WEEK – 2

Exercise 4: Functions

Nipuna.A

Superset id: 6432842

Scenario 1: Calculate the age of customers for eligibility checks.

* + Question: Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

Solution:

1)CalculageAge.java

import java.time.LocalDate;

import java.time.Period;

import java.util.Scanner;

public class CalculateAge {

    public static int calculateAge(LocalDate dob) {

        LocalDate currentDate = LocalDate.now();

        if ((dob != null) && (dob.isBefore(currentDate))) {

            return Period.between(dob, currentDate).getYears();

        } else {

            return 0;

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter year of birth (e.g., 1995): ");

        int year = scanner.nextInt();

        System.out.print("Enter month of birth (1-12): ");

        int month = scanner.nextInt();

        System.out.print("Enter day of birth (1-31): ");

        int day = scanner.nextInt();

        LocalDate dob = LocalDate.of(year, month, day);

        int age = calculateAge(dob);

        if (age > 0) {

            System.out.println("Customer age is: " + age + " years");

            if (age >= 18) {

                System.out.println("Customer is eligible.");

            } else {

                System.out.println("Customer is NOT eligible.");

            }

        } else {

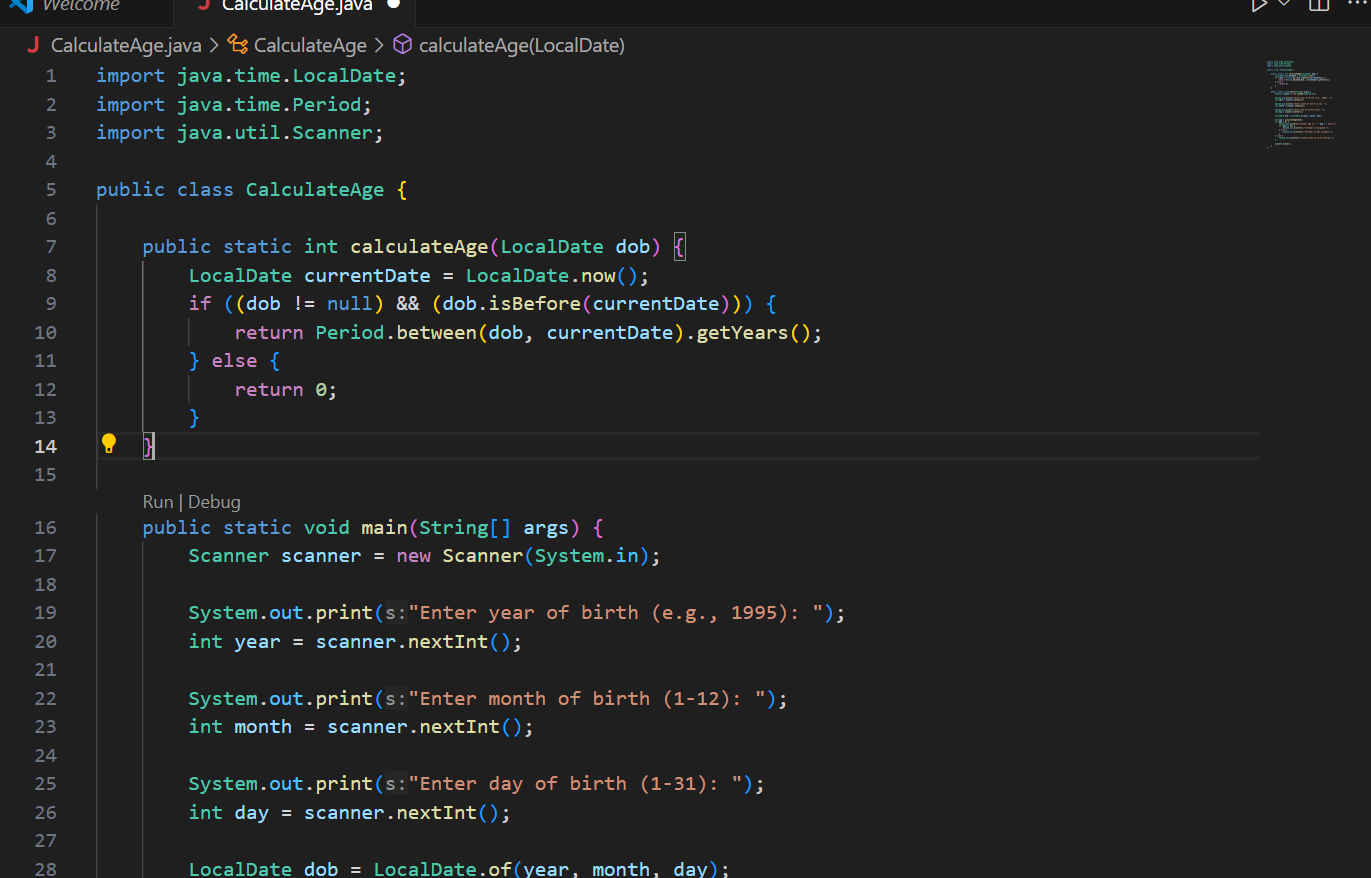
            System.out.println("Invalid date of birth entered.");

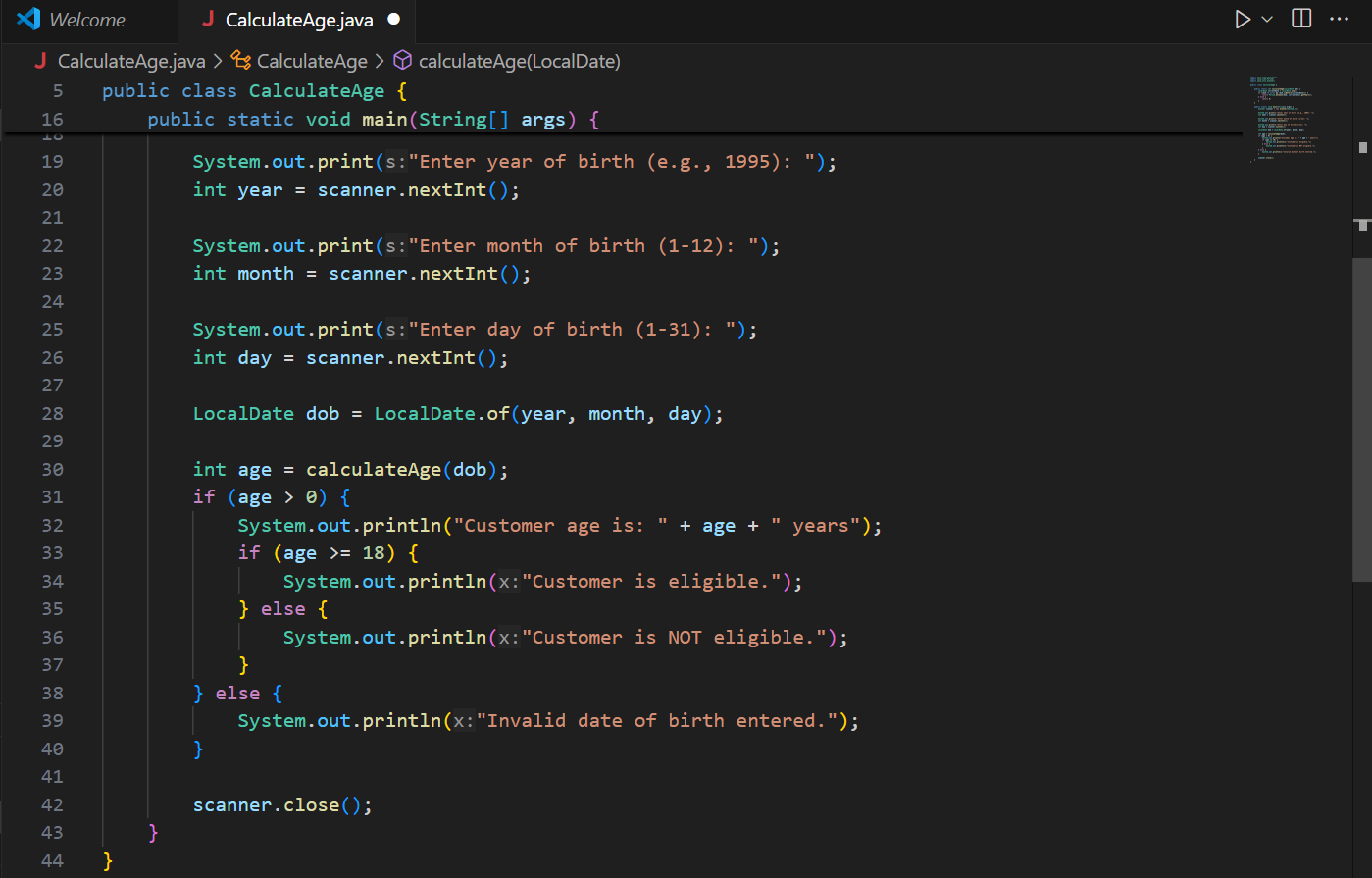
        }

        scanner.close();

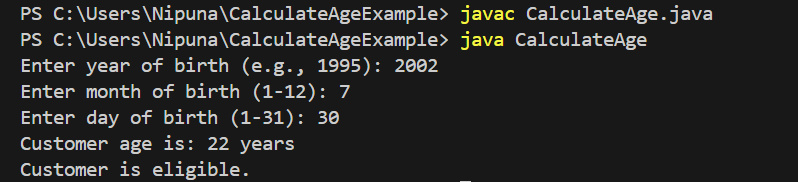
    }

}





Output:



**Scenario 2:** The bank needs to compute the monthly installment for a loan.

* + **Question:** Write a function **CalculateMonthlyInstallment** that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

Solution:

1. CalculateMonthlyInstallment.java

import java.util.Scanner;

public class CalculateMonthlyInstallment {

    public static double calculateMonthlyInstallment(double loanAmount, double annualInterestRate, int durationInYears) {

        int numberOfMonths = durationInYears \* 12;

        double monthlyInterestRate = annualInterestRate / 12 / 100;

        double emi = (loanAmount \* monthlyInterestRate \* Math.pow(1 + monthlyInterestRate, numberOfMonths))

                   / (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);

        return emi;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter loan amount: ");

        double loanAmount = scanner.nextDouble();

        System.out.print("Enter annual interest rate (in %): ");

        double annualInterestRate = scanner.nextDouble();

        System.out.print("Enter loan duration in years: ");

        int durationInYears = scanner.nextInt();

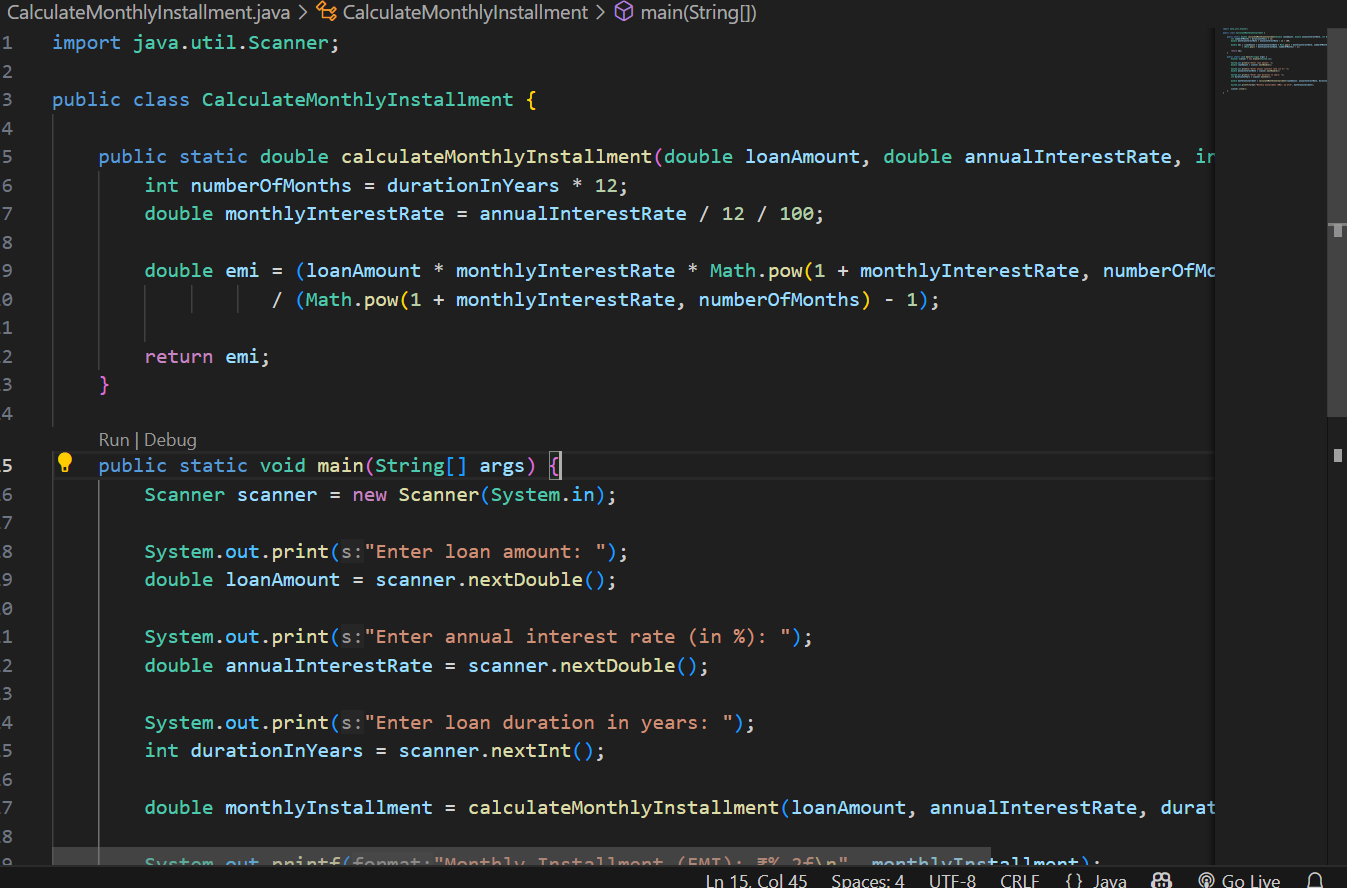
        double monthlyInstallment = calculateMonthlyInstallment(loanAmount, annualInterestRate, durationInYears);

        System.out.printf("Monthly Installment (EMI): ₹%.2f\n", monthlyInstallment);

        scanner.close();

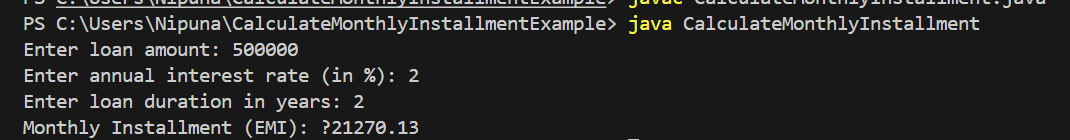
    }

}





Output:



**Scenario 3:** Check if a customer has sufficient balance before making a transaction.

* + **Question:** Write a function **HasSufficientBalance** that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

Solution:

import java.util.HashMap;

import java.util.Map;

public class HasSufficientBalanceExample {

    private static Map<Integer, Double> accountBalances = new HashMap<>();

    public static boolean HasSufficientBalance(int accountId, double amount) {

        Double balance = accountBalances.get(accountId);

        if (balance == null) {

            System.out.println("Account ID not found.");

            return false;

        }

        return balance >= amount;

    }

    public static void main(String[] args) {

        accountBalances.put(101, 5000.0);

        accountBalances.put(102, 1500.5);

        accountBalances.put(103, 250.0);

        int testAccountId = 102;

        double testAmount = 1000.0;

        boolean result = HasSufficientBalance(testAccountId, testAmount);

        System.out.println("Account " + testAccountId + " has sufficient balance for ₹" + testAmount + ": " + result);

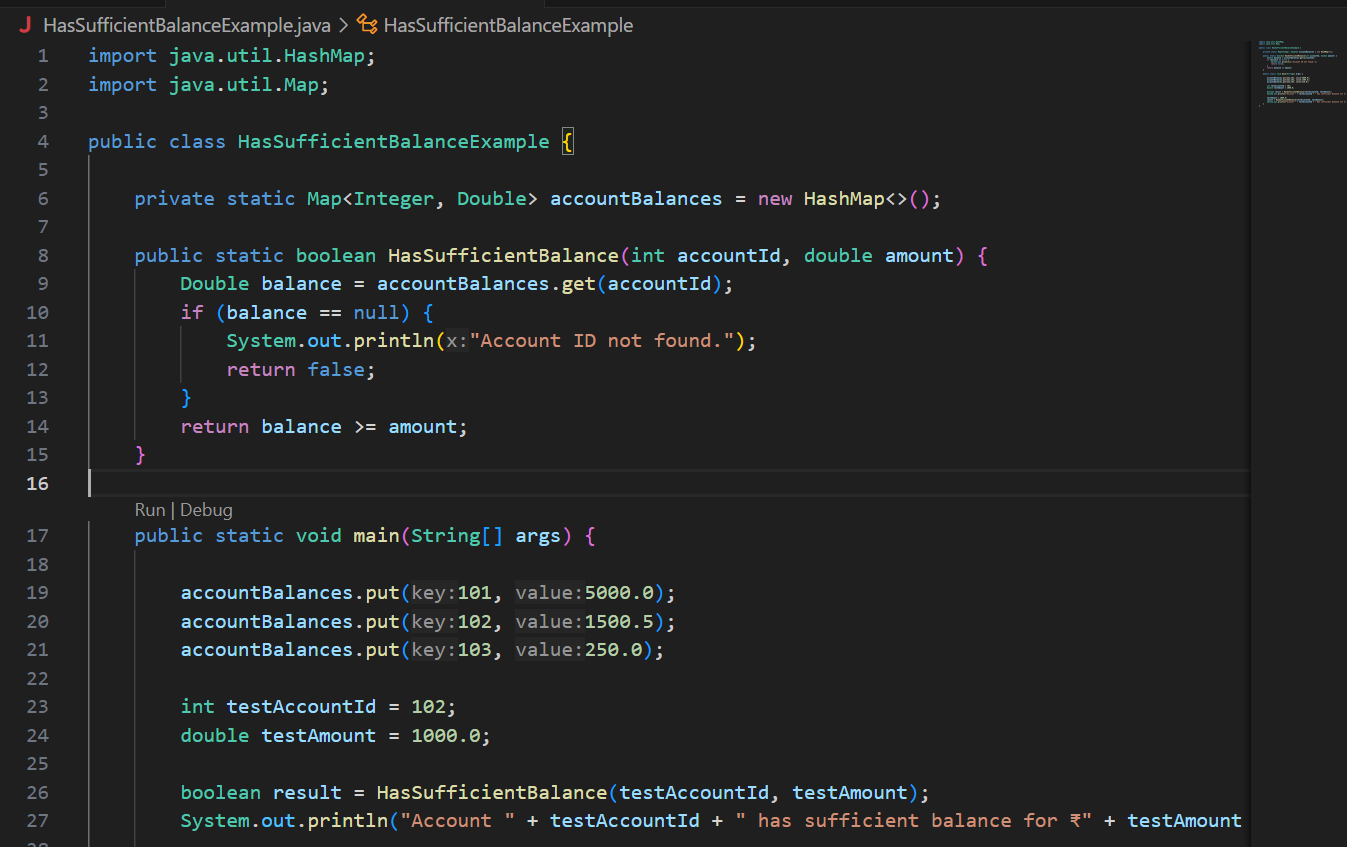
        testAmount = 2000.0;

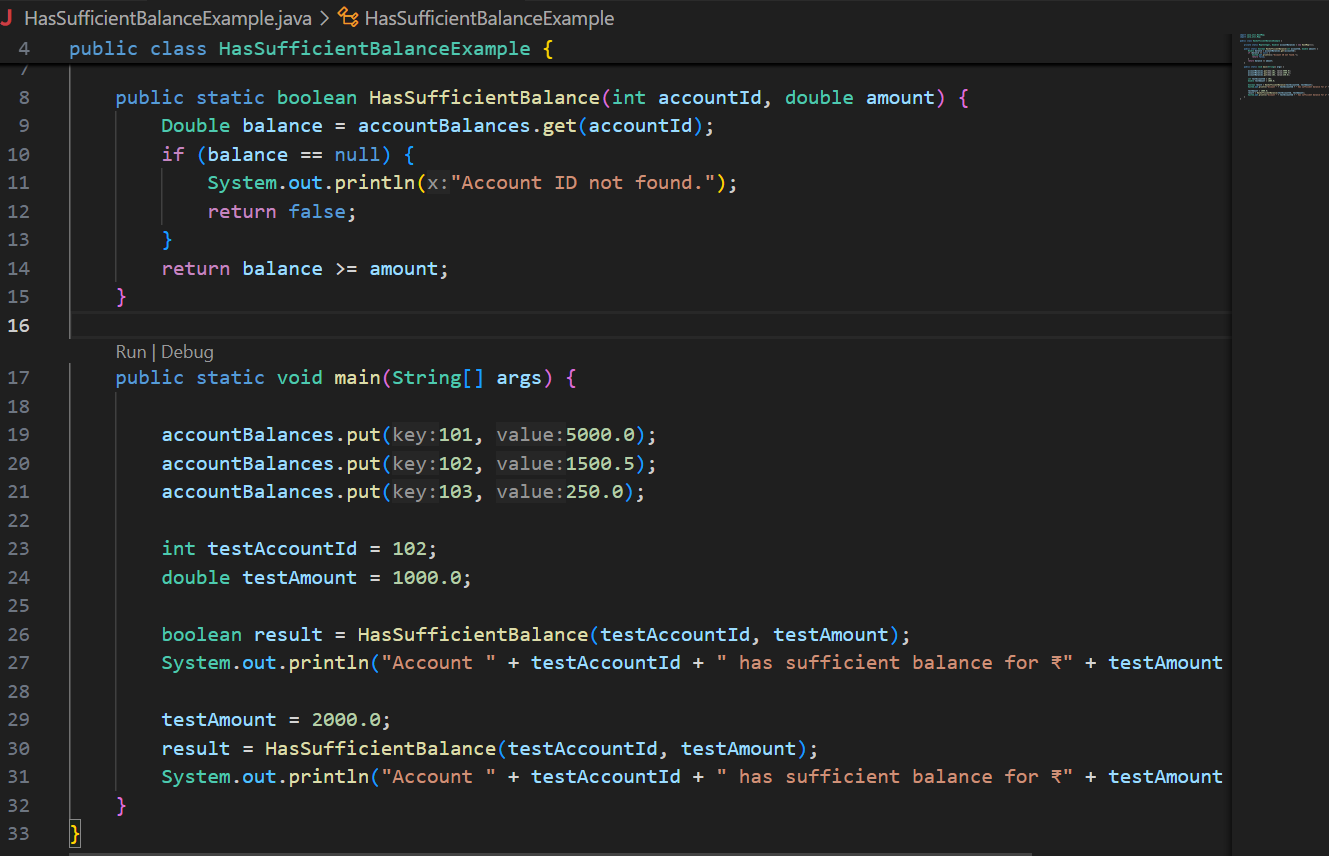
        result = HasSufficientBalance(testAccountId, testAmount);

        System.out.println("Account " + testAccountId + " has sufficient balance for ₹" + testAmount + ": " + result);

    }

}





Output:

