M. Hasan Basri
2241760139
SIB2C

# Jobsheet 04 - Relasi Kelas

## I. Kompetensi

Setelah menempuh pokok bahasan ini, mahasiswa mampu:

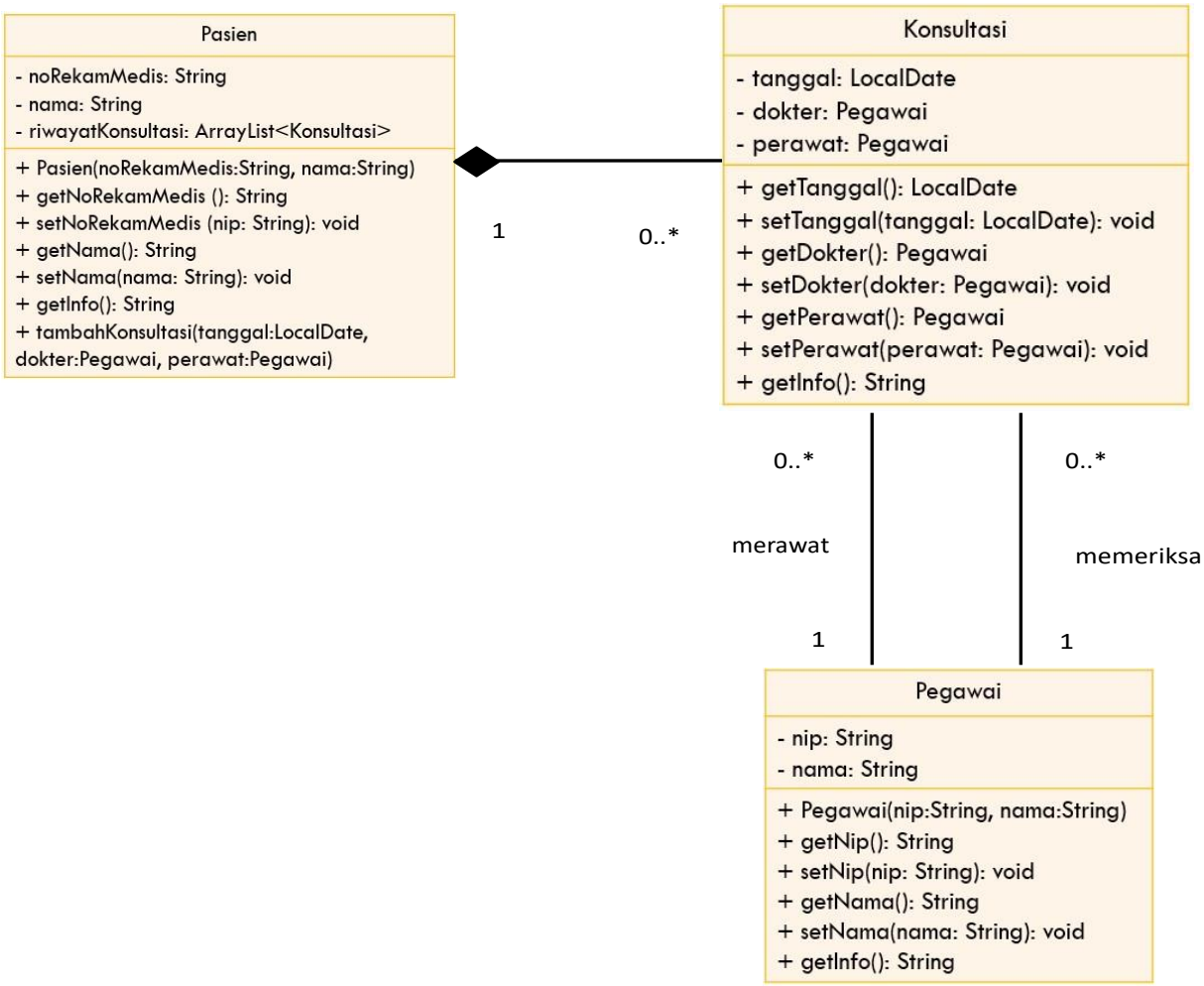
1. Memahami konsep relasi kelas;
2. Mengimplementasikan relasi asosiasi ke dalam program.

## II. Pendahuluan

Pada kasus yang lebih kompleks, dalam suatu sistem akan ditemukan lebih dari satu *class* yang saling memiliki keterkaitan antara *class* satu dengan yang lain. Pada percobaan-percobaan sebelumnya, mayoritas kasus yang sudah dikerjakan hanya fokus pada satu *class* saja. Pada jobsheet ini akan dilakukan percobaan yang melibatkan beberapa *class* yang saling berkaitan.

## III. Praktikum

Pada praktikum ini akan dikembangkan suatu sistem informasi rumah sakit yang menyimpan data riwayat konsultasi pasien.
Perhatikan diagram *class* berikut:




a. Buatlah folder baru dengan nama RumahSakit

b. Buatlah class Pegawai. Tambahkan atribut nip dan nama pada class Pegawai dengan akses modifier private

```
public class Pegawai {
    private String nip;
    private String nama;
}
```

c. Buatlah *constructor* untuk class Pegawai dengan parameter nip dan nama.

```
public Pegawai(String nip, String nama) {
    this.nip = nip;
    this.nama = nama;
}
```

d. Implementasikan **setter** dan **getter** untuk class Pegawai.

```
public String getNip() {
    return nip;
}

public void setNip(String nip) {
    this.nip = nip;
}

public String getNama() {
    return nama;
}

public void setNama(String nama) {
    this.nama = nama;
}
```

e. Implementasikan *method* getInfo()sebagai berikut:

```
public String getInfo(){
    return nama + " (" + nip + ")";
}
```

f. Selanjutnya buatlah class Pasien kemudian tambahkan atribut noRekamMedis dan nama pada class Pasien dengan access level modifier private. Sediakan pula setter dan getter untuk kedua atribut tersebut.

```
public class Pasien {
    private String noRekamMedis;
    private String nama;

    public String getNoRekamMedis() {
        return noRekamMedis;
    }

    public void setNoRekamMedis(String noRekamMedis) {
        this.noRekamMedis = noRekamMedis;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }
}
```

g. Buatlah constructor untuk class Pegawai dengan parameter noRekamMedis dan nama

```
public Pasien(String noRekamMedis, String nama) {
    this.noRekamMedis = noRekamMedis;
    this.nama = nama;
}
```

h. Implementasikan *method* getInfo()sebagai berikut:

```
public String getInfo(){
    String info = "";
    info += "No Rekam Medis      : " + this.noRekamMedis + "\n";
    info += "Nama                : " + this.nama + "\n";
    info += "\n";

    return info;
}
```

i. Sistem ini akan menyimpan data setiap konsultasi yang dilakukan pasien. Pasien bisa melakukan konsultasi lebih dari sekali. Oleh karena itu, data konsultasi akan disimpan dalam bentuk ArrayList dari objek-objek yang bertipe Konsultasi.

j. Buatlah class dengan nama Konsultasi dengan atribut tanggal bertipe LocalDate, dokter bertipe Pegawai, dan perawat bertipe Pegawai. Set access level modifier private untuk seluruh atribut. Lakukan import `java.time.LocalDate` agar dapat mendeklarasikan atribut tanggal bertipe LocalDate.

```
import java.time.LocalDate;

public class Konsultasi {
    private LocalDate tanggal;
    private Pegawai dokter;
    private Pegawai perawat;
}
```

k. Sediakan setter dan getter untuk masing-masing atribut pada class Konsultasi

```
public LocalDate getTanggal() {
    return tanggal;
}

public void setTanggal(LocalDate tanggal)
    this.tanggal = tanggal;
}

public Pegawai getDokter() {
    return dokter;
}

public void setDokter(Pegawai dokter) {
    this.dokter = dokter;
}

public Pegawai getPerawat() {
    return perawat;
}

public void setPerawat(Pegawai perawat) {
    this.perawat = perawat;
}
```

l. Implementasikan method getInfo() sebagai berikut:

```
public String getInfo(){
    String info = "";
    info += "\tTanggal: " + tanggal;
    info += ", Dokter: " +  dokter.getInfo();
    info += ", Perawat: " + perawat.getInfo();
    info += "\n";

    return info;
}
```

m. Untuk menyimpan data riwayat konsultasi pasien, maka tambahkan atribut riwayatKonsultasi pada class Pasien dengan tipe arrayList<Konsultasi>. Atribut ini

akan menyimpan serangkaian objek bertipe Konsultasi. Import java.util.ArrayList agar dapat mendeklarasikan atribut bertipe ArrayList of object.

```
private String noRekamMedis;
private String nama;
private ArrayList<Konsultasi> riwayatKonsultasi;
```

n. Buatlah constructor berparameter untuk class Pasien. Inisiasi nilai atribut noRekamMedis dan nama berdasarkan atribut nama. Instansiasi/buat ArrayList baru untuk atribut riwayatKonsultasi;

```
public Pasien(String noRekamMedis, String nama) {
    this.noRekamMedis = noRekamMedis;
    this.nama = nama;
    this.riwayatKonsultasi = new ArrayList<Konsultasi>();
}
```

o. Lakukan import java.time.LocalDate agar dapat mendeklarasikan atribut tanggal bertipe LocalDate pada class Pasien. Selanjutnya, implementasikan method tambahKonsultasi() sebagai berikut:

```
public void tambahKonsultasi(LocalDate tanggal, Pegawai dokter, Pegawai perawat){
    Konsultasi konsultasi = new Konsultasi();
    konsultasi.setTanggal(tanggal);
    konsultasi.setDokter(dokter);
    konsultasi.setPerawat(perawat);
    riwayatKonsultasi.add(konsultasi);
}
```

p. Modifikasi method getInfo() untuk mengembalikan info pasien dan daftar konsultasi yang pernah dilakukan

```
public String getInfo(){
    String info = "";
    info += "No Rekam Medis      : " + this.noRekamMedis + "\n";
    info += "Nama                : " + this.nama + "\n";

    if (!riwayatKonsultasi.isEmpty()) {
        info += "Riwayat Konsultasi :\n";

        for (Konsultasi konsultasi : riwayatKonsultasi) {
            info += konsultasi.getInfo();
        }
    }
    else{
        info += "Belum ada riwayat konsultasi";
    }

    info += "\n";

    return info;
}
```

q. Lakukan import java.time.LocalDate agar dapat mendeklarasikan atribut tanggal bertipe LocalDate pada class RumahSakitDemo. Test program yang sudah dibuat dengan membuat objek-objek pada class RumahSakitDemo. Instansiasi objek baru bertipe Pegawai dengan nama ani menggunakan constructor Pegawai(String nip, String nama) dengan nilai argumen nip “1234” dan nama “dr. Ani”. Lanjutkan instansiasi objek sebagai berikut:

```java
import java.time.LocalDate;

public class RumahSakitDemo {

    public static void main(String[] args) {
        Pegawai ani = new Pegawai("1234", "dr. Ani");
        Pegawai bagus = new Pegawai("4567", "dr. Bagus");

        Pegawai desi = new Pegawai("1234", "Ns. Desi");
        Pegawai eka = new Pegawai("4567", "Ns. Eka");

        Pasien pasien1 = new Pasien("Puspa Widya", "343298");
        pasien1.tambahKonsultasi(LocalDate.of(2021 , 8 , 11), ani, desi);
        pasien1.tambahKonsultasi(LocalDate.of(2021 , 9 , 11), bagus, eka);

        System.out.println(pasien1.getInfo());

        Pasien pasien2 = new Pasien("Yenny Anggraeni", "997744");
        System.out.println(pasien2.getInfo());
    }
}
```

r. *Compile* kemudian *run* RumahSakitDemo dan didapatkan hasil seperti berikut:

```
No Rekam Medis      : Puspa Widya
Nama                : 343298
Riwayat Konsultasi :
        Tanggal: 2021-08-11, Dokter: dr. Ani (1234), Perawat: Ns. Desi (1234)
        Tanggal: 2021-09-11, Dokter: dr. Bagus (4567), Perawat: Ns. Eka (4567)


No Rekam Medis      : Yenny Anggraeni
Nama                : 997744
Belum ada riwayat konsultasi
```

Output dari kode diatas tebalik

```
No Rekam Medis      : 343298
Nama                : Puspa Widya
Riwayat Konsultasi :
       Tanggal :2021-08-11, Dokter :dr. Ani ( 1234 } , Perawat :Ns. Desi ( 1234 }
       Tanggal :2021-09-11, Dokter :dr. Bagus ( 4567 } , Perawat :Ns. Eka ( 4567 }

No Rekam Medis      : 997744
Nama                : Yenny Anggraeni
Belum ada riwayat konsultasi

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum4>
```

**Pertanyaan**

Berdasarkan percobaan 1, jawablah pertanyaan-pertanyaan yang terkait:

1. Di dalam *class* Pegawai, Pasien, dan Konsultasi, terdapat method *setter* dan *getter* untuk masing-masing atributnya. Apakah gunanya *method setter* dan *getter* tersebut ?

   Jawab : Method setter dan getter digunakan untuk mengatur (set) dan mengambil (get) nilai dari atribut dalam sebuah objek. Setter digunakan untuk mengubah nilai dari atribut, sedangkan getter digunakan untuk mengambil nilai dari atribut tersebut. Dengan adanya method setter dan getter, akses terhadap atribut dapat dikontrol dan dilakukan dengan cara yang aman, karena nilai atribut tidak dapat diakses atau diubah secara langsung dari luar kelas.
2. Di dalam *class* Konsultasi tidak secara eksplisit terdapat constructor dengan parameter. Apakah ini berarti class Konsultasi tidak memiliki constructor?

   Jawab : Tidak secara eksplisit terdapat constructor dengan parameter dalam class Konsultasi. Namun, secara default Java akan menyediakan constructor tanpa parameter jika tidak ada constructor yang didefinisikan di dalam sebuah kelas. Jadi, walaupun tidak ada constructor yang didefinisikan, class Konsultasi tetap memiliki constructor.
3. Perhatikan *class* Konsultasi, atribut mana saja yang bertipe *object*?

   Jawab : Atribut bertipe object dalam class Konsultasi adalah dokter dan perawat, yang keduanya merupakan objek dari class Pegawai.
4. Perhatikan *class* Konsultasi, pada baris manakah yang menunjukan bahwa *class* Konsultasi memiliki relasi dengan *class* Pegawai?

   Jawab : Pada baris info += ", Dokter :" + dokter.getInfo(); dan info += ", Perawat :" + perawat.getInfo(); dalam method getInfo() class Konsultasi menunjukkan bahwa class Konsultasi memiliki relasi dengan class Pegawai.
5. Perhatikan pada *class* Pasien, apa yang dilakukan oleh kode `konsultasi.getInfo()`?

   Jawab : Pada kode konsultasi.getInfo(), ini mengambil informasi tentang konsultasi yang terjadi. Method getInfo() dalam class Konsultasi mengembalikan informasi tentang tanggal konsultasi, dokter yang melakukan konsultasi, dan perawat yang terlibat dalam konsultasi tersebut.
6. Pada method `getInfo()` dalam class Pasien, terdapat baris kode:
   ```
   if (!riwayatKonsultasi.isEmpty())
   ```
   Apakah yang dilakukan oleh baris tersebut?

   Jawab : Pada baris tersebut, kita memeriksa apakah daftar riwayatKonsultasi tidak kosong. Jika tidak kosong, artinya pasien memiliki riwayat konsultasi dan informasi tentang konsultasi-konsultasi tersebut akan ditambahkan ke informasi pasien. Jika kosong, maka akan ditampilkan pesan "Belum ada riwayat konsultasi".
7. Pada constructor class Pasien, terdapat baris kode:
   ```
   this.riwayatKonsultasi = new ArrayList<>();
   ```
   Apakah yang dilakukan oleh baris tersebut? Apakah yang terjadi jika baris tersebut dihilangkan?

   Jawab : Baris this.riwayatKonsultasi = new ArrayList<>(); dalam constructor class Pasien digunakan untuk membuat objek ArrayList baru yang akan menampung riwayat konsultasi pasien tersebut. Jika baris tersebut dihilangkan, maka

riwayatKonsultasi akan tetap null (belum diinisialisasi) dan akan menyebabkan NullPointerException ketika mencoba menambahkan konsultasi ke dalamnya.

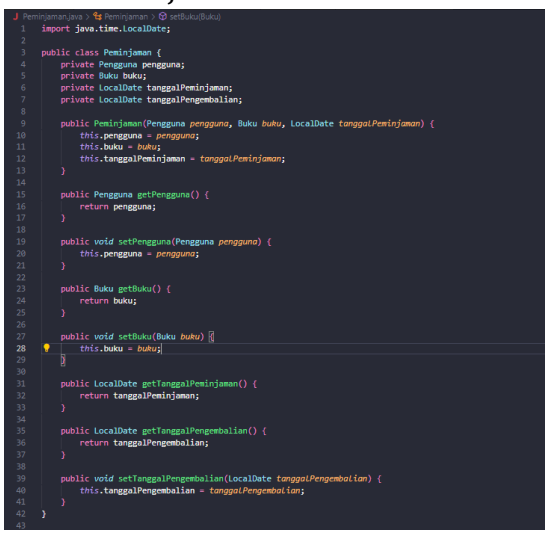
## IV. Tugas

Implementasikan studi kasus yang telah dibuat pada tugas PBO Teori ke dalam program

Class Buku

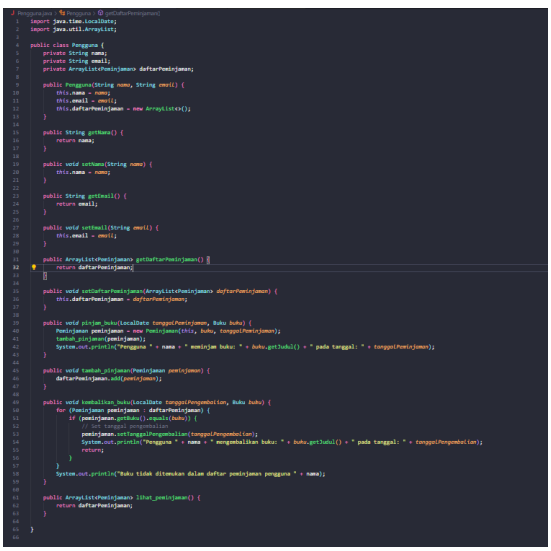
```java
public class Buku {
    private String judul;
    private String pengarang;
    private int jumlah_halaman;

    public Buku(String judul, String pengarang, int jumlah_halaman) {
        this.judul = judul;
        this.pengarang = pengarang;
        this.jumlah_halaman = jumlah_halaman;
    }

    public String getJudul() {
        return judul;
    }

    public void setJudul(String judul) {
        this.judul = judul;
    }

    public String getPengarang() {
        return pengarang;
    }

    public void setPengarang(String pengarang) {
        this.pengarang = pengarang;
    }

    public int getJumlahHalaman() {
        return jumlah_halaman;
    }

    public void setJumlahHalaman(int jumlah_halaman) {
        this.jumlah_halaman = jumlah_halaman;
    }
}
```

Class Peminjaman



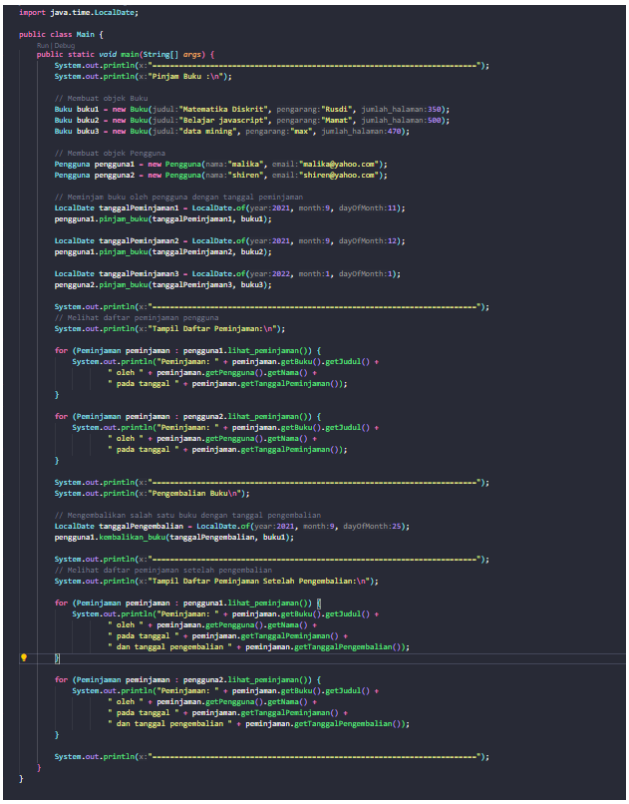
```java
import java.time.LocalDate;

public class Peminjaman {
    private Pengguna pengguna;
    private Buku buku;
    private LocalDate tanggalPeminjaman;
    private LocalDate tanggalPengembalian;

    public Peminjaman(Pengguna pengguna, Buku buku, LocalDate tanggalPeminjaman) {
        this.pengguna = pengguna;
        this.buku = buku;
        this.tanggalPeminjaman = tanggalPeminjaman;
    }

    public Pengguna getPengguna() {
        return pengguna;
    }

    public void setPengguna(Pengguna pengguna) {
        this.pengguna = pengguna;
    }

    public Buku getBuku() {
        return buku;
    }

    public void setBuku(Buku buku) {
        this.buku = buku;
    }

    public LocalDate getTanggalPeminjaman() {
        return tanggalPeminjaman;
    }

    public LocalDate getTanggalPengembalian() {
        return tanggalPengembalian;
    }

    public void setTanggalPengembalian(LocalDate tanggalPengembalian) {
        this.tanggalPengembalian = tanggalPengembalian;
    }
}
```

Class Pengguna



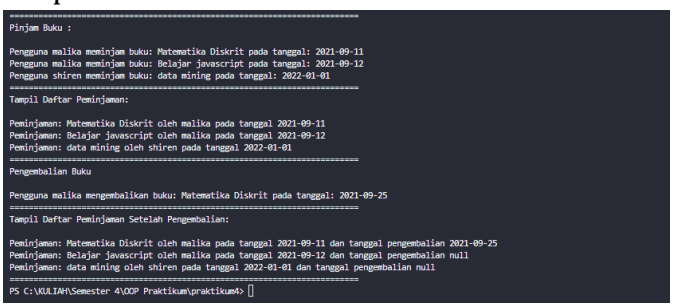
```java
import java.time.LocalDate;
import java.util.ArrayList;

public class Pengguna {
    private String nama;
    private String email;
    private ArrayList<Peminjaman> daftarPeminjaman;

    public Pengguna(String nama, String email) {
        this.nama = nama;
        this.email = email;
        this.daftarPeminjaman = new ArrayList<>();
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public ArrayList<Peminjaman> getDaftarPeminjaman() {
        return daftarPeminjaman;
    }

    public void setDaftarPeminjaman(ArrayList<Peminjaman> daftarPeminjaman) {
        this.daftarPeminjaman = daftarPeminjaman;
    }

    public void pinjam_buku(LocalDate tanggalPeminjaman, Buku buku) {
        Peminjaman peminjaman = new Peminjaman(this, buku, tanggalPeminjaman);
        tambah_pinjaman(peminjaman);
        System.out.println("Pengguna " + nama + " meminjam buku: " + buku.getJudul() + " pada tanggal: " + tanggalPeminjaman);
    }

    public void tambah_pinjaman(Peminjaman peminjaman) {
        daftarPeminjaman.add(peminjaman);
    }

    public void kembalikan_buku(LocalDate tanggalPengembalian, Buku buku) {
        for (Peminjaman peminjaman : daftarPeminjaman) {
            if (peminjaman.getBuku().equals(buku)) {
                peminjaman.setTanggalPengembalian(tanggalPengembalian);
                System.out.println("Pengguna " + nama + " mengembalikan buku: " + buku.getJudul() + " pada tanggal: " + tanggalPengembalian);
                return;
            }
        }
        System.out.println("Buku tidak ditemukan dalam daftar peminjaman pengguna " + nama);
    }

    public ArrayList<Peminjaman> lihat_peminjaman() {
        return daftarPeminjaman;
    }
}
```

Class Main



```java
import java.time.LocalDate;

public class Main {
    public static void main(String[] args) {
        System.out.println("------------------------------------------------------------------------");
        System.out.println("Pinjam Buku :\n");

        // Membuat objek Buku
        Buku buku1 = new Buku(judul:"Matematika Diskrit", pengarang:"Rusdi", jumlah_halaman:350);
        Buku buku2 = new Buku(judul:"Belajar javascript", pengarang:"Manat", jumlah_halaman:500);
        Buku buku3 = new Buku(judul:"data mining", pengarang:"max", jumlah_halaman:470);

        // Membuat objek Pengguna
        Pengguna pengguna1 = new Pengguna(nama:"malika", email:"malika@yahoo.com");
        Pengguna pengguna2 = new Pengguna(nama:"shiren", email:"shirer@yahoo.com");

        // Meminjam buku oleh pengguna dengan tanggal peminjaman
        LocalDate tanggalPeminjaman1 = LocalDate.of(year:2021, month:9, dayOfMonth:11);
        pengguna1.pinjam_buku(tanggalPeminjaman1, buku1);

        LocalDate tanggalPeminjaman2 = LocalDate.of(year:2021, month:9, dayOfMonth:12);
        pengguna1.pinjam_buku(tanggalPeminjaman2, buku2);

        LocalDate tanggalPeminjaman3 = LocalDate.of(year:2022, month:1, dayOfMonth:1);
        pengguna2.pinjam_buku(tanggalPeminjaman3, buku3);

        System.out.println("------------------------------------------------------------------------");
        // Melihat daftar peminjaman pengguna
        System.out.println("Tampil Daftar Peminjaman:\n");

        for (Peminjaman peminjaman : pengguna1.lihat_peminjaman()) {
            System.out.println("Peminjaman: " + peminjaman.getBuku().getJudul() +
                    " oleh " + peminjaman.getPengguna().getNama() +
                    " pada tanggal " + peminjaman.getTanggalPeminjaman());
        }

        for (Peminjaman peminjaman : pengguna2.lihat_peminjaman()) {
            System.out.println("Peminjaman: " + peminjaman.getBuku().getJudul() +
                    " oleh " + peminjaman.getPengguna().getNama() +
                    " pada tanggal " + peminjaman.getTanggalPeminjaman());
        }

        System.out.println("------------------------------------------------------------------------");
        System.out.println("Pengembalian Buku\n");

        // Mengembalikan salah satu buku dengan tanggal pengembalian
        LocalDate tanggalPengembalian = LocalDate.of(year:2021, month:9, dayOfMonth:25);
        pengguna1.kembalikan_buku(tanggalPengembalian, buku1);

        System.out.println("------------------------------------------------------------------------");
        // Melihat daftar peminjaman setelah pengembalian
        System.out.println("Tampil Daftar Peminjaman Setelah Pengembalian:\n");

        for (Peminjaman peminjaman : pengguna1.lihat_peminjaman()) {
            System.out.println("Peminjaman: " + peminjaman.getBuku().getJudul() +
                    " oleh " + peminjaman.getPengguna().getNama() +
                    " pada tanggal " + peminjaman.getTanggalPeminjaman() +
                    " dan tanggal pengembalian " + peminjaman.getTanggalPengembalian());
        }

        for (Peminjaman peminjaman : pengguna2.lihat_peminjaman()) {
            System.out.println("Peminjaman: " + peminjaman.getBuku().getJudul() +
                    " oleh " + peminjaman.getPengguna().getNama() +
                    " pada tanggal " + peminjaman.getTanggalPeminjaman() +
                    " dan tanggal pengembalian " + peminjaman.getTanggalPengembalian());
        }

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

Output :


Pinjam Buku :
Pengguna malika meminjam buku: Matematika Diskrit pada tanggal: 2021-09-11
Pengguna malika meminjam buku: Belajar javascript pada tanggal: 2021-09-12
Pengguna shiren meminjam buku: data mining pada tanggal: 2022-01-01
Tampil Daftar Peminjaman:
Peminjaman: Matematika Diskrit oleh malika pada tanggal 2021-09-11
Peminjaman: Belajar javascript oleh malika pada tanggal 2021-09-12
Peminjaman: data mining oleh shiren pada tanggal 2022-01-01
Pengembalian Buku
Pengguna malika mengembalikan buku: Matematika Diskrit pada tanggal: 2021-09-25
Tampil Daftar Peminjaman Setelah Pengembalian:
Peminjaman: Matematika Diskrit oleh malika pada tanggal 2021-09-11 dan tanggal pengembalian 2021-09-25
Peminjaman: Belajar javascript oleh malika pada tanggal 2021-09-12 dan tanggal pengembalian null
Peminjaman: data mining oleh shiren pada tanggal 2022-01-01 dan tanggal pengembalian null
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum4>