LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 3

Oleh :

Chris Hotasi Rajagukguk

2311103126

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS REKAYASA INDUSTRI

UNIVERSITAS TELKOM PURWOKERTO

2024

A. Studi Kasus

Sistem Informasi Akademik Universitas

B. Deskripsi

Universitas ingin mengembangkan sistem informasi akademik yang memungkinkan pengelolaan data akademik mahasiswa, dosen, matakuliah, dan pengelolaan KRS (Kartu Rencana Studi) mahasiswa. Fitur utama dari sistem ini adalah sebagai berikut:

1. Manajemen Mahasiswa: Setiap mahasiswa memiliki informasi dasar seperti nama, NIM, prodi, dan daftar KRS yang sudah diambil.
2. Manajemen Dosen: Setiap dosen memiliki nama, NIP, dan daftar matakuliah yang diajarkan.
3. Manajemen Matakuliah: Setiap matakuliah memiliki kode, nama, dan jumlah SKS.
4. Pengelolaan KRS (Kartu Rencana Studi): Mahasiswa dapat memilih dan menghapus matakuliah yang ingin mereka ambil setiap semester.

C. Guided

1. Buat project baru dengan LatihanP7**Nama**. Nama diganti dengan nama kalian, contoh : *LatihanP7SenaWijayanto*
2. Buatlah program struktur program dasar dengan kelas-kelas berikut:
   - **Person**: Kelas abstrak yang menjadi *superclass* bagi **Mahasiswa** dan **Dosen**.
   - **Mahasiswa**: Kelas turunan dari **Person** yang memiliki informasi KRS.
   - **Dosen**: Kelas turunan dari **Person** yang mengelola daftar matakuliah yang diajarkan.
   - **Matakuliah**: Kelas yang berisi informasi matakuliah, dengan metode **aturJadwal** yang di-overload.
   - **KRS**: Interface yang mengatur pengambilan dan penghapusan matakuliah oleh mahasiswa.

Kode

Person.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package latihanp7chrishotasirajagukguk;
```

```
/**
 *
 * @author Chris Hotasi Rajagukguk
 */
abstract class Person {
    protected String nama;
    protected String id;

    public Person(String nama, String id) {
        this.nama = nama;
        this.id = id;
    }

    // Abstract Method (Implementasi khusus di subclass)
    public abstract void showInfo();
}
```

Penjelasan : Kelas `Person` adalah kelas abstrak yang dirancang untuk menjadi dasar bagi kelas-kelas lain yang mewakili individu. Kelas ini memiliki dua atribut, yaitu `nama` dan `id`, yang diinisialisasi melalui konstruktor. Sebagai kelas abstrak, `Person` tidak dapat diinstansiasi secara langsung dan memiliki metode abstrak `showInfo()`, yang harus diimplementasikan oleh subclass-nya. Dengan cara ini, kelas-kelas turunan dapat memberikan implementasi spesifik untuk menampilkan informasi tentang individu, sementara tetap mempertahankan struktur umum yang ditetapkan oleh kelas `Person`. Desain ini mendukung konsep pemrograman berorientasi objek, seperti pewarisan dan polymorphism, memungkinkan pengelompokan dan pengelolaan data yang lebih baik dalam aplikasi.

MataKuliah.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
```

```
package latihanp7chrishotasirajagukguk;

/**
 *
 * @author Chris Hotasi Rajagukguk
 */
public class MataKuliah {
    private String kode;
    private String namaMatakuliah;
    private int sks;
    private String jadwalHari;
    private String jadwalJam;

    public MataKuliah(String kode, String namaMatakuliah, int sks) {
        this.kode = kode;
        this.namaMatakuliah = namaMatakuliah;
        this.sks = sks;
    }

    // Overloading: Metode aturJadwal dengan dua versi
    public void aturJadwal(String hari, String jam) {
        this.jadwalHari = hari;
        this.jadwalJam = jam;
    }

    public void aturJadwal(String hari) {
        this.jadwalHari = hari;
    }

    public void showInfo() {
```

```
        System.out.println("Kode MK: " + kode + ", Nama: " + namaMatakuliah + ", SKS: " + sks);
    }
}
```

Penjelasan : Kelas `MataKuliah` memiliki atribut privat, termasuk `kode`, `namaMatakuliah`, `sks`, `jadwalHari`, dan `jadwalJam`, yang menyimpan informasi penting tentang mata kuliah. Konstruktor `MataKuliah` digunakan untuk menginisialisasi `kode`, `namaMatakuliah`, dan `sks`, sementara jadwal hari dan jam dapat diatur setelah objek dibuat. Kelas ini juga mengimplementasikan metode overloading pada metode `aturJadwal()`, yang memungkinkan penetapan jadwal dengan dua cara: satu dengan menentukan hari dan jam, dan yang lain hanya dengan hari saja. Hal ini memberikan fleksibilitas dalam penggunaan kelas. Metode `showInfo()` digunakan untuk menampilkan informasi tentang mata kuliah dalam format yang jelas, mencakup kode, nama, dan jumlah SKS. Dengan desain ini, kelas `MataKuliah` tidak hanya menyimpan data tetapi juga menyediakan cara untuk mengelolanya, mendukung prinsip pemrograman berorientasi objek seperti enkapsulasi dan overloading metode.

KRS.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package latihanp7chrishotasirajagukguk;


/**
 *
 * @author Chris Hotasi Rajagukguk
 */
interface KRS {
    void tambahMatakuliah(MataKuliah mk);
    void hapusMatakuliah(MataKuliah mk);
}
```

Penjelasan : Interface `KRS` memiliki dua metode abstrak: `tambahMatakuliah(MataKuliah mk)`**: Menambahkan mata kuliah ke rencana studi. Dan `hapusMatakuliah(MataKuliah

mk)`**: Menghapus mata kuliah dari rencana studi. Interface ini memungkinkan kelas yang mengimplementasikannya untuk mengelola daftar mata kuliah mahasiswa, mendukung prinsip pemrograman berorientasi objek seperti fleksibilitas dan pemisahan antarmuka dari implementasi.

Dosen.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package latihanp7chrishotasirajagukguk;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Chris Hotasi Rajagukguk
 */
class Dosen extends Person {
    private String nip;
    private List<MataKuliah> daftarMatakuliah = new ArrayList<>();

    public Dosen(String nama, String id, String nip) {
        super(nama, id);
        this.nip = nip;
    }

    // Overriding showInfo (Implementasi khusus untuk Dosen)
```

```
    @Override
    public void showInfo() {
        System.out.println("Nama Dosen: " + nama + ", NIP: " + nip);
    }

    // Tambah Matakuliah untuk Dosen
    public void tambahMatakuliah(MataKuliah mk) {
        daftarMatakuliah.add(mk);
    }
}
```

Penjelasan : Kelas `Dosen` adalah subclass dari kelas abstrak `Person`, Kelas ini memiliki atribut tambahan berupa `nip` (Nomor Induk Pegawai) dan daftar mata kuliah yang diajarkan, yang disimpan dalam `daftarMatakuliah`, berupa list dari objek `MataKuliah`. Konstruktor `Dosen` menginisialisasi nama, ID, dan NIP menggunakan konstruktor dari kelas induk `Person`. Metode `showInfo()` dioverride untuk memberikan implementasi khusus yang menampilkan informasi dosen, termasuk nama dan NIP. Selain itu, kelas ini juga menyediakan metode `tambahMatakuliah(MataKuliah mk)`, yang memungkinkan dosen untuk menambahkan mata kuliah ke dalam daftar yang diajarkan. Dengan demikian, kelas `Dosen` menggabungkan atribut dan perilaku yang spesifik untuk dosen, mendukung prinsip pewarisan dan enkapsulasi dalam pemrograman berorientasi objek.

Mahasiswa.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package latihanp7chrishotasirajagukguk;

import java.util.ArrayList;
import java.util.List;
```

```
/**
 *
 * @author Chris Hotasi Rajagukguk
 */
class Mahasiswa extends Person implements KRS {
    private String nim;
    private String prodi;
    private List<MataKuliah> daftarKRS = new ArrayList<>();

    public Mahasiswa(String nama, String id, String nim, String prodi) {
        super(nama, id);
        this.nim = nim;
        this.prodi = prodi;
    }

    // Overriding showInfo (Implementasi khusus untuk Mahasiswa)
    @Override
    public void showInfo() {
        System.out.println("Nama Mahasiswa: " + nama + ", NIM: " + nim + ", Prodi: " + prodi);
    }

    // Implementasi metode interface untuk tambah dan hapus matakuliah
    @Override
    public void tambahMatakuliah(MataKuliah mk) {
        daftarKRS.add(mk);
    }

    @Override
```

```
    public void hapusMatakuliah(MataKuliah mk) {
        daftarKRS.remove(mk);
    }
}
```

Penjelasan : Kelas `Mahasiswa` adalah subclass dari kelas abstrak `Person` dan juga mengimplementasikan interface `KRS`, Kelas ini memiliki atribut tambahan seperti `nim` (Nomor Induk Mahasiswa) dan `prodi` (program studi), serta daftar mata kuliah yang diambil, disimpan dalam `daftarKRS` berupa list dari objek `MataKuliah`. Konstruktor `Mahasiswa` menginisialisasi nama, ID, NIM, dan prodi dengan memanggil konstruktor kelas induk `Person`. Metode `showInfo()` dioverride untuk memberikan implementasi khusus yang menampilkan informasi mahasiswa, termasuk nama, NIM, dan prodi. Dalam konteks interface `KRS`, kelas ini mengimplementasikan dua metode `tambahMatakuliah(MataKuliah mk)`: Menambahkan mata kuliah ke daftar KRS dan `hapusMatakuliah(MataKuliah mk)`: Menghapus mata kuliah dari daftar KRS.

Main.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package latihanp7chrishotasirajagukguk;


/**
 *
 * @author Chris Hotasi Rajagukguk
 */
public class LatihanP7ChrisHotasiRajagukguk {


    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String[] args) {
        // Membuat Objek Mahasiswa dan Dosen
        Mahasiswa mhs = new Mahasiswa("Chris", "123", "2311103126", "Sistem Informasi");
        Dosen dosen = new Dosen("Sena Wijayanto", "456", "D001");

        // Membuat Objek Matakuliah
        MataKuliah mk1 = new MataKuliah("SI101", "PBO", 3);
        MataKuliah mk2 = new MataKuliah("SI102", "Basis Data", 3);

        // Mengatur Jadwal Matakuliah (Overloading)
        mk1.aturJadwal("Senin", "08:00");
        mk2.aturJadwal("Selasa");

        // Menambahkan Matakuliah ke KRS Mahasiswa
        mhs.tambahMatakuliah(mk1);
        mhs.tambahMatakuliah(mk2);

        // Menampilkan Informasi
        mhs.showInfo(); // Menampilkan informasi Mahasiswa (Polimorfisme)
        dosen.showInfo(); // Menampilkan informasi Dosen (Polimorfisme)
        mk1.showInfo(); // Informasi Matakuliah 1
        mk2.showInfo(); // Informasi Matakuliah 2
    }
}
```

Penjelasan : Kelas `LatihanP7ChrisHotasiRajagukguk` adalah kelas utama yang menjalankan program manajemen pendidikan. Dalam metode `main()`, objek `Mahasiswa` dan `Dosen` dibuat dengan informasi identitas masing-masing. Dua objek `MataKuliah` untuk "PBO" dan "Basis Data" juga dibuat. Jadwal mata kuliah diatur menggunakan metode overloading `aturJadwal()`, dengan satu mata kuliah diatur lengkap dengan hari dan jam, sedangkan yang lain hanya dengan hari. Mata kuliah kemudian ditambahkan ke daftar KRS mahasiswa menggunakan metode

`tambahMatakuliah()`. Akhirnya, informasi tentang mahasiswa, dosen, dan kedua mata kuliah ditampilkan menggunakan metode `showInfo()`,

Output

```
Nama Mahasiswa: Chris, NIM: 2311103126, Prodi: Sistem Informasi
Nama Dosen: Sena Wijayanto, NIP: D001
Kode MK: SI101, Nama: PBO, SKS: 3
Kode MK: SI102, Nama: Basis Data, SKS: 3
```

D. Unguided

Kembangkan Sistem Informasi Akademik dengan menambahkan fitur berikut:

1. Menampilkan Daftar Mata Kuliah yang Diambil oleh Mahasiswa. Tambahkan metode **showKRS()** dalam kelas **Mahasiswa** yang menampilkan seluruh mata kuliah yang ada dalam daftar KRS mahasiswa tersebut.

   Mahasiswa.java

```
public void showKRS() {
      System.out.println("Daftar KRS Mahasiswa : ");
      if (daftarKRS.isEmpty()) {
         System.out.println("Tidak ada mata kuliah yang diambil");
      } else {
         for (MataKuliah mk : daftarKRS) {
            mk.showInfo();
         }
      }
   }
```

   Penjelasan : Metode `showKRS()` mencetak judul yang menunjukkan nama mahasiswa. Kemudian, metode ini memeriksa apakah daftar mata kuliah (`daftarKRS`) kosong. Jika kosong, akan dicetak pesan bahwa mahasiswa belum mendaftar untuk mata kuliah. Jika tidak, metode ini menggunakan looping untuk iterasi melalui setiap objek `MataKuliah` dalam daftar dan memanggil `showInfo()` untuk menampilkan informasi lengkap tentang setiap mata kuliah, seperti kode, nama, dan jumlah SKS. Dengan demikian, `showKRS()` memberikan gambaran jelas tentang mata kuliah yang diambil mahasiswa dan status akademik mereka.

   Main.java

```
mhs.showKRS(); // Menampilkan daftar mata kuliah
```

Penjelasan : Dalam file main cukup menambahkan metode showKRS() agar metode nya dapat berfungsi

Output

```
Nama Mahasiswa: Chris, NIM: 2311103126, Prodi: Sistem Informasi
Nama Dosen: Sena Wijayanto, NIP: D001
Daftar KRS Mahasiswa :
Kode MK: SI101, Nama: PBO, SKS: 3
Kode MK: SI102, Nama: Basis Data, SKS: 3
```

2. Menghitung Total SKS yang Diambil oleh Mahasiswa. Tambahkan metode **hitungTotalSKS()** dalam kelas **Mahasiswa** untuk menghitung total SKS dari seluruh mata kuliah yang telah diambil

Mahasiswa.java

```
public int hitungTotalSKS() {
        int totalSks = 0;
        for (MataKuliah mk : daftarKRS) {
            totalSks += mk.getSks();
        }
        return totalSks;
    }

    public void showKRS() {
        System.out.println("Daftar KRS Mahasiswa");
        if (daftarKRS.isEmpty()) {
            System.out.println("Tidak ada mata kuliah yang diambil.");
        } else {
            for (MataKuliah mk : daftarKRS) {
                mk.showInfo();
            }
            System.out.println("Total SKS: " + hitungTotalSKS());
        }
    }
```

Penjelasan : Metode `hitungTotalSKS()` berfungsi untuk menghitung total SKS dari mata kuliah yang diambil oleh mahasiswa. Proses dimulai dengan menginisialisasi variabel `totalSks` dengan nilai 0. Metode ini kemudian menggunakan loop for-each untuk iterasi melalui objek `MataKuliah` dalam `daftarKRS`, memanggil `getSks()` pada setiap objek untuk mendapatkan jumlah SKS, dan menambahkannya ke `totalSks`. Setelah loop, metode ini mengembalikan nilai `totalSks`, yang merupakan total SKS yang diambil mahasiswa. Di

dalam metode `showKRS()`, `hitungTotalSKS()` dipanggil untuk menampilkan jumlah SKS yang ada.

MataKuliah.java

```
public int getSks() {
        return sks;
    }
```

Penjelasan : Metode `getSks()` adalah getter yang memberikan akses ke nilai atribut `sks` dalam kelas `MataKuliah`. Atribut `sks` dideklarasikan sebagai privat, sehingga tidak dapat diakses langsung dari luar kelas. Dengan adanya metode ini, pengguna dapat memperoleh nilai SKS tanpa mengubah atau mengakses atribut secara langsung.

Output

```
Nama Mahasiswa: Chris, NIM: 2311103126, Prodi: Sistem Informasi
Nama Dosen: Sena Wijayanto, NIP: D001
Daftar KRS Mahasiswa
Kode MK: SI101, Nama: PBO, SKS: 3
Kode MK: SI102, Nama: Basis Data, SKS: 3
Total SKS: 6
```