

Nama : Abdul Jawad azizi

Kelas : X RPL 2

No absen : 01

Latihan 1

Screenshot coding

Class Vehicle :

```
package aeroplane;

public abstract class Vehicle {
    public void function(){
        System.out.println("Tools transportation");
    }
    public void fuel(){
        System.out.println("fuels");
    }
    public abstract void walk(); // penulisan method abstract
}
```

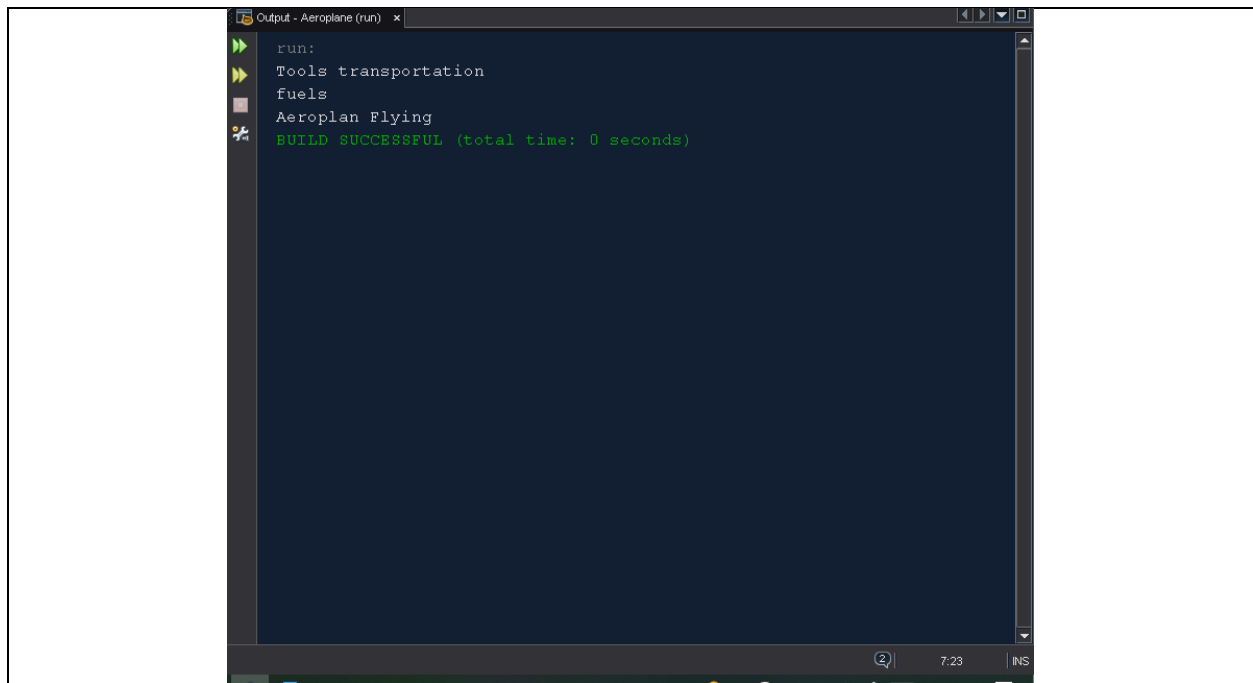
Class Aeroplane :

```
package aeroplane;

public class Aeroplane extends Vehicle {

    @Override
    public void walk(){
        System.out.println("Aeroplan Flying");
    }
    public static void main(String[] args) {
        // TODO code application logic here
        Aeroplane t = new Aeroplane();
        t.function();
        t.fuel();
        t.walk();
    }
}
```

Screenshot output



```
run:
Tools transportation
fuels
Aeroplane Flying
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan :

Kode ini merepresentasikan kelas untuk kendaraan. Kelas Kendaraan adalah kelas abstrak, yang berarti tidak dapat diinstansiasi sendiri, tetapi berfungsi sebagai kelas dasar untuk jenis kendaraan yang lebih spesifik. Kelas ini mendefinisikan dua metode: `function()` dan `fuel()`, yang mencetak beberapa informasi tentang apa yang dilakukan kendaraan.

Kelas Aeroplane adalah subkelas dari Kendaraan, yang berarti bahwa ia mewarisi metode yang didefinisikan di Kendaraan dan juga dapat mendefinisikan metodenya sendiri. Dalam kasus ini, Aeroplane menerima metode `walk()` yang didefinisikan di Vehicle untuk menentukan bahwa pesawat terbang. Pada metode `main()` dari Aeroplane, sebuah instance dari Aeroplane dibuat dan metode-metodenya dipanggil untuk mendemonstrasikan fungsionalitas kelas.

Latihan 2

Screenshot coding

Class Relation :

```
package inter1;

public interface Relation {
    public boolean isGreater(Object a, Object b);
    public boolean isLess (Object a, Object b);
    public boolean isEqual (Object a, Object b);
}
```

Class Line :

```
package inter1;

public class Line implements Relation {

    private double x1;
    private double x2;
    private double y1;
    private double y2;

    public Line (double x1,double x2,double y1,double y2){
        this.x1=x1;
        this.x2=x2;
        this.y1=y1;
        this.y2=y2;
    }
    public double getLength(){
        double length=Math.sqrt((x1-x2)*(x2-x1)+(y2-y1)*(y2-y1));
        return length;
    }
    @Override
    public boolean isGreater(Object a, Object b){
        double aLen=((Line)a).getLength();
        double bLen=((Line)b).getLength();
        return (aLen>bLen);
    }
    @Override
    public boolean isLess(Object a, Object b){
        double aLen=((Line)a).getLength();
        double bLen=((Line)b).getLength();
        return (aLen<bLen);
    }
    @Override
    public boolean isEqual(Object a, Object b){
        double aLen=((Line)a).getLength();
        double bLen=((Line)b).getLength();
        return (aLen==bLen);
    }
}
```

Class Coba :

```
package inter1;

public class coba {
    public static void main(String[] args) {
        Line coba = new Line(2.5,3,4,5);
        Line pertama = new Line(3,4,5,6);
        coba.getLength();
        coba.isGreater(coba, pertama);
        System.out.println(coba.isGreater(coba, pertama));
    }
}
```

Screenshot output

```
run:
true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan :

Kode pertama mendefinisikan sebuah antarmuka bernama "Relation" yang memiliki tiga metode: "isGreater", "isLess", dan "isEqual". Metode-metode ini membandingkan dua objek dan mengembalikan nilai boolean berdasarkan perbandingan.

Kode kedua mendefinisikan kelas bernama "Line" yang mengimplementasikan antarmuka "Relation". Kelas ini memiliki empat variabel privat yang menyimpan koordinat dua titik pada sebuah garis. Kelas ini juga memiliki metode bernama "getLength" yang menghitung panjang garis menggunakan rumus jarak.

Metode "isGreater", "isLess", dan "isEqual" di kelas "Line" membandingkan panjang dua garis dan mengembalikan nilai boolean berdasarkan perbandingan.

Kode ketiga membuat dua objek "Line" dan membandingkannya menggunakan metode "isGreater". Hasilnya dicetak ke konsol.

Latihan 3

Screenshot coding

Class Direktur :

```
package mengtugas1;

import java.util.Scanner;

public class Direktur extends Pegawai {
    private static final int gajiDir=100000;
    private static final int tunjangan=50000;

    Direktur(){

    }
    public Direktur(String nm) {
        super(nm);
    }
    @Override
    public int gaji(){
        return gajiDir;
    }
    public int tunjangan(){
        return tunjangan;
    }

    public void display(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukan nama Direktur :");
        String nama = sc.nextLine();
        setName(nama);
        System.out.println("Nama Direktur : "+nama);
    }
}
```

Class Pegawai :

```

package mengtugas1;

import java.util.Scanner;

public class Pegawai {

    private String nama;
    int gaji;

    Pegawai(){

    }

    Pegawai(String nm){
        this.nama = nm;
        System.out.println("Pegawai");
    }

    public int gaji(){
        return 0;
    }

    public String getNama() {
        return nama;
    }

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

    public void display(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukan Nama anda : ");
        String nama = sc.nextLine();
        setNama(nama);
    }

}

```

Class staff :

```

package mengtugas1;

import java.util.Scanner;

public class Staf extends Pegawai {
    private static final int gajiStaf = 500000;
    private static final int bonusStaf = 100000;

    @Override
    public int gaji(){
        return gajiStaf;
    }

    public int bonus(){
        return bonusStaf;
    }

    @Override
    public void display(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukan nama Staf :");
        String nama = sc.nextLine();
        setNama(nama);
        System.out.println("Nama Staf : "+nama);
    }

}

```

Class PembayaranGaji :

```
package mengtugas1;

public class pembayaranGaji {

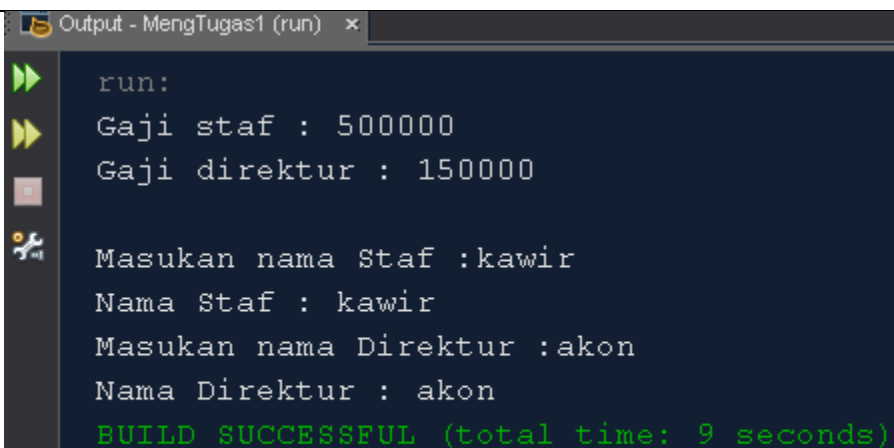
    public int hitungGaji(Pegawai peg){
        int uang = peg.gaji();
        if (peg instanceof Direktur) {
            uang += ((Direktur)peg).tunjangan();
            if (peg instanceof Staf) {
                uang += ((Staf)peg).bonus();
            }
        }
        return uang;
    }

    public static void main(String[] args) {
        pembayaranGaji pg = new pembayaranGaji();
        Staf s = new Staf();
        Direktur d = new Direktur();

        System.out.println("Gaji staf : "+pg.hitungGaji(s));
        System.out.println("Gaji direktur : "+pg.hitungGaji(d));

        System.out.println("");
        s.display();
        d.display();
    }
}
```

Screenshot output



```
run:
Gaji staf : 500000
Gaji direktur : 150000

Masukan nama Staf :kawir
Nama Staf : kawir
Masukan nama Direktur :akon
Nama Direktur : akon
BUILD SUCCESSFUL (total time: 9 seconds)
```

Penjelasan :

Kode ini merupakan contoh implementasi pewarisan dan polimorfisme di Java dengan menggunakan kelas dan antarmuka. Kode ini terdiri dari empat kelas: Pegawai, Direktur, Staf, dan pembayaranGaji.

Kelas Pegawai adalah kelas abstrak yang mendefinisikan perilaku umum untuk semua pegawai, dan diperluas oleh kelas Direktur dan Staf. Kelas Direktur dan Staf mewarisi properti dan metode dari kelas Pegawai, dan mereka mengimplementasikan metode gaji abstrak dengan cara mereka sendiri.

Kelas PembayaranGaji menghitung gaji seorang pegawai, dengan mempertimbangkan bonus atau tunjangan yang mungkin mereka dapatkan berdasarkan posisi mereka. Hal ini dilakukan dengan menggunakan polimorfisme dengan mengoper sebuah instance dari kelas Pegawai ke metode hitungGaji dan memeriksa tipenya untuk menentukan apakah bonus atau tunjangan akan ditambahkan.

Metode utama dari kelas pembayaranGaji membuat instance dari kelas Staf dan Direktur, dan menggunakan metode hitungGaji untuk menghitung gaji mereka. Terakhir, ia memanggil metode tampilkan dari setiap kelas untuk menampilkan nama karyawan.

Praktikum

Screenshot coding

Class Rect :

```
public class Rect {  
  
    public int x1,x2,x3,x4,y1,y2;  
  
    public Rect(int x1, int x2, int x3, int x4) {  
        this.x1 = x1;  
        this.x2 = x2;  
        this.x3 = x3;  
        this.x4 = x4;  
    }  
    public Rect(int w, int h){  
        this(0,0, w, h);  
    }  
    public Rect(){  
        this(0,0,0,0);  
    }  
    public void move (int deltax,int deltay){  
        x1 += deltax; x2 += deltax;  
        y1 += deltay; y2 += deltay;  
    }  
    public boolean isInside(int x, int y){  
        return ((x >= x1) && (x <=x2) && (y >= y1) && (y<= y2));  
    }  
    public Rect union(Rect r){  
        return new Rect(  
            (this.x1 < r.x1) ? this.x1 : r.x1,  
            (this.y1 <r.y1) ? this.y1 : r.y1,  
            (this.x2 >r.x2) ? this.x2 : r.x2,  
            (this.y2 > r.y2) ? this.y2 : r.y2  
        );  
    }  
    public Rect intersection (Rect r){  
        Rect result = new Rect(  
            (this.x1 < r.x1) ? this.x1 : r.x1,  
            (this.y1 <r.y1) ? this.y1 : r.y1,  
            (this.x2 >r.x2) ? this.x2 : r.x2,  
            (this.y2 > r.y2) ? this.y2 : r.y2  
        );  
        if (result.x1 > result.x2) {  
            result.x1 = result.x2 = 0;  
        }  
        if (result.y1 > result.y2) {  
            result.y1 = result.y2 = 0;  
        }  
        return result;  
    }  
    @Override  
    public String toString() {  
        return "[" + this.x1 + "," + this.y1 + "; " + this.x2 + "," + this.y2 + "];"  
    }  
}
```

Main Class :

```
public class Main {

    public static void main(String[] args) {
        Rect r1 = new Rect(1, 1, 4, 4);
        Rect r2 = new Rect(2, 3, 5, 6);

        Rect union = r1.union(r2);
        Rect intersection = r1.intersection(r2);

        System.out.println(r1 + " union " + r2 + " = " + union);
        System.out.println(r1 + " intersect " + r2 + " = " + intersection);
    }

}
```

Screenshot output

```
Output - praktikum polimorfisme (run) x
run:
[1,0; 1,0] union [2,0; 3,0] = [1,0; 0,0]
[1,0; 1,0] intersect [2,0; 3,0] = [0,0; 0,0]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan :

Kode ini mendefinisikan kelas Rect yang merepresentasikan persegi panjang dengan koordinat dua dimensi. Kelas Rect memiliki beberapa metode yang beroperasi pada persegi panjang:

- Konstruktor Rect(int, int, int, int) membuat objek Rect baru dengan koordinat yang ditentukan.
 - Konstruktor Rect(int, int) membuat objek Rect baru dengan lebar dan tinggi yang ditentukan.
 - Konstruktor Rect() membuat objek Rect baru dengan semua koordinat yang disetel ke 0.
- Metode move(int, int) memindahkan persegi panjang dengan nilai delta yang ditentukan.

- Metod `isInside(int, int)` memeriksa apakah titik yang ditentukan berada di dalam persegi panjang.
- Metod `union(Rect)` menghitung gabungan dari persegi panjang ini dan persegi panjang lain, dan mengembalikan objek `Rect` baru yang merepresentasikan persegi panjang yang dihasilkan.
- Metod `intersection(Rect)` menghitung perpotongan persegi panjang ini dan persegi panjang lain, dan mengembalikan objek `Rect` baru yang mewakili persegi panjang yang dihasilkan.
- Metod `toString()` mengembalikan representasi string dari persegi panjang.

Kelas `Main` adalah kelas uji coba yang membuat dua objek `Rect`, menghitung penyatuan dan perpotongan menggunakan metode `union(Rect)` dan `intersection(Rect)`, dan mencetak hasilnya ke konsol menggunakan metode `toString()`.