|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ACKNOWLEDGEMENT** | I |
|  | **ABSTRACT** | II |
|  | **TABLE OF CONTENT**  Introduction | II  2 |
| 1 | UML Diagram | 3 |
| 2 | Flowchart | 4 |
| 3 | Code Of Automated Teller Machine | 5 |
| 4 | OUTPUT | 9 |
| 5 | Explanation Of Code | 12 |
|  | **CONCLUSIONS** | 16 |

Introduction

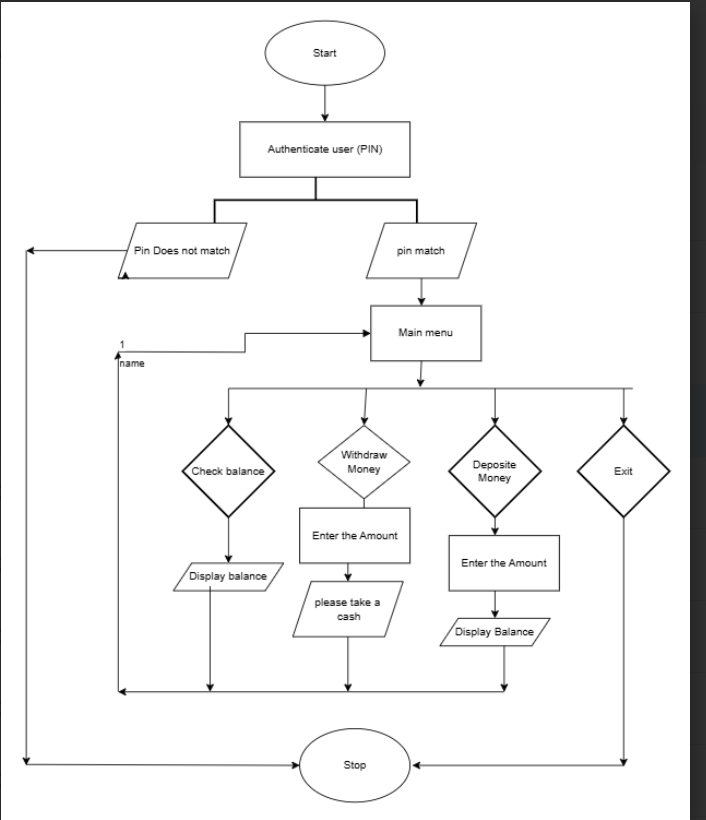
An Automated Teller Machine (ATM) is a computerized device that allows customers to perform basic banking transactions, such as cash withdrawals, deposits, fund transfers, and balance inquiries, without the need for a human teller. Introduced in the late 1960s, ATMs revolutionized banking by providing 24/7 access to banking services, which improved customer convenience and reduced the workload for bank staff.

The typical operation of an ATM involves inserting a bank card, entering a Personal Identification Number (PIN) for security, and then selecting the desired transaction. Modern ATMs are equipped with advanced features, such as cardless cash withdrawal, bill payments, and even loan processing in some cases. They are connected to a network that links banks and financial institutions, enabling customers to access their accounts from various locations around the world.

1.UML Diagram

|  |
| --- |
| Automated Teller Machine  + main(String[]): void  + authenticate(Scanner): Boolean  + checkBalance(): void  + withdrawMoney(Scanner): void  + depositMoney(Scanner): void  **+ u**serchoice: int |

2.Flowchart



3. Code of Automated Teller Machine

// Source code is decompiled from a .class file using FernFlower decompiler.

import java.util.Scanner;

public class Attm {

static int[] account = new int[]{1000, 1234};

public Attm() {

}

public static void main(String[] var0) {

Scanner var1 = new Scanner(System.in);

System.out.println("Welcome to the ATM");

if (authenticate(var1)) {

boolean var2 = false;

while (! var2) {

System.out.println("\nPlease choose an option:");

System.out.println("1. Check Balance");

System.out.println("2. Withdraw Money");

System.out.println("3. Deposit Money");

System.out.println("4. Exit");

int var3 = var1.nextInt();

switch (var3) {

case 1:

checkBalance();

break;

case 2:

withdrawMoney(var1);

break;

case 3:

depositMoney(var1);

break;

case 4:

var2 = true;

System.out.println("Thank you for using the ATM. Goodbye!");

break;

default:

System.out.println("Invalid option. Please try again.");

}

}

} else {

System.out.println("Authentication failed. Exiting.");

}

var1.close();

}

public static boolean authenticate(Scanner var0) {

System.out.print("Please enter your PIN: ");

int var1 = var0.nextInt();

return var1 == account[1];

}

public static void checkBalance() {

System.out.println("Your current balance is: $" + account[0]);

}

public static void withdrawMoney(Scanner var0) {

System.out.print("Enter amount to withdraw: ");

int var1 = var0.nextInt();

if (var1 > account[0]) {

System.out.println("Insufficient funds.");

} else if (var1 <= 0) {

System.out.println("Invalid amount.");

} else {

int[] var10000 = account;

var10000[0] -= var1;

System.out.println("Please take your cash. Your new balance is: $" + account[0]);

}

}

public static void depositMoney(Scanner var0) {

System.out.print("Enter amount to deposit: ");

int var1 = var0.nextInt();

if (var1 <= 0) {

System.out.println("Invalid amount.");

} else {

account[0] = var1 + var1;

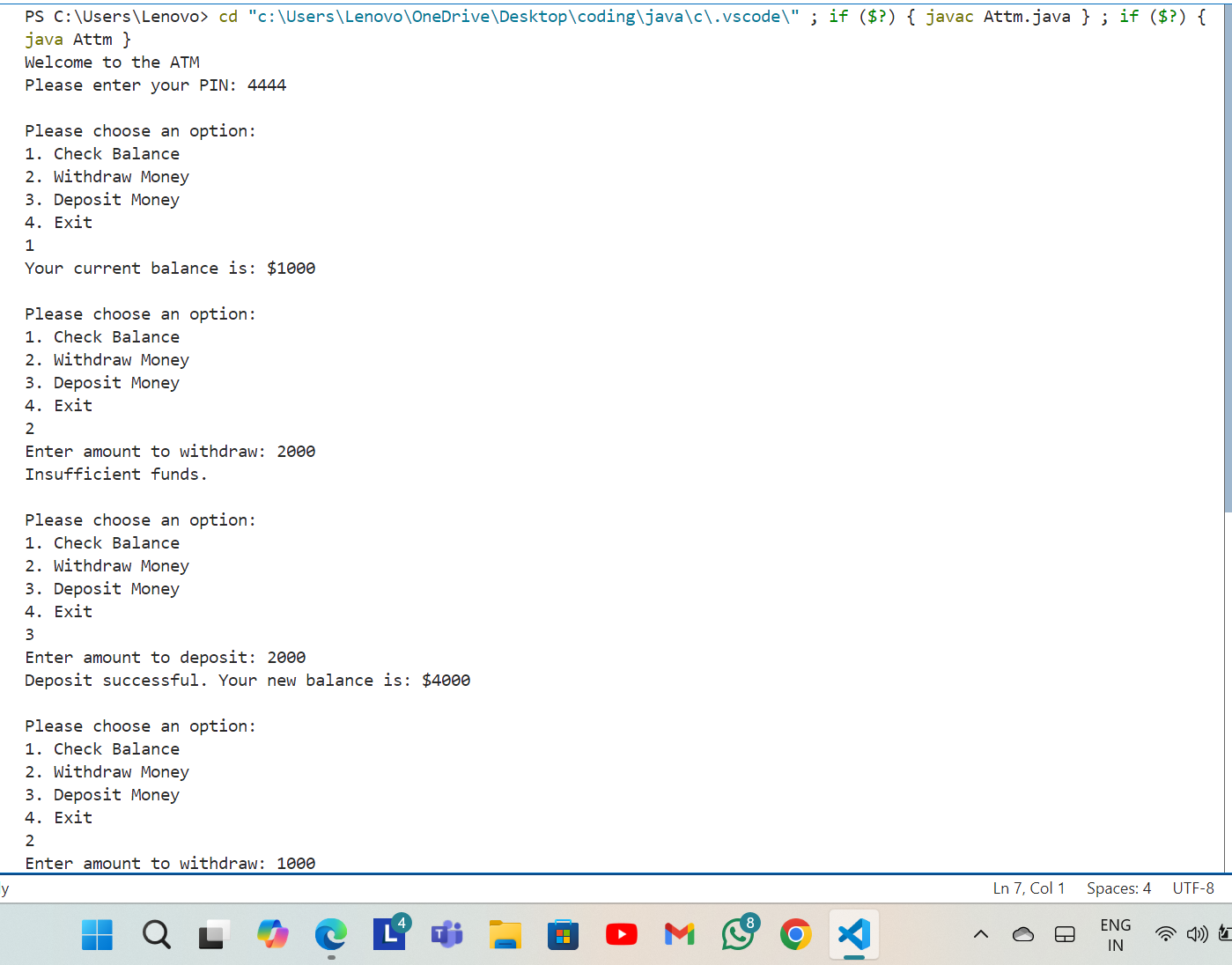
System.out.println("Deposit successful. Your new balance is: $" + account[0]);

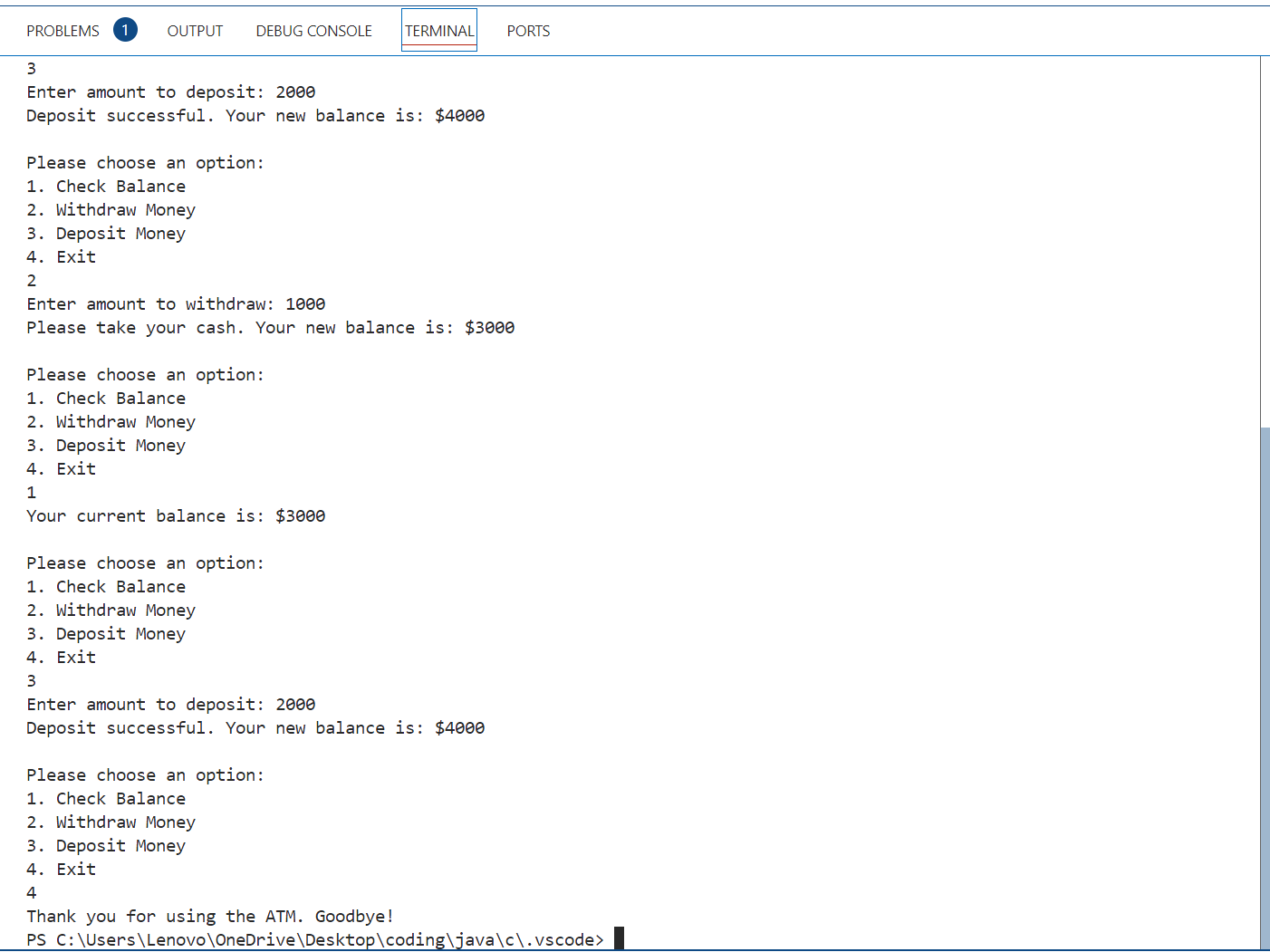
}

}

}

4.Output





.

5.**Explanation of code**

**public class Atm {**

* **Defines a public class named Atm. In Java, the file name should match the class name, so this file should be named Attm.java.**

**java**

**// Array to hold user data: [0] = balance, [1] = PIN**

**static int[] account = {1000, 4444}; // Initial balance and PIN**

* **This creates a static array account that holds user information, specifically the balance and PIN. account[0] is the initial balance (1000) and account[1] is the PIN (4444)**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**System.out.println("Welcome to the ATM");**

* **The main method is the entry point of the program.**
* **A Scanner object named scanner is created to take input from the user.**
* **A welcome message is printed to the console.**

**if (authenticate(scanner)) {**

**boolean exit = false;**

* **Calls the authenticate method to verify the user's PIN.**
* **If authentication is successful, a boolean variable exit is set to false to keep track of whether the user wants to continue using the ATM.**

**while (!exit) {**

* **Begins a loop that continues as long as exit is false.**

**System.out.println("\nPlease choose an option:");**

**System.out.println("1. Check Balance");**

**System.out.println("2. Withdraw Money");**

**System.out.println("3. Deposit Money");**

**System.out.println("4. Exit");**

* **Displays a menu with four options: check balance, withdraw money, deposit money, and exit.**

**int choice = scanner.nextInt();**

* **Reads the user's choice using scanner.nextInt().**

**switch (choice) {**

**case 1:**

**checkBalance();**

**break;**

**case 2:**

**withdrawMoney(scanner);**

**break;**

**case 3:**

**depositMoney(scanner);**

**break;**

**case 4:**

**exit = true;**

**System.out.println("Thank you for using the ATM. Goodbye!");**

**break;**

**default:**

**System.out.println("Invalid option. Please try again.");**

**}**

**}**

* **Uses a switch statement to perform actions based on the user's choice.**
  + **Case 1: Calls checkBalance() to display the current balance.**
  + **Case 2: Calls withdrawMoney(scanner) to withdraw money.**
  + **Case 3: Calls depositMoney(scanner) to deposit money.**
  + **Case 4: Sets exit to true and displays a goodbye message.**
  + **default: Handles invalid input by displaying an error message.**

**} else {**

**System.out.println("Authentication failed. Exiting.");**

**}**

**scanner.close();**

**}**

* **If the authenticate method returns false, it indicates that the user entered an incorrect PIN, so an authentication failure message is displayed.**
* **The scanner is closed to release resources.**

**Method Explanations**

**public static boolean authenticate(Scanner scanner) {**

**System.out.print("Please enter your PIN: ");**

**int enteredPin = scanner.nextInt();**

**return enteredPin == account[1];**

**}**

* **authenticate method prompts the user to enter their PIN and checks if it matches account[1]. It returns true if the PIN matches, otherwise false.**

**public static void checkBalance() {**

**System.out.println("Your current balance is: $" + account[0]);**

**}**

* **checkBalance method prints the current balance (account[0]) to the console.**

**public static void withdrawMoney(Scanner scanner) {**

**System.out.print("Enter amount to withdraw: ");**

**int amount = scanner.nextInt();**

* **withdrawMoney method asks the user for an amount to withdraw.**
  + **Checks if the amount is more than the balance (account[0]). If so, it displays an "Insufficient funds" message.**
  + **If the amount is less than or equal to zero, it displays an "Invalid amount" message.**
  + **Otherwise, it subtracts the amount from account[0] and displays the new balance.**

**public static void depositMoney(Scanner scanner) {**

**System.out.print("Enter amount to deposit: ");**

**int amount = scanner.nextInt();**

* **depositMoney method asks the user for an amount to deposit.**
  + **If the amount is less than or equal to zero, it displays an "Invalid amount" message.**
  + **Otherwise, it adds the amount to account[0] and displays the new balance.**

**However, note that the line account[0] = amount + amount; should be corrected to account[0] += amount; to add the deposit to the balance, rather than doubling the deposit amount**

**CONCLUSIONS**

**In conclusion, the ATM program you wrote is a good starting point for simulating a basic ATM system. It successfully handles user authentication, balance checking, withdrawals, and deposits. The primary issue was in the deposit function, where you mistakenly doubled the deposit amount. Once that was corrected to properly add the deposit to the balance (account[0] += amount;), the program should function as expected.**

**This basic ATM system could be expanded with additional features, such as handling multiple accounts, adding transaction history, or improving user input validation. But for now, with the corrected deposit functionality, it should work well for its intended purpose**