The University of Azad Jammu and Kashmir, Muzaffarabad

Name	Kamal Ali Akmal
Course Name	Data Structure & Algorithm
Submitted to	Engr. Sidra Rafique
Semester	3rd
Session	2024-2028
Roll No	2024-SE-38
Lab No	03
Submission date	29 October 2025

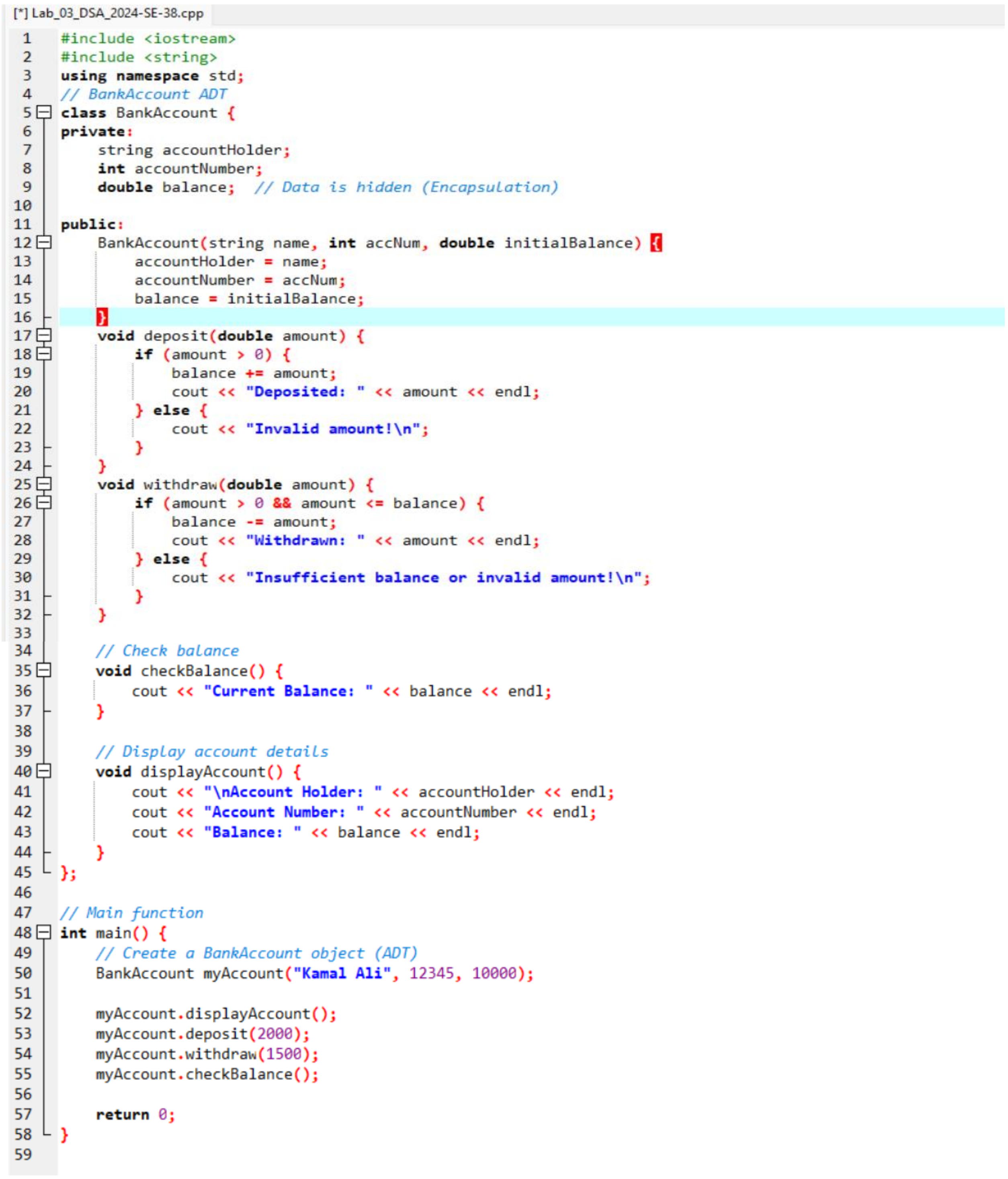
Department of Software Engineering

Abstract Data Type (ADT) – Bank Account System

Objective:

To design and implement an Abstract Data Type (ADT) for a real-world application using C++. This program demonstrates abstraction and encapsulation through a simple Bank Account system.

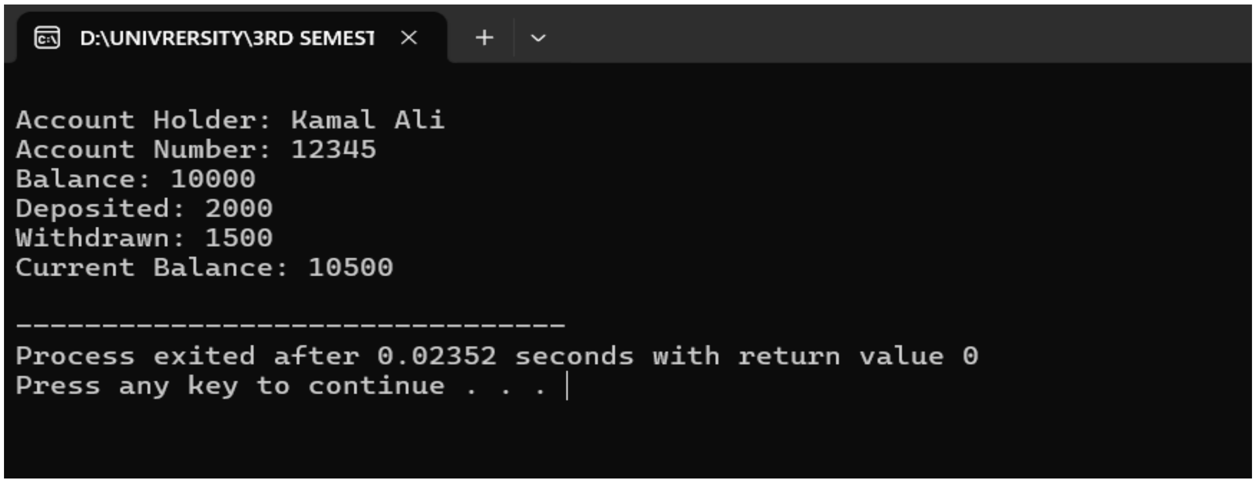
Code:



[*] Lab_03_DSA_2024-SE-38.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;
// BankAccount ADT
class BankAccount {
private:
    string accountHolder;
    int accountNumber;
    double balance;  // Data is hidden (Encapsulation)

public:
    BankAccount(string name, int accNum, double initialBalance) {
        accountHolder = name;
        accountNumber = accNum;
        balance = initialBalance;
    }
    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            cout << "Deposited: " << amount << endl;
        } else {
            cout << "Invalid amount!\n";
        }
    }
    void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            cout << "Withdrawn: " << amount << endl;
        } else {
            cout << "Insufficient balance or invalid amount!\n";
        }
    }

    // Check balance
    void checkBalance() {
        cout << "Current Balance: " << balance << endl;
    }

    // Display account details
    void displayAccount() {
        cout << "\nAccount Holder: " << accountHolder << endl;
        cout << "Account Number: " << accountNumber << endl;
        cout << "Balance: " << balance << endl;
    }
};

// Main function
int main() {
    // Create a BankAccount object (ADT)
    BankAccount myAccount("Kamal Ali", 12345, 10000);

    myAccount.displayAccount();
    myAccount.deposit(2000);
    myAccount.withdraw(1500);
    myAccount.checkBalance();

    return 0;
}

```

Output Example:



```
Account Holder: Kamal Ali
Account Number: 12345
Balance: 10000
Deposited: 2000
Withdrawn: 1500
Current Balance: 10500

--------------------------------
Process exited after 0.02352 seconds with return value 0
Press any key to continue . . .
```

Explanation:

1. The class `BankAccount` represents the Abstract Data Type (ADT).

2. Private members (accountHolder, accountNumber, balance) store data internally.

3. Public functions (deposit, withdraw, checkBalance, displayAccount) allow safe interaction with data.

4. The implementation details are hidden, showing only the essential operations to the user.

Operations Supported by the ADT:

• Deposit money – Adds amount to the account balance.

• Withdraw money – Subtracts amount if balance is sufficient.

• Check balance – Displays current available balance.

• Display account details – Shows account holder information and current balance.

Demonstration of Abstraction and Encapsulation:

• Encapsulation: The data members are private and cannot be accessed directly outside the class. Only class methods can modify them, ensuring data protection.

• Abstraction: The user interacts through simple operations like deposit () and withdraw(), without knowing the internal implementation details.

Conclusion:

The Bank Account ADT provides a clean and modular representation of a real-world system. It defines necessary operations while hiding implementation details, demonstrating the key principles of abstraction and encapsulation in the context of Data Structures and Object-Oriented Programming.