**Inheritance**

package inheritance;

import java.util.Scanner;

public class BangunDatar {

protected int panjang, lebar;

public BangunDatar() {

}

public BangunDatar(int panjang, int lebar) {

this.panjang = panjang;

this.lebar = lebar;

}

public int luas() {

return panjang \* lebar;

public int keliling() {

return (panjang + lebar) \* 2;

}

}

class Segitiga extends BangunDatar {

private int alas, tinggi;

public Segitiga() {

}

public Segitiga(int alas, int tinggi) {

super(0, 0);

this.alas = alas;

this.tinggi = tinggi;

}

@Override

public int luas() {

return (alas \* tinggi) / 2;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

**// Input data bangun datar**

System.out.println("Masukkan panjang: ");

int panjang = scanner.nextInt();

System.out.println("Masukkan lebar: ");

int lebar = scanner.nextInt();

**// Buat objek bangun datar**

BangunDatar bangunDatar = new BangunDatar(panjang, lebar);

**// Cetak luas dan keliling bangun datar**

System.out.println("Luas bangun datar: " + bangunDatar.luas());

System.out.println("Keliling bangun datar: " + bangunDatar.keliling());

**// Input data segitiga**

System.out.println("Masukkan alas segitiga: ");

int alas = scanner.nextInt();

System.out.println("Masukkan tinggi segitiga: ");

int tinggi = scanner.nextInt();

**// Buat objek segitiga**

Segitiga segitiga = new Segitiga(alas, tinggi);

**// Cetak luas dan keliling segitiga**

System.out.println("Luas segitiga: " + segitiga.luas());

System.out.println("Keliling segitiga: " + segitiga.keliling());

}

}

Program ini akan menghitung luas dan keliling bangun datar. Terdapat dua kelas, yaitu BangunDatar dan Segitiga. Kelas BangunDatar merupakan kelas induk, sedangkan kelas Segitiga merupakan kelas turunan. Kelas BangunDatar memiliki dua atribut, yaitu panjang dan lebar, serta dua method, yaitu luas() dan keliling(). Kelas Segitiga memiliki dua atribut tambahan, yaitu alas dan tinggi.

**Encapsulation**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

**// Buat objek Person**

Person person = new Person();

**// Ambil input nama dari user**

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

String name = scanner.nextLine();

**// Atur nama objek Person**

person.setName(name);

**// Ambil input umur dari user**

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

int age = scanner.nextInt();

**// Atur umur objek Person**

person.setAge(age);

**// Cetak nama dan umur objek Person**

System.out.println("Nama: " + person.getName());

System.out.println("Umur: " + person.getAge());

}

}

class Person {

private String name;

private int age;

**// Metode getter untuk mendapatkan nama**

public String getName() {

return name;

}

**// Metode setter untuk mengatur nama**

public void setName(String name) {

this.name = name;

}

**// Metode getter untuk mendapatkan umur**

public int getAge() {

return age;

}

**// Metode setter untuk mengatur umur**

public void setAge(int age) {

this.age = age;

}

}

Pada kelas Person, terdapat dua variabel privat, yaitu name dan age. Kedua variabel ini tidak dapat diakses langsung dari luar kelas.

Untuk mengakses variabel name dan age, disediakan metode getter dan setter. Metode getter digunakan untuk mendapatkan nilai variabel, sedangkan metode setter digunakan untuk mengatur nilai variabel.

Pada kelas Main, terdapat objek person dari kelas Person. Objek ini dibuat menggunakan constructor default.

Dengan menggunakan konsep enkapsulasi, data dari objek Person tidak dapat diakses langsung dari luar kelas. Hal ini dapat mencegah terjadinya kesalahan data yang tidak disengaja.

**Polymorphism**

package polymorphism;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

**// Input jenis hewan**

System.out.print("Masukkan jenis hewan: ");

String jenisHewan = scanner.nextLine();

**// Buat objek hewan**

Hewan hewan = null;

if (jenisHewan.equals("kucing")) {

hewan = new Kucing();

} else if (jenisHewan.equals("anjing")) {

hewan = new Anjing();

} else {

System.out.println("Jenis hewan tidak dikenal.");

return;

}

**// Panggil method bersuara()**

hewan.bersuara();

}

}

class Hewan {

public void bersuara() {

System.out.println("Hewan bersuara...");

}

}

class Kucing extends Hewan {

@Override

public void bersuara() {

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

}

}

class Anjing extends Hewan {

@Override

public void bersuara() {

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

}

}

Class Main berfungsi sebagai titik masuk program.

Class Hewan adalah class induk dari Kucing dan Anjing. Class Hewan memiliki method bersuara() yang akan dipanggil oleh class turunannya.

Class Kucing dan Anjing adalah class turunan dari Hewan. Class Kucing mengimplementasikan method bersuara() dengan suara "Meong...", sedangkan class Anjing mengimplementasikan method bersuara() dengan suara "Guk...".

**Method Overloading**

package overloading;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

**// Buat objek Scanner**

Scanner scanner = new Scanner(System.in);

**// Input bilangan bulat**

System.out.print("Masukkan bilangan bulat: ");

int bil = scanner.nextInt();

**// Panggil method hitungLuas() dengan parameter bilangan bulat**

System.out.println("Luas persegi dengan sisi " + bil + " adalah " + hitungLuas(bil));

**// Input panjang dan lebar persegi panjang**

System.out.print("Masukkan panjang persegi panjang: ");

int p = scanner.nextInt();

System.out.print("Masukkan lebar persegi panjang: ");

int l = scanner.nextInt();

**// Panggil method hitungLuas() dengan parameter panjang dan lebar**

System.out.println("Luas persegi panjang dengan panjang " + p + " dan lebar " + l + " adalah " + hitungLuas(p, l));

}

**// Method hitungLuas() dengan parameter bilangan bulat**

public static int hitungLuas(int bil) {

return bil \* bil;

}

**// Method hitungLuas() dengan parameter panjang dan lebar**

public static int hitungLuas(int p, int l) {

return p \* l;

}

}

Pada program ini, kita memiliki dua method hitungLuas(). Method pertama memiliki satu parameter dengan tipe data int. Method kedua memiliki dua parameter dengan tipe data int. Kedua method ini memiliki nama yang sama, tetapi parameter dan tipe datanya berbeda. Oleh karena itu, kedua method ini disebut sebagai method overloading.

Method overloading memungkinkan kita untuk membuat method dengan nama yang sama, tetapi memiliki parameter dan tipe data yang berbeda. Hal ini dapat mempermudah penulisan program, terutama jika kita memiliki fungsi yang mirip tetapi memiliki perbedaan pada parameternya.

**Interface**

import java.util.Scanner;

interface Hewan {

void makan();

void tidur();

}

class Kucing implements Hewan {

@Override

public void makan() {

System.out.println("Kucing makan ikan");

}

@Override

public void tidur() {

System.out.println("Kucing tidur di atas karpet");

}

}

class Anjing implements Hewan {

@Override

public void makan() {

System.out.println("Anjing makan daging");

}

@Override

public void tidur() {

System.out.println("Anjing tidur di atas kasur");

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Pilih hewan: 1. Kucing, 2. Anjing");

int pilihan = scanner.nextInt();

Hewan hewan = null;

if (pilihan == 1) {

hewan = new Kucing();

} else if (pilihan == 2) {

hewan = new Anjing();

} else {

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

return;

}

hewan.makan();

hewan.tidur();

}

}

Dalam program ini, kita menggunakan interface Hewan untuk mendefinisikan perilaku umum dari hewan. Interface ini memiliki dua method abstrak, yaitu makan() dan tidur().

Class Kucing dan Anjing adalah class implementasi dari interface Hewan. Kedua class ini mengimplementasikan method makan() dan tidur() sesuai dengan perilaku hewan masing-masing.

Pada class Main, kita menggunakan Scanner untuk membaca inputan dari user. Setelah user memilih hewan, kita membuat objek hewan sesuai dengan pilihan user. Kemudian, kita memanggil method makan() dan tidur() pada objek hewan tersebut.

Dengan menggunakan konsep interface, kita dapat membuat program yang lebih fleksibel dan mudah dimodifikasi. Hal ini karena kita dapat membuat class implementasi baru dari interface tanpa perlu memodifikasi class utama.

**Abstract**

**// Import library yang dibutuhkan**

import java.util.Scanner;

**// Definisikan class abstrak Shape**

public abstract class Shape {

**// Definisikan atribut**

protected double p1;

protected double p2;

**// Definisikan method abstrak untuk menghitung luas**

public abstract double getArea();

**// Definisikan constructor**

public Shape(double p1, double p2) {

this.p1 = p1;

this.p2 = p2;

}

}

**// Definisikan class persegi yang mengimplementasikan class abstrak Shape**

public class Persegi extends Shape {

**// Definisikan constructor**

public Persegi(double p1, double p2) {

super(p1, p2);

}

**// Definisikan method getArea untuk menghitung luas persegi**

@Override

public double getArea() {

return p1 \* p2;

}

}

**// Definisikan class lingkaran yang mengimplementasikan class abstrak Shape**

public class Lingkaran extends Shape {

**// Definisikan constructor**

public Lingkaran(double p1, double p2) {

super(p1, p2);

}

**// Definisikan method getArea untuk menghitung luas lingkaran**

@Override

public double getArea() {

return Math.PI \* p1 \* p1;

}

}

**// Definisikan class main**

public class Main {

**// Definisikan variabel scanner**

static Scanner scanner = new Scanner(System.in);

**// Definisikan method main**

public static void main(String[] args) {

**// Menerima input dari user**

System.out.print("Masukkan panjang sisi persegi: ");

double p1 = scanner.nextDouble();

**// Menerima input dari user**

System.out.print("Masukkan lebar sisi persegi: ");

double p2 = scanner.nextDouble();

**// Membuat objek persegi**

Persegi persegi = new Persegi(p1, p2);

**// Menampilkan luas persegi**

System.out.println("Luas persegi adalah: " + persegi.getArea());

**// Menerima input dari user**

System.out.print("Masukkan jari-jari lingkaran: ");

double r = scanner.nextDouble();

**// Membuat objek lingkaran**

Lingkaran lingkaran = new Lingkaran(r, r);

**// Menampilkan luas lingkaran**

System.out.println("Luas lingkaran adalah: " + lingkaran.getArea());

}

}

Pada program di atas, kita telah menerapkan konsep abstract pada class Shape. Class Shape merupakan class abstrak karena memiliki method abstrak getArea().

Class Persegi dan class Lingkaran merupakan class turunan dari class Shape. Kedua class turunan tersebut mengimplementasikan method getArea() untuk menghitung luas persegi dan luas lingkaran