LAPORAN UTS PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

NAMA : **AFRIDHO IKHSAN**
NPM : **2210631170002**
KELAS : **3A INFORMATIKA**

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2023

DAFTAR ISI

DAFTAR ISI i
SOAL 2
JAWABAN 3

SOAL

1. Anda memiliki kelas dasar Hewan dan dua kelas turunan, Kucing dan Anjing. Setiap hewan memiliki atribut jenisSuara dan metode bersuara(). Kucing bersuara "Meong" dan Anjing bersuara "Guk Guk." Buat program yang meminta pengguna memasukkan jenis hewan (Kucing/Anjing) dan kemudian mencetak suara hewan tersebut dengan memanfaatkan konsep inheritance, polimorfisme, dan tambahkan percabangan untuk mengecek jika hewan tersebut adalah mamalia atau bukan.

2. Buatlah sebuah sistem pengelolaan perpustakaan yang menerapkan konsep 4 pilar pemograman berbasis objek yaitu Inheritance, Encapsulation, Polymorphism, Abstraction. Pada system ini nantinya akan memiliki kelas pengguna dan dua kelas turunannya yaitu user dan admin. Dan system ini nantinya juga akan memiliki kelas buku yang berfungsi untuk menyimpan buku berdasarkan genre nya. System pengelolaan perpustakaan ini harus dapat menidentifikasi customer atau pun admin. seorang admin dapat menambahkan, melihat, mendelete buku. sedangkan user hanya dapat melihat
buku saja.

JAWABAN

1. Anda memiliki kelas dasar Hewan dan dua kelas turunan, Kucing dan Anjing. Setiap hewan memiliki atribut jenisSuara dan metode bersuara(). Kucing bersuara "Meong" dan Anjing bersuara "Guk Guk." Buat program yang meminta pengguna memasukkan jenis hewan (Kucing/Anjing) dan kemudian mencetak suara hewan tersebut dengan memanfaatkan konsep inheritance, polimorfisme, dan tambahkan percabangan untuk mengecek jika hewan tersebut adalah mamalia atau bukan.

   a) Class Hewan

```
class Hewan {
    protected String jenisSuara;

    public void bersuara() {
        System.out.println(this.jenisSuara);
    }
}
```

- Atribute jenisSuara digunakan untuk menampung string suara hewan pada subclass yang akan dibuat menggunakan class Hewan ini. Attribute ini diatur protected karena nilai dari attribute ini hanya digunakan di dalam class Hewan ini beserta subclass nya (digunakan di dalam method bersuara()).
- Method bersuara() digunakan untuk mencetak nilai dari attribute jenisSuara. Method ini diatur ke public karena akan digunakan di class lain selain class subclass, yaitu class DemoHewan.

   b) Class Kucing

```
class Kucing extends Hewan {
    protected String jenisSuara = "Meong";

    public void bersuara() {
        System.out.println("Kucing: " + this.jenisSuara);
//Polymorphism Override
    }
}
```

- Salah satu subclass dari class Hewan.
- Pada class Kucing, attribute jenisSuara diisi dengan nilai "Meong"
- Terdapat karakteristik override polymorphism pada method bersuara() . (Dengan tambahan output "Kucing: ")

c) Class Anjing

```
class Anjing extends Hewan {
    protected String jenisSuara = "Guk Guk";

    public void bersuara() {
        System.out.println("Anjing: " + this.jenisSuara);
//Polymorphism Override
    }
}
```

- Salah satu subclass dari class Hewan.
- Pada class Anjing, attribute jenisSuara diisi dengan nilai "Guk Guk"
- Terdapat karakteristik override polymorphism pada method bersuara() . (Dengan tambahan output "Anjing: ")

d) Class DemoHewan

```
import java.util.Scanner;

class DemoHewan {
    static public void main (String[] args) {
        Scanner masukkan = new Scanner(System.in);


        System.out.println("Pilihan Hewan");
        System.out.println("1.  Kucing");
        System.out.println("2.  Anjing");
        System.out.print("Masukkan jenis hewan : ");

        int jenisHewan = masukkan.nextInt();

        if (jenisHewan == 1) {
            Kucing kucing = new Kucing();
            kucing.bersuara(); //Inheritence (Penggunaan method
milik Class Hewan dari class Kucing)
            System.out.println("Kucing adalah Mamalia.");
        } else if (jenisHewan == 2) {
            Anjing anjing =new Anjing();
            anjing.bersuara();  //Inheritence (Penggunaan method
milik Class Hewan dari class Anjing)
            System.out.println( "Anjing juga Mamalia.");
        } else {
            System.out.println("Inputan anda salah.");
        }

    }
}
```

- Class DemoHewan ini digunakan untuk menjalankan program utama, dengan menambahkan method main di dalamnya.
- Pada method main, terdapat implementasi penerimaan input dan penampilan output dari pengguna dengan memanfaatkan salah satu library dari Java, yaitu java.util.Scanner.
- Dari input yang pengguna masukkan, dilakukan pengecekkan hewan apa yang ingin ditampilkan suaranya menggunakan percabangan.
- Karakteristik inheritence juga dapat dilihat di dalam method main dalam class ini, yaitu saat pemanggilan class Kucing & Anjing yang juga dapat method (bersuara()) milik class Hewan.

2. Buatlah sebuah sistem pengelolaan perpustakaan yang menerapkan konsep 4 pilar pemograman berbasis objek yaitu Inheritance, Encapsulation, Polymorphism, Abstraction. Pada system ini nantinya akan memiliki kelas pengguna dan dua kelas turunannya yaitu user dan admin. Dan system ini nantinya juga akan memiliki kelas buku yang berfungsi untuk menyimpan buku berdasarkan genre nya. System pengelolaan perpustakaan ini harus dapat menidentifikasi customer atau pun admin. seorang admin dapat menambahkan, melihat, mendelete buku. sedangkan user hanya dapat melihat buku saja.
   a) Class Pengguna

```
public abstract class Pengguna {
    public String tipePengguna;
    public int adminID;

    abstract void menambahkanBuku(Buku[] rakBuku, Buku bukuBaru);

    abstract void menghapusBuku(Buku[] rakBuku, String
bukuYangDihapus);

    public void melihatBuku(Buku[] rakBuku, String buku) {
        for (int i = 0; i < rakBuku.length; i++) {
            if
(rakBuku[i].judulBuku.toLowerCase().contains(buku.toLowerCase()))
{
                System.out.println("Buku ditemukan ");
                System.out.println("Judul buku: " +
rakBuku[i].judulBuku);
                System.out.println("Tahun terbit: " +
rakBuku[i].tahunTerbit);
                System.out.println("Penulis: " +
rakBuku[i].penulis);
```

```
                System.out.println("Halaman buku: " +
rakBuku[i].halamanBuku);
                System.out.println("Genre buku: " +
rakBuku[i].genreBuku);
                return;
            }
        }

        System.out.println("Buku yang dicari tidak ditemukan.");
        return;
    }

    public void melihatDaftarBuku(Buku[] rakBuku, String
genreDitampilkan) {
        if (rakBuku[0] == null) {
            System.out.println("Daftar buku kosong.");
            return;
        }

        if (genreDitampilkan == null) {
            System.out.println("Daftar buku yang tersedia");
            for (int i = 0; i < rakBuku.length; i++) {
                System.out.println((i + 1) + ". " +
rakBuku[i].judulBuku + " (Genre: " + rakBuku[i].genreBuku + ").");

                if (rakBuku[i + 1] == null)
                    return;
            }
        } else {
            System.out.println("Daftar buku dengan genre '" +
genreDitampilkan + "':");

            // Buat variabel untuk melacak apakah ada buku yang
sesuai dengan genre
            boolean adaBuku = false;

            // Iterasi melalui rakBuku dan tampilkan buku dengan
genre yang sesuai
            for (Buku buku : rakBuku) {
                if (buku != null &&
buku.genreBuku.toLowerCase().equals(genreDitampilkan.toLowerCase()
)) {
                    System.out.println("Judul: " +
buku.judulBuku);
                    System.out.println("Penulis: " +
buku.penulis);
                    System.out.println("Genre: " +
buku.genreBuku);
```

```
                    System.out.println("Tahun Terbit: " +
buku.tahunTerbit);
                    System.out.println("Halaman Buku: " +
buku.halamanBuku);
                    System.out.println("----------------");
                    adaBuku = true; // Setel adaBuku menjadi true
jika ada buku yang sesuai dengan genre
                }
            }

            // Tampilkan pesan jika tidak ada buku yang sesuai
dengan genre
            if (!adaBuku) {
                System.out.println("Tidak ada buku dengan genre '"
+ genreDitampilkan + "'.");
            }
        }

    }
}
```

- Class Pengguna berperan sebagai superclass untuk class User & class Admin yang menurunkan (inherit) attribute dan method nya kepada kedua class tersebut.
- Attribute :
  - jenisPengguna : Menampung identitas jenis pengguna pada instance yang dibuat menggunakan subclass dari class Pengguna dalam bentuk string
  - adminID : Menampung ID dari admin pengguna, bernilai 0 jika selain subclass Admin.
- Method :
  - menambahkanBuku() : Masih dideklarasikan dalam bentuk abstract (Akan diatur di dalam subclass)
  - menghapusBuku() : Masih dideklarasikan dalam bentuk abstract (Akan diatur di dalam subclass)
  - melihatBuku() : Mencari dan menampilkan data dari buku yang ingin dicari (Menggunakan perulangan untuk mencari kesamaan judul buku yang ada dalam daftar buku dengan apa yang pengguna cari)
  - melihatDaftarBuku() : Menampilkan seluruh daftar buku yang tersedia. Memiliki 2 fungsionalitas, menampilkan seluruh buku jika argument genreDitampilkan diatur ke null dan hanya menampilkan buku dengan genre yang dicari jika argument genreDitampilkan diisi dengan sebuah nilai.

b) Class User

```
public class User extends Pengguna {
    public String tipePengguna = "Pengguna";

    public User (String tipePengguna) { //Overloading Polymorphism
        this.tipePengguna = tipePengguna;
        this.adminID = 0;
    }

    void menambahkanBuku(Buku[] rakBuku, Buku bukuBaru) {
       System.out.println("Anda tidak memiliki akses untuk
menambahkan buku. (Hanya admin)");
    }

    void menghapusBuku(Buku[] rakBuku, String bukuYangDihapus) {
       System.out.println("Anda tidak memiliki akses untuk
menghapus buku. (Hanya admin)");
    }
}
```

- Class User merupakan subclass dari class Pengguna.
- Memiliki Parameterized Constructor untuk memberikan nilai hanya pada attribute tipePengguna. (attribute adminID diatur ke 0, untuk menandakan bahwa instance dari User bukan merupakan admin)
- Method :
  - menambahkanBuku() : Mengatur apa yang dilakukan oleh method menambahkanBuku pada class User, yaitu hanya menampilkan string yang menjelaskan bahwa User tidak memiliki akses untuk menambahkan buku. (Overriding Polymorphism)
  - menghapusBuku() : Mengatur apa yang dilakukan oleh method menghapusBuku pada class User, yaitu hanya menampilkan string yang menjelaskan bahwa User tidak memiliki akses untuk menghapus buku. (Overriding Polymorphism)

c) Class Admin

```
public class Admin extends Pengguna {
    public String tipePengguna = "Admin";

    public Admin(String tipePengguna, int adminID) { //
Overloading Polymorphism
        this.tipePengguna = tipePengguna;
        this.adminID = adminID;
    }

    void menambahkanBuku(Buku[] rakBuku, Buku bukuBaru) {
        // Cari indeks pertama yang memiliki nilai null pada array
rakBuku
```

```
        int indeksKosong = -1;

        for (int i = 0; i < rakBuku.length; i++) {
            if (rakBuku[i] == null) {
                indeksKosong = i;
                break;
            }
        }

        // Jika ada indeks yang kosong, tambahkan buku baru
        if (indeksKosong != -1) {
            rakBuku[indeksKosong] = bukuBaru;
            System.out.println("Buku '" + bukuBaru.judulBuku + "'
telah ditambahkan ke rak buku.");
        } else {
            // Jika tidak ada indeks kosong, keluarkan pesan
kesalahan (rak penuh)
            System.out.println("Rak buku penuh. Tidak dapat
menambahkan buku baru.");
        }
    }

    void menghapusBuku(Buku[] rakBuku, String bukuYangDihapus) {
        // Inisialisasi indeks buku yang akan dihapus dengan -1
(tidak ditemukan)
        int indeksDihapus = -1;

        // Mencari indeks buku yang akan dihapus
        for (int i = 0; i < rakBuku.length; i++) {
            // Periksa apakah elemen array tidak null sebelum
mengakses properti judul
            if (rakBuku[i] != null &&
rakBuku[i].judulBuku.toLowerCase().equals(bukuYangDihapus.toLowerC
ase())) {
                indeksDihapus = i;
                break; // Keluar dari loop saat buku ditemukan
            }
        }

        // Jika buku yang akan dihapus ditemukan
        if (indeksDihapus != -1) {
            // Menggeser elemen-elemen setelah indeksDihapus ke
kiri
            for (int i = indeksDihapus; i < rakBuku.length - 1;
i++) {
                rakBuku[i] = rakBuku[i + 1];
            }

            // Mengurangi panjang array rakBuku
```

```
            rakBuku[rakBuku.length - 1] = null;

            // Output pesan bahwa buku telah dihapus
            System.out.println("Buku '" + bukuYangDihapus + "'
telah dihapus dari rak buku.");
        } else {
            // Output pesan jika buku tidak ditemukan
            System.out.println("Buku '" + bukuYangDihapus + "'
tidak ditemukan di rak buku.");
        }
    }

}
```

- Class Admin merupakan subclass dari class Pengguna.
- Memiliki Parameterized Constructor untuk memberikan nilai pada attribute tipePengguna dan adminID.
- Method :
  - menambahkanBuku() : Mengatur apa yang dilakukan oleh method menambahkanBuku pada class Admin, yaitu logika untuk menambahkan buku pada array daftar buku. (Overriding Polymorphism)
  - menghapusBuku() : Mengatur apa yang dilakukan oleh method menghapusBuku pada class Admin, yaitu logika untuk menghapus buku pada array daftar buku. (Overriding Polymorphism)

d) Class Buku

```
public class Buku {
    public String judulBuku;
    public int tahunTerbit;
    public String penulis;
    public int halamanBuku;
    public String genreBuku;

    public Buku(String judulBuku, int tahunTerbit, String penulis,
int halamanBuku, String genreBuku) {
        this.judulBuku = judulBuku;
        this.tahunTerbit = tahunTerbit;
        this.penulis = penulis;
        this.halamanBuku = halamanBuku;
        this.genreBuku = genreBuku;
    }

}
```

- Class Buku yang dibuat untuk menggambarkan entitas sebuah buku, yang memiliki berbagai data terkait pada buku tersebut.
- Memiliki Parameterized Constructor untuk memberikan nilai pada setiap attribute yang class Buku ini miliki.

e) Class SistemPerpustakaan

```
import java.util.Scanner;

class SistemPerpustakaan {
    static public void main(String[] args) {
        Scanner inputPengguna = new Scanner(System.in);
        boolean ulangProgram = true;

        Pengguna pengguna;

        Buku[] daftarBuku = new Buku[100];
        daftarBuku[0] = new Buku("Filosofi Teras", 2004, "Henry
Manampiring", 204, "Self improvement");

        System.out.println("Login sebagai ");
        System.out.println("1. User");
        System.out.println("2. Admin");
        System.out.print("Pilih: ");
        int loginSebagai = inputPengguna.nextInt();

        bersihkanTerminal();

        if (loginSebagai == 1) {
            pengguna = new User("Pengguna");
        } else if (loginSebagai == 2) {
            pengguna = new Admin("Admin", 2210);
        } else {
            pengguna = new User("Pengguna");
            System.out.print("Input yang anda inputPengguna
salah.");
        }

        do {
            tampilMenu();
            int pilMenu = inputPengguna.nextInt();

            switch (pilMenu) {
                case 1:
                    bersihkanTerminal();
                    menuMelihatDaftarBuku(pengguna, daftarBuku);
                    break;
                case 2:
                    bersihkanTerminal();
                    menuMencariBuku(pengguna, daftarBuku);
                    break;
                case 3:
                    bersihkanTerminal();
                    menuMenambahkanBuku(pengguna, daftarBuku);
```

```
                    break;
                case 4:
                    bersihkanTerminal();
                    menuMenghapusBuku(pengguna, daftarBuku);
                    break;
                case 5:
                    bersihkanTerminal();
                    ulangProgram = false;

                    break;
                default:
                    break;
            }
        } while (ulangProgram == true);

    }

    static protected boolean apakahAdmin(Pengguna pengguna) {
        if (pengguna.adminID == 0)
            return false;
        else
            return true;
    }

    static protected void bersihkanTerminal() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }

    static protected void tampilMenu() {
        bersihkanTerminal();
        System.out.println("Menu program");
        System.out.println("1. Melihat daftar buku");
        System.out.println("2. Mencari buku");
        System.out.println("3. Menambahkan buku");
        System.out.println("4. Menghapus buku.");
        System.out.println("5. Keluar");
        System.out.print("inputPengguna pilihan: ");
    }

    static protected void menuMelihatDaftarBuku(Pengguna pengguna,
Buku[] daftarBuku) {
        int pilMenu;
        Scanner inputPengguna = new Scanner(System.in);

        System.out.println("Pilih Menu");
        System.out.println("1. Semua buku");
        System.out.println("2. Buku berdasarkan genre");
        System.out.print("inputPengguna pilihan: ");
```

```
        pilMenu = inputPengguna.nextInt();

        switch (pilMenu) {
            case 1:
                bersihkanTerminal();
                pengguna.melihatDaftarBuku(daftarBuku, null);
                break;
            case 2:
                bersihkanTerminal();
                String pilGenre;
                System.out.print("Inputkan genre buku yang ingin
anda lihat: ");
                inputPengguna.nextLine();
                pilGenre = inputPengguna.nextLine();

                pengguna.melihatDaftarBuku(daftarBuku, pilGenre);
                break;
            default:
                break;
        }

        if (pilMenu == 1) inputPengguna.nextLine();
        inputPengguna.nextLine();
    }

    static protected void menuMencariBuku(Pengguna pengguna,
Buku[] daftarBuku) {
        Scanner inputPengguna = new Scanner(System.in);
        System.out.print("Inputkan judul buku yang ingin anda
cari: ");
        String bukuDicari = inputPengguna.nextLine();

        pengguna.melihatBuku(daftarBuku, bukuDicari);
        inputPengguna.nextLine();
    }

    static protected void menuMenambahkanBuku(Pengguna pengguna,
Buku[] daftarBuku) {
        Scanner inputPengguna = new Scanner(System.in);

        if (apakahAdmin(pengguna)) {
            System.out.println("inputPengguna data yang
diperlukan");
            System.out.print("Judul buku: ");
            String judulBuku = inputPengguna.nextLine();
            System.out.print("Tahun terbit: ");
            int tahunTerbit = inputPengguna.nextInt();
            System.out.print("Penulis: ");
            inputPengguna.nextLine();
```

```
            String penulis = inputPengguna.nextLine();
            System.out.print("Halaman buku: ");
            int halamanBuku = inputPengguna.nextInt();
            System.out.print("Genre Buku: ");
            inputPengguna.nextLine();
            String genreBuku = inputPengguna.nextLine();

            pengguna.menambahkanBuku(daftarBuku,
                    new Buku(judulBuku, tahunTerbit, penulis,
halamanBuku, genreBuku));

            System.out.println("Buku berhasil ditambahkan.");
        } else {
            pengguna.menambahkanBuku(null, null);
        }

        inputPengguna.nextLine();
    }

    static protected void menuMenghapusBuku(Pengguna pengguna,
Buku[] daftarBuku) {
        Scanner inputPengguna = new Scanner(System.in);

        if (apakahAdmin(pengguna)) {
            System.out.print("Inputkan judul buku yang ingin
dihapus : ");
            String judulBuku = inputPengguna.nextLine();

            pengguna.menghapusBuku(daftarBuku, judulBuku);
        } else {
            pengguna.menghapusBuku(null, null);
        }

        inputPengguna.nextLine();
    }
}
```

- Class SistemPerpustakaan merupakan class utama yang digunakan untuk titik awal eksekusi program.
- import java.util.Scanner : Mengimpor kelas Scanner dari paket java.util, yang digunakan untuk membaca input dari pengguna.
- Deklarasi variabel :

```
static public void main(String[] args) {
        Scanner inputPengguna = new Scanner(System.in);
        Buku[] daftarBuku = new Buku[500];
        Pengguna pengguna;
        boolean ulangProgram = true;
```

  - inputPengguna digunakan untuk membaca input dari pengguna.

- daftarBuku adalah array yang menyimpan objek buku. Array ini berukuran 500, yang artinya sistem perpustakaan pada program ini hanya dapat menampung maksimal 500 buku.
- pengguna adalah objek dari kelas Pengguna.
- ulangProgram digunakan untuk mengontrol perulangan program akan terus berlanjut atau tidak.

- Inisialisasi array daftarBuku untuk menampung seluruh buku yang tersedia di perpustakaan.

```
        daftarBuku[0] = new Buku("You do you", 2020,
"Fellexandro Ruby", 236, "Self improvement");
```

Inisialisasi elemen pertama dari array daftarBuku dengan objek buku.

- Login dan Pemilihan Peran

```
System.out.println("Login sebagai ");
        System.out.println("1. User");
        System.out.println("2. Admin");
        System.out.print("Pilih: ");
        int loginSebagai = inputPengguna.nextInt();
```

Ditahap eksekusi kode ini, pengguna akan diminta untuk memilih peran (User atau Admin) melalui input.

- Loop utama pada program

```
do {
            tampilMenu();
            int pilMenu = inputPengguna.nextInt();
            //..
} while (ulangProgram == true);
```

Program berada dalam loop do-while utama yang terus berjalan selama ulangProgram bernilai true. Pilihan menu pengguna dievaluasi dan metode yang sesuai dipanggil.

- Method :
  - main() : Method main sebagai titik awal eksekusi program.
  - apakahAdmin() : Memeriksa apakah pengguna adalah admin atau tidak.
  - bersihkanTerminal() : Menghapus tulisan/teks pada terminal.
  - tampilMenu() : Menampilkan menu program.
  - menuMelihatDaftarBuku() : Menampung logika untuk submenu Melihat Daftar Buku.
  - mencariBuku() : Menampung logika untuk submenu Mencari Buku.
  - menuMenambahkanBuku() : Menampung logika untuk submenu Menambahkan Buku.
  - menuMenghapusBuku() : Menampung logika untuk submenu Menghapus Buku.