**LAPORAN RESMI**

Praktikum 17 Abstract Class dan Interface

Mata Kuliah: Praktek Pemrograman Berbasis Objek

****

Disusun oleh:

Bayu Hadi Leksana (3122500046)

2 D3 Teknik Informatika B

Dosen Pengampu: Andhik Ampuh Yunanto S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

1. **PERCOBAAN**

**Percobaan 1: Abstract Class**

Source Code:

Animal.java

public abstract class Animal {  
 public void bernafas(){  
 System.*out*.println("semua binatang bernafas");  
 }  
 public void makan(){  
 System.*out*.println("semua binatang makan");  
 }  
 public abstract void berkembangBiak();  
}

Burung.java

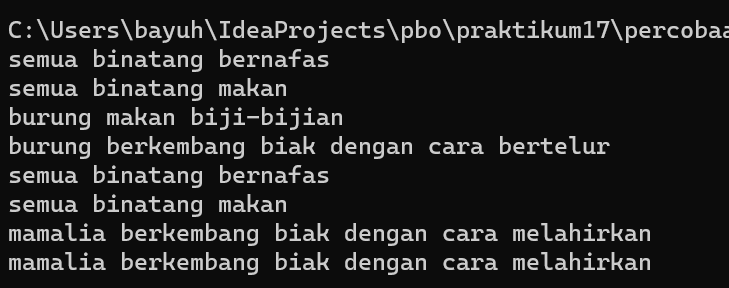
public class Burung extends Animal{  
 public void makan(){  
 super.makan();  
 System.*out*.println("burung makan biji-bijian");  
 }  
 public void berkembangBiak (){  
 System.*out*.println("burung berkembang biak dengan cara bertelur");  
 }  
}

Mamalia.java

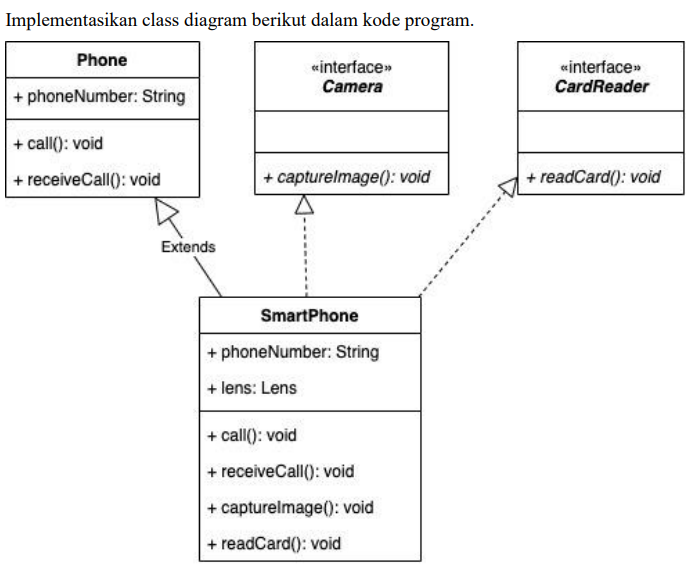
public class Mamalia extends Animal{  
 public void berkembangBiak (){  
 System.*out*.println("mamalia berkembang biak dengan cara melahirkan");  
 }  
}

TestAnimal.java

public class TestAnimal {  
 public static void main(String args[]){  
 Animal lovebird = new Burung();  
 Animal cat = new Mamalia();  
 Mamalia dolphin = new Mamalia();  
 lovebird.bernafas();  
 lovebird.makan();  
 lovebird.berkembangBiak();  
 cat.bernafas();  
 cat.makan();  
 cat.berkembangBiak();  
 dolphin.berkembangBiak();  
 }  
}

Output:  


Percobaan 2: Interface Class



Source Code:

Phone.java

public class Phone {  
 public String phoneNumber;  
 public void call() {  
 System.*out*.println("Calling...");  
 }  
 public void reveiceCall() {  
 System.*out*.println("Receiving...");  
 }  
}

Camera.java

public interface Camera {  
 public void captureImage();  
}

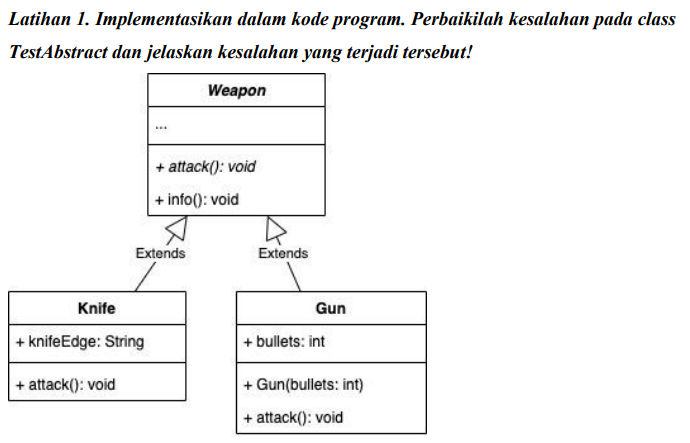
CardReader.java

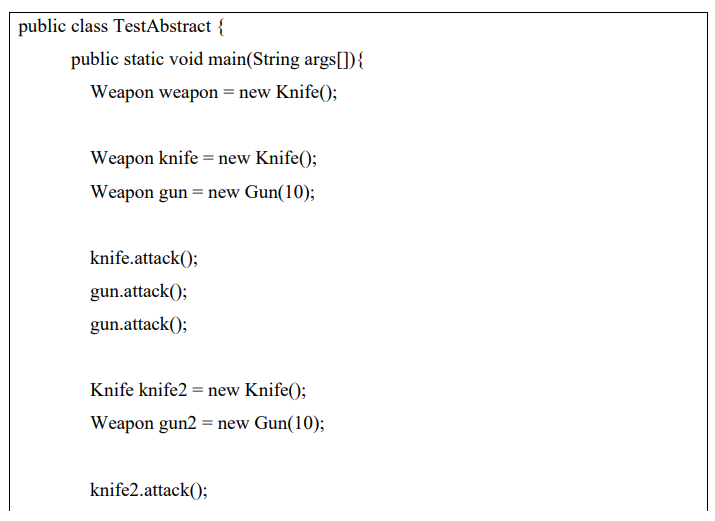
public interface CardReader {  
 public void readCard();  
}

SmartPhone.java

public class SmartPhone extends Phone implements Camera, CardReader {  
 public String phoneNumber;  
 public Lens lens;  
 public void call() {  
 System.*out*.println("Smartphone calling...");  
 }  
 public void reveiceCall() {  
 System.*out*.println("Smartphone receiving...");  
 }  
 public void captureImage() {  
 System.*out*.println("Smartphone capturing Image...");  
 }  
 public void readCard() {  
 System.*out*.println("Smartphone reading card...");  
 }  
}

1. **LATIHAN**







Jawab:

Source Code:

Weapon.java

public abstract class Weapon {  
 public abstract void attack();  
 public void info() {  
 System.*out*.println("Senjata");  
 }  
}

Knife.java

public class Knife extends Weapon{  
 public String knifeEdge;  
 public void attack() {  
 System.*out*.println("Menyayat dan menusuk musuh");  
 }  
}

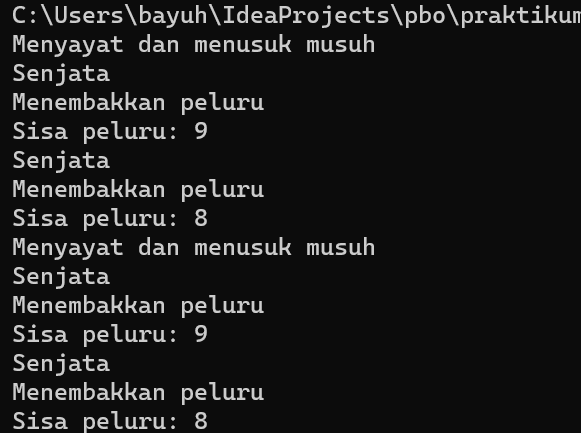
Gun.java

public class Gun extends Weapon{  
 public int bullets = 0;  
 public Gun(int bullets) {  
 this.bullets = bullets;  
 }  
 public void attack() {  
 System.*out*.println("Menembakkan peluru");  
 this.bullets--;  
 System.*out*.println("Sisa peluru: " + this.bullets);  
 }  
}

TestAbstract.java

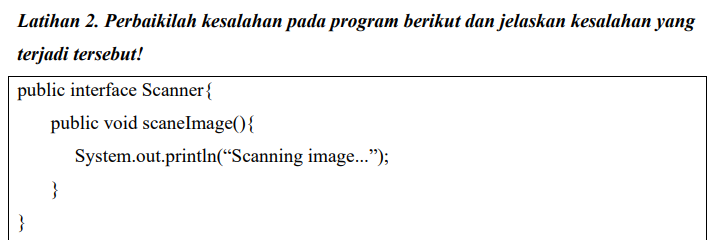
public class TestAbstract {  
 public static void main(String args[]){  
 Weapon weapon = new Knife();  
  
 Weapon knife = new Knife();  
 Weapon gun = new Gun(10);  
  
 knife.attack();  
 knife.info();  
 gun.attack();  
 gun.info();  
 gun.attack();  
  
 Knife knife2 = new Knife();  
 Weapon gun2 = new Gun(10);  
  
 knife2.attack();  
 knife2.info();  
 gun2.attack();  
 gun2.info();  
 gun2.attack();  
  
 }  
}

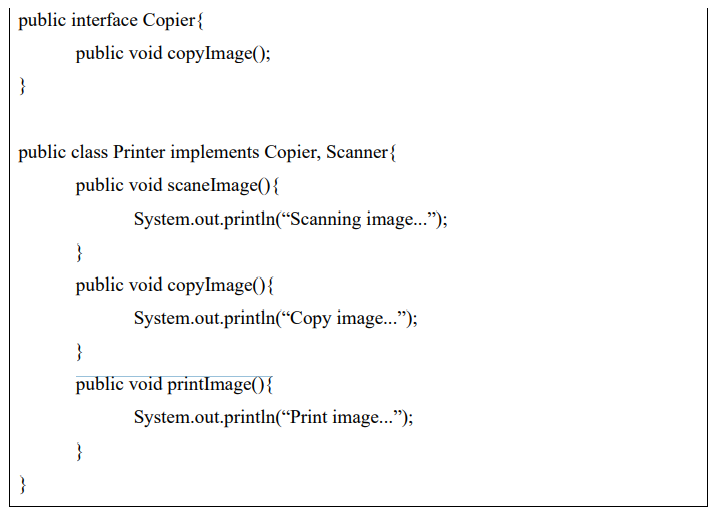
Output:



Penjelasan:

Kesalahan pada TestAbstract.java adalah dengan tidak memanggil method info() pada masing-masing objek. Karena method info() akan menampilkan kata “Senjata”, maka jika ingin menampilkan kata “Senjata” pada output program, maka method itu harus dipanggil. Oleh karena itu, saya memanggil method info() pada masing-masing objek yang dibuat. Method info() ini sendiri berada di class parent, yakni class Weapon dan pada class child nya tidak ada yang melakukan override.





Jawab:

Source Code:

Scanner.java

public interface Scanner {  
 public void scaneImage();  
}

Copier.java

public interface Copier {  
 public void copyImage();  
}

Printer.java

public class Printer implements Copier, Scanner{  
 public void scaneImage(){  
 System.*out*.println("Scanning image...");  
 }  
 public void copyImage(){  
 System.*out*.println("Copy image...");  
 }  
 public void printImage(){  
 System.*out*.println("Print image...");  
 }  
}

TestLatihan2.java

public class TestLatihan2 {  
 public static void main(String[] args) {  
 Printer printer = new Printer();  
 printer.scaneImage();  
 printer.copyImage();  
 printer.printImage();  
 }  
}

Penjelasan:

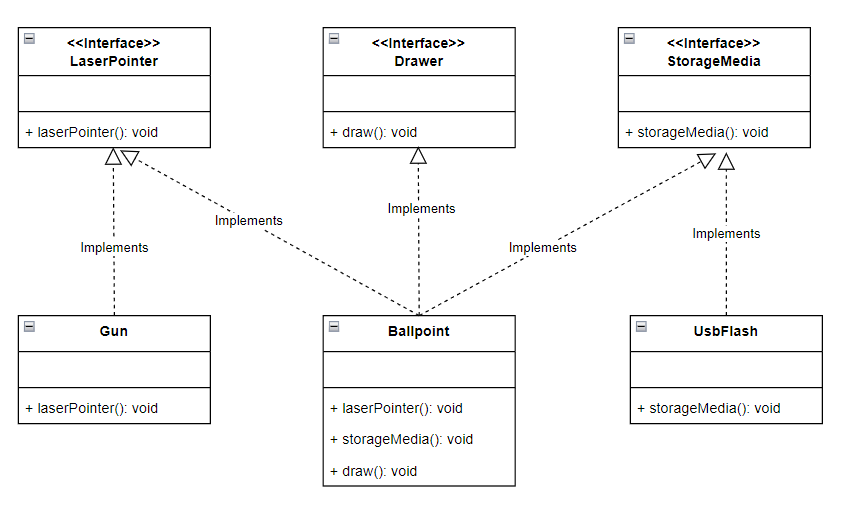
Kesalahan yang ada adalah pada interface Scanner yang pada method scaneImage(), seharusnya tidak diberi body method (harusnya tidak ada isinya, sama seperti pada interface Copier). Karena salah satu aturan pada interface adalah methodnya harus tidak memiliki body. Oleh karena itu, saya memperbaiki kode itu dengan menghapus body method scaneImage() pada interface Scanner.

1. **TUGAS**

**Dalam sebuah program terdapat tiga buah object yaitu : ballpoint, gun, usbFlash. usbFlash memiliki fungsi storageMedia. Gun memiliki fungsi laserPointer. Ballpoint memiliki fungsi draw, laserPointer, dan storageMedia. Buatlah class diagram untuk object-object tersebut kemudian implementasikan menjadi kode program.**

Jawab:

Class Diagram:



Source Code:

Interface LaserPointer

public interface LaserPointer {  
 public void laserPointer();  
}

Interface StorageMedia

public interface StorageMedia {  
 public void storageMedia();  
}

Interface Drawer

public interface Drawer {  
 public void draw();  
}

Class UsbFlash

public class UsbFlash implements StorageMedia {  
 public void storageMedia() {  
 System.*out*.println("UsbFlash storage media accessed...");  
 }  
}

Class Gun

public class Gun implements LaserPointer {  
 public void laserPointer() {  
 System.*out*.println("Gun laser pointer activated...");  
 }  
}

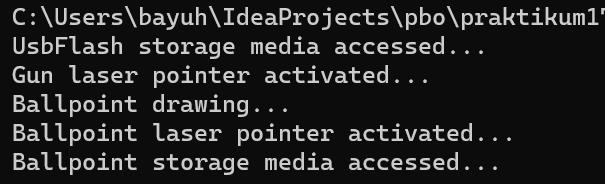
Class Ballpoint

public class Ballpoint implements LaserPointer, StorageMedia, Drawer {  
 public void draw() {  
 System.*out*.println("Ballpoint drawing...");  
 }  
 public void laserPointer() {  
 System.*out*.println("Ballpoint laser pointer activated...");  
 }  
 public void storageMedia() {  
 System.*out*.println("Ballpoint storage media accessed...");  
 }  
}

TestProgram.java

public class TestProgram {  
 public static void main(String[] args) {  
 UsbFlash usbFlash = new UsbFlash();  
 Gun gun = new Gun();  
 Ballpoint ballpoint = new Ballpoint();  
  
 usbFlash.storageMedia();  
 gun.laserPointer();  
 ballpoint.draw();  
 ballpoint.laserPointer();  
 ballpoint.storageMedia();  
 }  
}

Output:



Penjelasan:

Kode di atas menerapkan Interface. Terdapat tiga interface, yaitu LaserPointer, StorageMedia, dan Drawer, masing-masing menyediakan deklarasi metode yang harus diimplementasikan oleh kelas-kelas yang menggunakannya. Kelas UsbFlash mengimplementasikan interface StorageMedia, menyediakan fungsionalitas untuk mengakses media penyimpanan USB. Kelas Gun mengimplementasikan interface LaserPointer, menunjukkan aktivasi laser pointer pada senjata api. Sementara itu, kelas Ballpoint mengimplementasikan interface Drawer, LaserPointer dan StorageMedia, menunjukkan fungsionalitas pena untuk menggambar serta aktivasi laser pointer dan akses media penyimpanan.

Pada bagian pengujian (TestProgram), objek-objek dari ketiga kelas tersebut dibuat dan metode-metode yang didefinisikan dalam antarmuka diakses. Hal ini menunjukkan fleksibilitas dalam penggunaan antarmuka, di mana objek-objek yang berbeda dapat mengimplementasikan fungsionalitas sesuai kebutuhan mereka, tetapi masih dapat dipanggil dan digunakan secara seragam melalui antarmuka yang sesuai.

1. **KESIMPULAN**

Dalam pemrograman berorientasi objek (OOP), interface dan abstract class adalah dua konsep kunci yang memfasilitasi abstraksi dan polimorfisme. Interface menyediakan cara untuk mendeklarasikan metode tanpa memberikan implementasi, memungkinkan kelas-kelas lain untuk mengimplementasikan fungsionalitas sesuai kebutuhan mereka. Sementara itu, abstract class memberikan kerangka kerja untuk kelas-kelas turunannya dengan memungkinkan deklarasi metode tanpa implementasi, serta memberikan kemampuan untuk memberikan implementasi parsial atau lengkap untuk metode tersebut. Keduanya memungkinkan desain yang fleksibel, modular, dan memungkinkan berbagai kelas untuk bersatu dalam konsep polimorfisme, di mana objek dapat dipergunakan dengan cara yang seragam tanpa peduli pada implementasi kelas spesifiknya.