Nama : Dhimas Luthfi Arnanda

NIM : 24060123120040

Lab : PBO B2

Hari/Tanggal : Selasa / 22 April 2025

POST TEST PRAKTIKUM PBO MINGGU 6

b. Polimorfisme Ad Hoc: Coercion

No	Program	System.out.println (output);	Apakah Berhasil Dieksekusi Atau Gagal? Catatlah Penjelasan Terkait Pemahaman tentang Casting dan Coercion
1	int output = ‘a’;	97	Berhasil. karakter 'a' secara langsung dikonversi ke ASCII (97). coercion dari char - int.
2	double x = 15.5; int output = x;		Error (type incompatible). Tidak bisa melakukan coercion dari double - int karena punya resiko kehilangan data . Harus dilakukan casting (int) output = (int) x;.
3	int y = 25; double output = y;	25.0	Berhasil. Terjadi coercion dari int - double secara otomatis, karena double dapat menyimpan nilai int tanpa kehilangan data.
4	int z = 78; char output = (char) z;	N	Berhasil. Terjadi casting dari int - char, menghasilkan karakter Unicode pada posisi 78 'N'.
5	char a = ‘a’; double output = a;	97.0	Berhasil. Terjadi coercion dari char - double. Nilai 'a' dikonversi ke ASCII (97.0).

e. Post Test

Program :

```java
public class Pegawai {
    /* Attribut */
    protected String nama;
    protected static int gajiPokok = 5000000;
    /* Method */
    public Pegawai() {

    }
    public Pegawai(String nama) {
        this.nama = nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public void tampilData() {
        System.out.println("Nama : " + nama + ", Gaji Pokok : " + gajiPokok);
    }
}
```

```java
public class Manajer extends Pegawai {
    /* Attribut */
    private static int tunjangan = 700000;
    /* Method */
    public Manajer() {

    }
    public Manajer(String nama) {
        super(nama);
    }
    @Override
    public void tampilData() {
        super.tampilData();
        System.out.println("Tunjangan : " + tunjangan);
    }
}
```

```java
public class Programmer extends Pegawai{
    /* Attribut */
    private static int Bonus = 450000;
    /* Method */
    public Programmer() {

    }
    public Programmer(String nama) {
        super(nama);
    }
    @Override
    public void tampilData() {
        super.tampilData();
        System.out.println("Bonus : " + Bonus);
    }
}
```

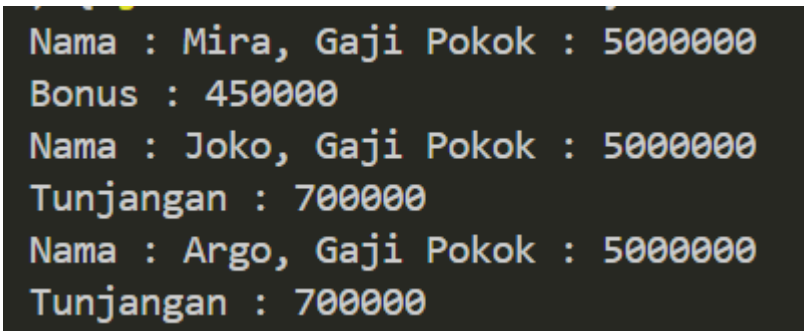
```java
import java.util.ArrayList;

public class TestPolimorfisme {
    public static void main(String[] args) {
        Pegawai pegawai = new Programmer("Mira");
        Pegawai pegawai2 = new Manajer("Joko");
        Manajer pegawai3 = new Manajer("Argo");

        ArrayList<Pegawai> emps = new ArrayList<>();
        emps.add(pegawai);
        emps.add(pegawai2);
        emps.add(pegawai3);

        for (Pegawai emp : emps) {
            emp.tampilData();
        }
    }
}
```

Output :



1. Jelaskan manfaat polimorfisme pada kasus ini.

   Mnfaat polimorfisme dalam program ini membuat code jadi fleksibel dan efisien. objek dari kelas turunan Pegawai yaitu Programmer sama Manajer dapat diperlakukan sebagai objek dari kelas induk Pegawai, jadi bisa menyimpan berbagai jenis pegawai dalam satu ArrayList<Pegawai>. Karena bisa disimpan ke satu ArrayList<Pegawai> maka cukup sekali perulangan saja untuk proses pemanggilan method tampilData().

2. Apabila pada main program perlu menambahkan pegawai4 dan pegawai5! Apa permasalahan yang muncul jika diterapkan tanpa polimorfisme (inclusion)?

   karena penambahan objek dilakukan tanpa polimorfisme (inclusion), maka masing masing kelas turunannya harus membuat ArrayList<> dan perulangan pemanggilan tampilData() yang terpisah untuk tiap jenis class. code pasti lebih panjang, terutama kalau jumlah tipe pegawai bertambah. Jadinya program tidak fleksibel dan efisien.