NAME : GONDALIYA MIHIR DILIPBHAI

ID NO: 25IT027

Problem Definition: A digital payment company wants to develop a simple account management module for its users. Each user maintains a digital wallet that allows loading money, transferring money to another user, and checking the current wallet balance. Each wallet has:

- A unique wallet ID
- User name
- Current balance

The system should allow:

- **Loading money into the wallet**
- **Transferring money to another wallet only if sufficient balance exists**
- **Displaying wallet details on request**

If transfer amount exceeds the current balance, the system should show an appropriate error message. The solution must be implemented using Object-Oriented Programming principles, especially encapsulation and data hiding.

Learning Outcome : A class is a blueprint or template used to create objects. It contains data members (variables) and member functions .An object is an instance of a class. It represents a real-world entity and can access the data and functions of the class.

In a class, data members are declared in the private section and public member functions are declared in the public section.

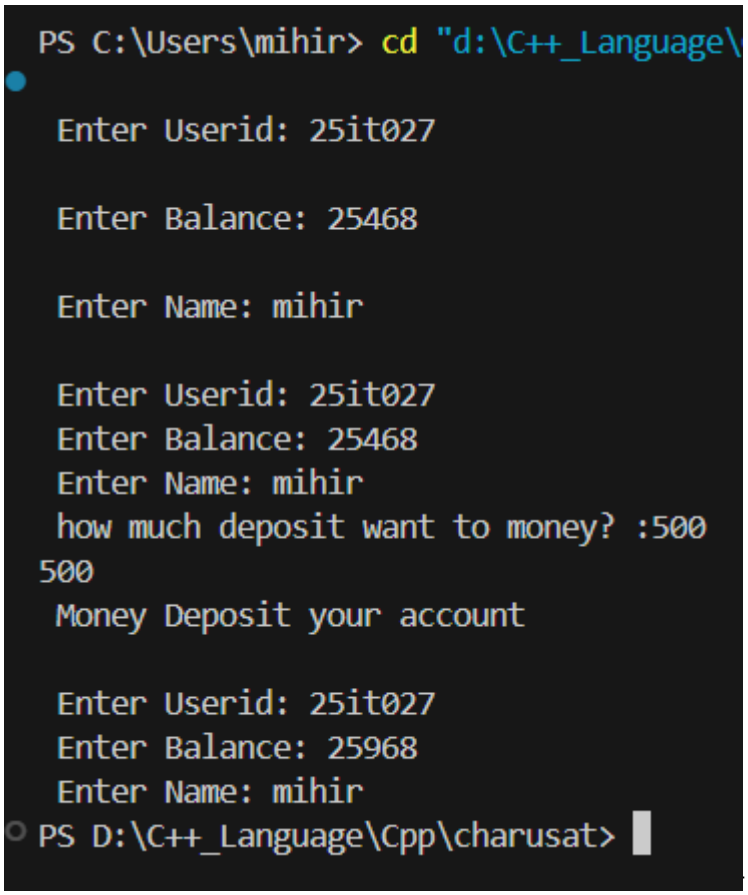
- The private section contains data members so that the data remains secure and cannot be accessed directly from outside the class.
- The public section contains member functions, which are used to access and modify the private data members.

Code :-

```cpp
#include <iostream>

class account
{
private: // Data Member
    std::string user_id;
    double balance;
    std::string name;

public: // Member Function
    void input()
    {
        std::cout << "\n Enter Userid: ";
        std::cin >> user_id;
        std::cout << " \n Enter Balance: ";
        std::cin >> balance;
        std::cout << "\n Enter Name: ";
        std::cin >> name;
    }
    void output()
    {
        std::cout << "\n Enter Userid: " << user_id;
        std::cout << " \n Enter Balance: " << balance;
        std::cout << "\n Enter Name: " << name;
    }
    void chkBalance()
    {

        std::cout << "\n Your Balance is: " << balance << std::endl;
    }
    void MoneyTran()
    {
        float amount;
        std::cout << "\n how much Trancfer want to money? :";
        std::cin >> amount;
        balance -= amount;
        std::cout << amount << "\n Money Trancfer your account " << std::endl;
```

```
    void chkBalance()
    }
    void MoneyTran()
    {
        float amount;
        std::cout << "\n how much Trancfer want to money? :";
        std::cin >> amount;
        balance -= amount;
        std::cout << amount << "\n Money Trancfer your account " << std::endl;
    }
    void MoneyDepo()
    {
        float amount;
        std::cout << "\n how much deposit want to money? :";
        std::cin >> amount;
        balance += amount;
        std::cout << amount << "\n Money Deposit your account " << std::endl;
    }
};

int main()
{
    account MIHIR;
    MIHIR.input();
    MIHIR.output();
    MIHIR.MoneyDepo();
    MIHIR.output();
    return 0;
}
```

OUT_PUT:



1. Key Questions:

- 1. How is encapsulation used to protect user balance?

ANS : Encapsulation protects the user balance by keeping it as a private data member inside the class. This prevents direct access to the balance from outside the class.The balance can only be accessed or modified using public member functions, which ensures data security.

- 2. How are member functions used to validate transactions?

  ANS : Member functions check certain conditions before performing a transaction. For example, the verify that the deposit amount is positive and that the withdrawal amount does not exceed the available balance. Only valid transactions are allowed, which helps prevent errors.

- 3. How does the system ensure that balance cannot be modified directly?

  ANS : The system declares the balance variable as **private**.
  Because private members cannot be accessed outside the class, the balance cannot be modified directly.
  Any change to the balance must be done through **public member functions**, ensuring controlled access.