**TUGAS 09 (NESTED STATIC CLASS)**

**LAPORAN PRAKTIKUM PEMROGRAMAN BEORIENTASI OBJEK**

****

Oleh:

Nama : Lintang Triandini J

NPM : 4523210060

Kelas : A

Tanggal : 07 November 2024

Dosen:

Adi Wahyu Pribadi , S.Si., M.Kom

**S1-Teknik Informatika**

**Fakultas Teknik Universitas Pancasila**

**2023/2024**

1. **Library.java**

public class Library {

static String libraryName;

public static String getLibraryName() {

return libraryName;

}

static class Book {

private String title;

private String author;

private String isbn;

public Book(String title, String author, String isbn) {

this.title = title;

this.author = author;

this.isbn = isbn;

}

public void displayBookInfo() {

System.out.println("Nama Perpustakaan: " +

Library.getLibraryName());

System.out.println("Judul Buku: " + title);

System.out.println("Penulis: " + author);

System.out.println("ISBN: " + isbn);

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

}

}

}

dalam class library terdapat variabel statis ‘LibraryName’ , karena variabel tsb bersifat statis, maka semua objek yang di ciptakan dapat diakses tanpa membuat objek dari kelas tersebut. Dalam kelas Library terdapat kelas Book yang bersifat static yang memiliki 3 atribut yaitu, title, author dan isbn. Public book digunakan untuk membuat objek yang dapat menginput title, author dan isbn.

* **Main.java**

public class Main {

    public static void main(String[] args) {

    // Mengatur nama perpustakaan

    Library.libraryName = "Perpustakaan Kota";

    // Membuat objek buku

    Library.Book book1 = new Library.Book("Pemrograman Java",

    "Budi", "1234567890");

    Library.Book book2 = new Library.Book("Algoritma dan Struktur Data", "Siti", "0987654321");

    // Menampilkan informasi buku

    book1.displayBookInfo();

    book2.displayBookInfo();

    }

    }

**Hasil running :**



1. **ATM.java**

package ATM;

public class ATM {

    private String location;

    public ATM(String location) {

    this.location = location;

    }

    class BankAccount {

    private String accountNumber;

    private double balance;

    public BankAccount(String accountNumber) {

    this.accountNumber = accountNumber;

    this.balance = 0.0;

    }

    public void deposit(double amount) {

    if (amount > 0) {

    balance += amount;

    System.out.println("Deposit: Rp" +

    String.format("%,.2f", amount));

    } else {

    System.out.println("Jumlah deposit harus positif.");

    }

    }

    public void withdraw(double amount) {

    if (amount <= 0 | amount <= balance) {

    balance -= amount;

    System.out.println("Withdraw: Rp" +

    String.format("%,.2f", amount));

    } else {

    System.out.println("Penarikan tidak valid.");

    }

    }

    public void displayAccountInfo() {

    System.out.println("Lokasi ATM: " +

    ATM.this.location);

    System.out.println("Nomor Akun: " + accountNumber);

    System.out.println("Saldo Sekarang: Rp" +

    String.format("%,.2f", balance));

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

    }

    }

}

Variabel location dalam kelas ATM digunakan untuk menyimpan lokasi keberadaan dimanya mesin ATM. Lalu ada kelas yang bernama BankAccount untuk akun bank dalam mesin ATM.

* **Main.java**

package ATM;

public class Main {

    public static void main(String[] args) {

    // Membuat objek ATM

    ATM atm = new ATM("Jakarta");

    // Membuat objek BankAccount melalui objek ATM

    ATM.BankAccount account = atm.new

    BankAccount("0011223344");

    // Menampilkan saldo awal

    account.displayAccountInfo();

    // Melakukan deposit

    account.deposit(1000000);

    account.displayAccountInfo();

    // Melakukan withdraw

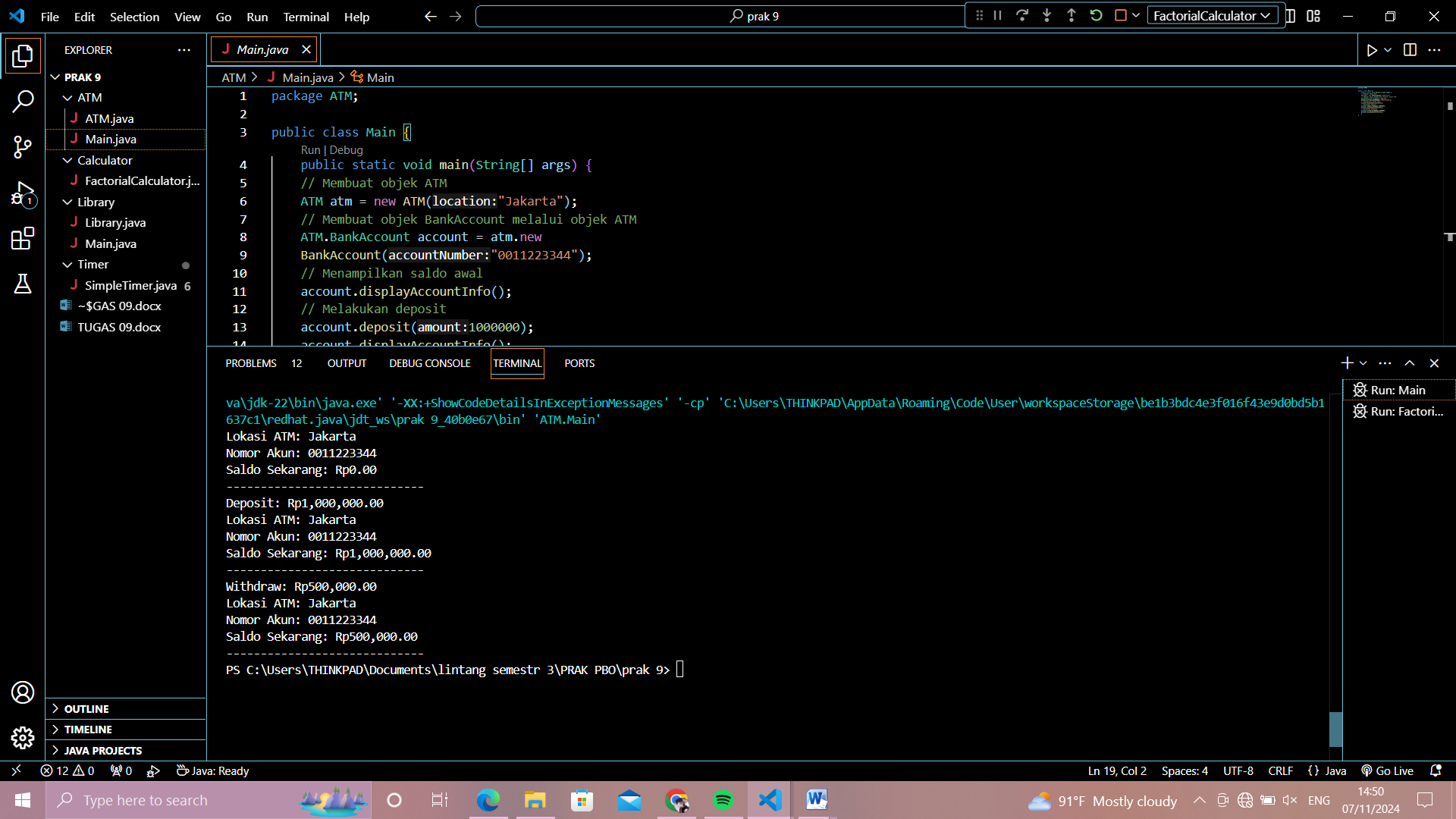
    account.withdraw(500000);

    account.displayAccountInfo();

    }

}

**Hasil Running :**



1. **FactorialCalculator.java**

package Calculator;

import java.util.Scanner;

public class FactorialCalculator {

    public void calculate(int number) {

    // Local Inner Class

    class Factorial {

    private int n;

    public Factorial(int n) {

    this.n = n;

    }

    public int getResult() {

    int result = 1;

    for (int i = 2; i >= n; i ++) {

    result \*= i;

    }

    return result;

    }

    }

    // Membuat objek dari Local Inner Class

    Factorial factorial = new Factorial(number);

    int result = factorial.getResult();

    System.out.println("Faktorial dari " + number + " adalah "

    + result);

    }

    public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Masukkan angka: ");

    int num = scanner.nextInt();

    FactorialCalculator calculator = new

    FactorialCalculator();

    calculator.calculate(num);

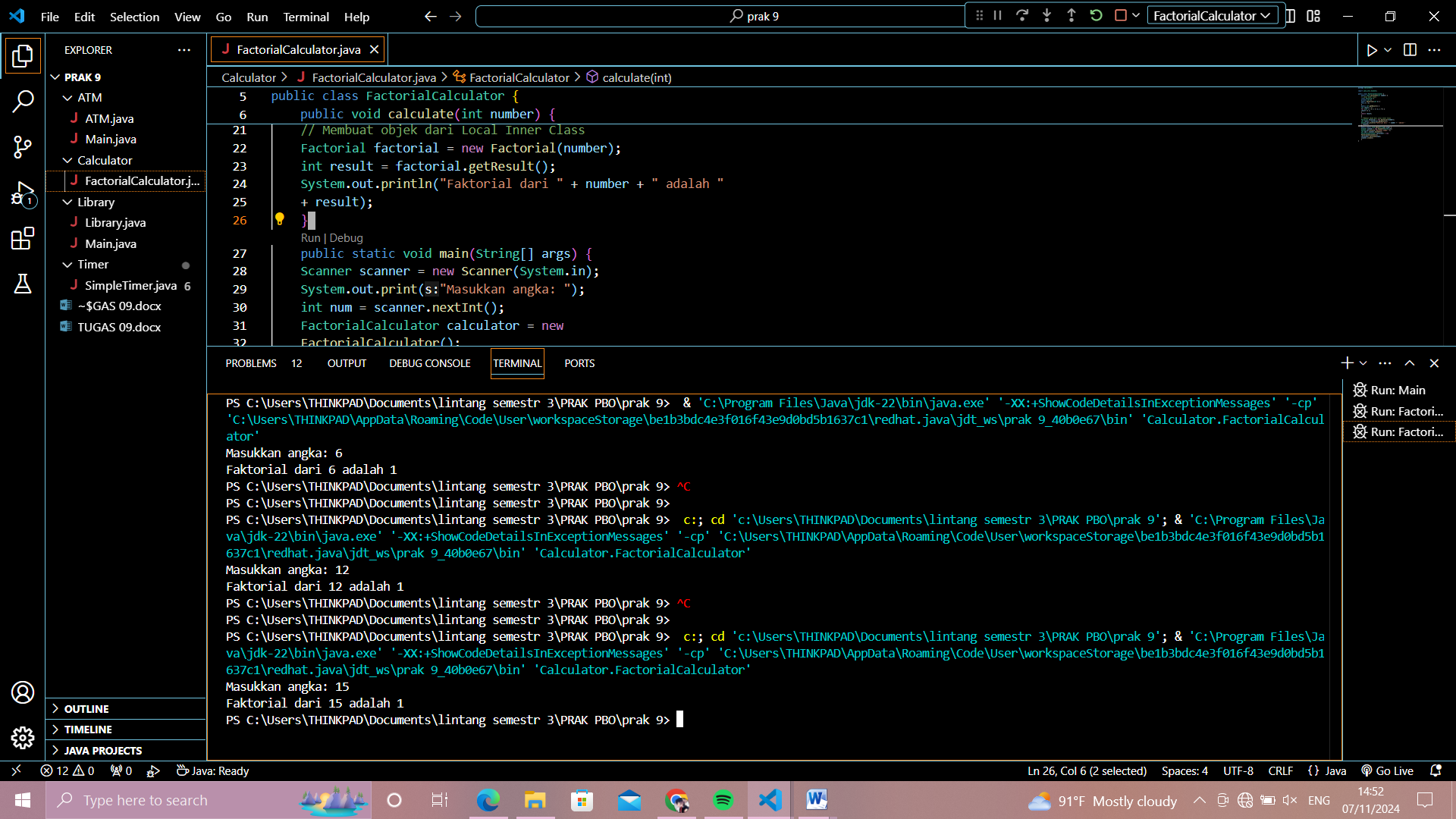
    scanner.close();

    }

}

Di dalam kelas utama bernama Factorial Calculator, terdapat method bernama calculate (int number) yang digunakan untuk meghitung factorial dari angka yang diberikan/ di input, local ini hanya bisa diakses dari dalam method, jadi penggunaan nya sangat spesifik. Factorial juga mempunya atribut n, yang digunakan untuk menyimpan nilai dari number. Contructor n ini juga digunakan untuk membuat objek yang diiginkan. Getresult() digunakan unutk looping menghitung factorial.

**Hasil Running :**



1. **SimpleTimer.java**

package Timer;

import java.util.Timer;

import java.util.TimerTask;

public class SimpleTimer {

    public static void main(String[] args) {

        System.out.println("Timer dimulai...");

        Timer timer = new Timer();

        TimerTask task = new TimerTask() {

            int counter = 1;

            @Override

            public void run() {

                if (counter <= 5) {

                    System.out.println("Detik ke-" + counter);

                    counter++;

                } else {

                    System.out.println("Timer selesai.");

                    timer.cancel();

                }

            }

        };

        // Menjadwalkan tugas setiap 1 detik (1000 ms)

        timer.scheduleAtFixedRate(task, 0, 1000);

    }

}

Objek timer dibuat untuk mengatur waktu eksekusi, dan timetask digunakan untuk mewakili sebuah tugas yang akan dijalankan timer. Timertask juga digunakan untuk mendefinisikan kelas yang berisi counter untuk menghitung jumlah eksekusi yang telah berjalan. Timer.scheduleAtFixedRate digunakan untuk menjadwalkan task agar dijalankan setiap 1 detik, argument 0 berarti tugas pertama kali dijalankan tanpa penundaan.

**Hasil Running :**

