public class BankCard {

protected double balance;

protected double interestRate;

public BankCard(double balance, double interestRate) {

this.balance = balance;

this.interestRate = interestRate;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

} else {

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

}

}

public void addInterest() {

balance \*= (1 + interestRate);

}

}

public class CreditCard extends BankCard {

private double creditLimit;

public CreditCard(double balance, double interestRate, double creditLimit) {

super(balance, interestRate);

this.creditLimit = creditLimit;

}

public void makePayment(double amount) {

if (balance + creditLimit >= amount) {

balance -= amount;

} else {

System.out.println("Insufficient credit available.");

}

}

public void addInterest() {

balance \*= (1 + interestRate \* 1.5);

}

}

public class DebitCard extends BankCard {

private int dailyLimit;

public DebitCard(double balance, double interestRate, int dailyLimit) {

super(balance, interestRate);

this.dailyLimit = dailyLimit;

}

public void setDailyLimit(int limit) {

this.dailyLimit = limit;

}

public void withdraw(double amount) {

if (balance >= amount && amount <= dailyLimit) {

balance -= amount;

} else {

System.out.println("Transaction exceeds daily limit or insufficient funds.");

}

}

public void addInterest() {

balance \*= (1 + interestRate \* 0.5);

}

}

public class Main {

public static void main(String[] args) {

CreditCard myCreditCard = new CreditCard(1000, 0.2, 500);

DebitCard myDebitCard = new DebitCard(2000, 0.1, 300);

// Deposit some funds

myCreditCard.deposit(500);

myDebitCard.deposit(1000);

// Make some transactions

myCreditCard.makePayment(750);

myDebitCard.withdraw(200);

// Add interest

myCreditCard.addInterest();

myDebitCard.addInterest();

// Print out the balances

System.out.println("Credit card balance: " + myCreditCard.balance);

System.out.println("Debit card balance: " + myDebitCard.balance);

}

}

Credit card balance: 689.04

Debit card balance: 1799.55 (As you can see, the **CreditCard** balance is lower than the initial balance due to the interest and the payment made, while the **DebitCard** balance is higher due to the interest and the smaller withdrawal made.)

package Stack;

public class Stack {

int[] stack = new int[5];

int top = -1;

public void push(int data) {

if(top == 5) {

System.out.println("stack is full!");

}else {

top++;

stack[top] = data;

}

}

public void pop() {

if(top < 0) {

System.out.println("stack is empty!");

}else {

int data = stack[top];

top--;

System.out.println(data);

}

}

public void peek() {

int data = stack[top];

System.out.println(data);

}

public void show() {

for(int i = 0; i <= top; i++) {

System.out.print(stack[i] +" ");

}

}

}

package Queue;

public class Queue {

int[] elements = new int[5];

int size = 0;

int front = 0;

int rear = 0;

public void enqueue(int data) {

if(isFull()) {

System.out.println("queue is full!");

}else {

elements[rear] = data;

rear= (rear+1)%5;

size++;

}

}

public int dequeue() {

int data = elements[front];

if(isEmpty()) {

System.out.println("queue is empty!");

}else {

front=(front+1)%5;

size--;

}

return data;

}

public boolean isEmpty() {

return getSize()==0;

}

public int getSize() {

return size;

}

public boolean isFull() {

return getSize() == 5;

}

public void show() {

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

for(int i = 0; i < size; i++) {

System.out.print(elements[(front+i)%5] + " ");

}

}

}

package linkedList;

public class LinkedList {

Node head;

class Node{

int data;

Node next;

Node(int data){

this.data = data;

next = null;

}

}

public void insertFirst(int new\_data) {

Node newNode = new Node(new\_data);

newNode.next = head;

head = newNode;

}

public void insertAfter(Node prev\_node, int new\_data){

//prev node cannot be null

Node new\_node = new Node(new\_data);

new\_node.next = prev\_node.next;

prev\_node.next = new\_node;

}

public void insertEnd(int new\_data) {

Node last = head;

Node new\_node = new Node(new\_data);

new\_node.next = null;

while(last.next != null) {

last = last.next;

}

last.next = new\_node;

return;

}

}