1. WALLET SYSTEM

**Submitted by:**

Pier Vincent D. Caingcoy

**Submitted to :**

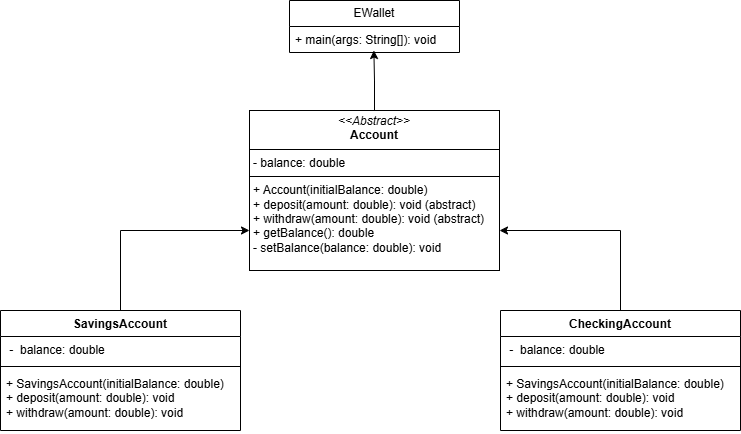
Francisco B. Bacamante Jr.

**Program:**

BSIT-2A (2022-2025)

**DESCRIPTION**

This program simulates an **E-Wallet** system that manages two types of accounts: **Savings Account** and **Checking Account**. It uses **inheritance**, **encapsulation**, and **polymorphism** in object-oriented programming (OOP) to handle account balances, deposits, and withdrawals for both account types. The user can interact with the program by choosing options to view balances or perform transactions.

**UML**

**CODES**

import java.util.Scanner;

abstract class Account {//Abstract

    private double balance;  // Encapsulation

    public Account(double initialBalance) { // Constructor

        if (initialBalance >= 0) {

            this.balance = initialBalance;

        } else {

            this.balance = 0;

        }

    }

    // Abstract methods

    public abstract void deposit(double amount);

    public abstract void withdraw(double amount);

    public double getBalance() { // Getter method

        return balance;

    }

    protected void setBalance(double balance) { // Setter method

        this.balance = balance;

    }

}

class SavingsAccount extends Account {// SavingsAccount class inherits Account class

    public SavingsAccount(double initialBalance) {

        super(initialBalance);

    }

    @Override

    public void deposit(double amount) {

        if (amount > 0) {

            setBalance(getBalance() + amount);

            //System.out.println("Deposited " + amount + " into Savings Account.");

        } else {

            System.out.println("Deposit amount must be positive.");

        }

    }

    @Override

    public void withdraw(double amount) {

        if (amount > 0 && amount <= getBalance()) {

            setBalance(getBalance() - amount);

           // System.out.println("Withdrew " + amount + " from Savings Account.");

        } else {

            System.out.println("Insufficient balance or invalid amount.");

        }

    }

}

class CheckingAccount extends Account {// CheckingAccount class inherits Account class

    public CheckingAccount(double initialBalance) {

        super(initialBalance);

    }

    @Override

    public void deposit(double amount) {

        if (amount > 0) {

            setBalance(getBalance() + amount);

            //System.out.println("Deposited " + amount + " into Checking Account.");

        } else {

            System.out.println("Deposit amount must be positive.");

        }

    }

    @Override

    public void withdraw(double amount) {

        if (amount > 0 && amount <= getBalance()) {

            setBalance(getBalance() - amount);

           // System.out.println("Withdrew " + amount + " from Checking Account.");

        } else {

            System.out.println("Insufficient balance or invalid amount.");

        }

    }

}

public class EWallet

{

    public static void main(String[] args) {

       Scanner s = new Scanner(System.in);

        Account savings = new SavingsAccount(1000);

        Account checking = new CheckingAccount(500);

        System.out.println("-------EWALLET-------");

        System.out.println("| 1. SBalance        |");

        System.out.println("| 2. CBalance        |");

        System.out.println("| 3. Savings Account |");

        System.out.println("| 4. Checking Account|");

        System.out.println("| 5. Exit            |");

        System.out.println("----------------------");

        while(true){

        System.out.print("Enter number: ");

        int choice = s.nextInt();

        switch(choice){

             case 1:

             System.out.println("Current Balance: " + savings.getBalance());

             break;

             case 2:

             System.out.println("Current Balance: " + checking.getBalance());

             break;

             case 3:

             System.out.print("Deposit        :");

             int deposit = s.nextInt();

             savings.deposit(deposit);

             break;

             case 4:

             System.out.print("Withdraw       :");

             int withdraw = s.nextInt();

             savings.withdraw(withdraw);

             break;

             case 5:

             System.out.println("Exiting EWallet.");

             return;

             default:

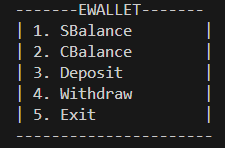
             System.out.println("Invalid choice.");

             }

            }

        }

    }

**OUTPUTS**

This is the E-Wallet System where you can see a different option which is:

**Option 1:** Display the current balance of the Savings account.

**Option 2:** Display the current balance of the Checking account.

**Option 3:** Deposit money into the Savings account.

**Option 4:** Withdraw money from the Savings account.

**Option 5:** Exit the program.



The user is prompted to enter a choice. If they choose **1**, the current balance of the savings account will be shown (1000.0 in this case).



If they choose **2**, the current balance of the checking account will be shown (500.0 in this case).



If the user chooses **3** (Savings Account deposit), the program will prompt them to enter an amount to deposit.

If they enter a valid amount, the deposit will be processed, and the new balance will be reflected in the account. If the user enters a non-positive value, an error message will be shown.





If the user chooses **4** (Savings Account withdrawal), the program will prompt them to enter the amount to withdraw.

If the withdrawal amount is greater than 0 and less than or equal to the balance, the withdrawal will be successful, and the new balance will be shown. If the user enters an invalid amount (example, more than the current balance or a negative value), an error message will appear.



If the user chooses **5**, the program will print "Exiting EWallet." and terminate.