GLEN COSMAS KAMAU DSE-02-8598/2024

```
#include <iostream>
#include <string>
#include <memory>
using namespace std;

//  Base Class for Transactions
class Transaction {
public:
    virtual void execute() = 0; // Pure virtual function (polymorphism)
    virtual ~Transaction() {}
};

// Account Class
class Account {
private:
    string accountNumber;
    string pin;
    double balance;

public:
    Account(string accNum, string pinCode, double initialBalance)
        : accountNumber(accNum), pin(pinCode), balance(initialBalance) {}

    bool verifyPin(const string& enteredPin) const {
        return pin == enteredPin;
    }

    double getBalance() const {
```

```
        return balance;
    }

    void deposit(double amount) {
        balance += amount;
    }

    bool withdraw(double amount) {
        if (amount > balance) {
            return false; // Insufficient funds
        }
        balance -= amount;
        return true;
    }

    string getAccountNumber() const {
        return accountNumber;
    }
};

// Derived Class for Deposit
class Deposit : public Transaction {
private:
    Account* account;
    double amount;

public:
    Deposit(Account* acc, double amt) : account(acc), amount(amt) {}
```

```
    void execute() override {
        account->deposit(amount);
        cout << "Successfully deposited $" << amount << ". New Balance: $" << account->getBalance() << endl;
    }
};

// Derived Class for Withdrawal
class Withdrawal : public Transaction {
private:
    Account* account;
    double amount;

public:
    Withdrawal(Account* acc, double amt) : account(acc), amount(amt) {}

    void execute() override {
        if (account->withdraw(amount)) {
            cout << "Successfully withdrew $" << amount << ". New Balance: $" << account->getBalance() << endl;
        } else {
            cout << "Insufficient funds. Withdrawal failed." << endl;
        }
    }
};

// ATM Class
class ATM {
private:
```

```
    Account* account;

public:
    ATM(Account* acc) : account(acc) {}

    void displayMenu() {
        cout << "\nATM Menu:\n";
        cout << "1. Check Balance\n";
        cout << "2. Deposit\n";
        cout << "3. Withdraw\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
    }

    void start() {
        string enteredPin;
        cout << "Welcome to the ATM. Please enter your PIN: ";
        cin >> enteredPin;

        if (!account->verifyPin(enteredPin)) {
            cout << "Invalid PIN. Access Denied.\n";
            return;
        }

        int choice;
        do {
            displayMenu();
            cin >> choice;
```

```
    switch (choice) {
        case 1:
            cout << "Your current balance is: $" << account->getBalance() << endl;
            break;
        case 2: {
            double depositAmount;
            cout << "Enter amount to deposit: ";
            cin >> depositAmount;

            unique_ptr<Transaction> deposit(new Deposit(account, depositAmount));
            deposit->execute();
            break;
        }
        case 3: {
            double withdrawalAmount;
            cout << "Enter amount to withdraw: ";
            cin >> withdrawalAmount;

            unique_ptr<Transaction> withdrawal(new Withdrawal(account, withdrawalAmount));
            withdrawal->execute();
            break;
        }
        case 4:
            cout << "Thank you for using the ATM. Goodbye!\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
    }
} while (choice != 4);
```

```
    }
};

int main() {
    // Create an Account with initial balance
    Account myAccount("123456789", "1234", 1000.0);

    // Create an ATM instance with the Account
    ATM atm(&myAccount);

    // Start the ATM simulation
    atm.start();

    return 0;
}
```