS
0 0 Java: Ready
Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Java Go Live Prettier
36°C
Search
ENG
15:56

OUTPUT


Search the web
File Edit Selection View Go Run Terminal Help
ATMProject
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
ATMPROJECT
Account.class
Account.java
AccountHolder.class
AccountHolder.java
ATM.class
ATM.java
Bank.class
Bank.java
BankTransaction.class
BankTransaction.java
OUTLINE
TIMELINE
JAVA PROJECTS
PS C:\Users\Logeshwari\Downloads\ATMProject> javac *.java
PS C:\Users\Logeshwari\Downloads\ATMProject> java ATM
Enter User ID: user123
Enter User PIN: 1234
ATM Menu:
1. Show Transaction History
2. Withdraw
3. Deposit
4. Transfer
5. Quit
Enter your choice: 1
No transactions yet.
ATM Menu:
1. Show Transaction History
2. Withdraw
3. Deposit
4. Transfer
5. Quit
Enter your choice: 2
Enter amount to withdraw: 1000
Insufficient balance.
ATM Menu:
1. Show Transaction History
2. Withdraw
3. Deposit
4. Transfer
5. Quit
Enter your choice: 4
Enter amount to transfer: 200
Transfer failed.
ATM Menu:
1. Show Transaction History
2. Withdraw
3. Deposit
Java: Ready
Ln 67, Col 2 Spaces: 4 UTF-8 LF Java Go Live Prettier
36°C
Search
ENG
15:55

Conclusion

- Successfully simulated a working ATM interface using Java.
- Effectively applied Object-Oriented Programming (OOP) principles like encapsulation and modularity.
- Gained hands-on experience with console-based user interaction and structured application logic.
- Integrate persistent storage using a database (e.g., MySQL) to store user data and transaction history.
- Upgrade from console-based UI to a Graphical User Interface (GUI) using JavaFX or Swing.

Thank you!

This project helped me learn Java and build a working ATM system.