

NAMA : Haniifan Thoriqul Ibad

NIM : 225150407111089

KELAS : Sistem Informasi B

BAB : 6 Inheritance

ASISTEN : Adin Rama Ariyanto Putran dan Fahru Setiawan Iskandar

1. Data dan Analisis hasil percobaan

1. Jalankan code program diatas dan benahi jika menemukan kesalahan! **Jawab :**

Tidak terdapat kesalahan penulisan syntax pada source code tersebut.

2. Bagaimana cara konstruktor pada subclass memanggil konstruktor di superclass nya? Apakah hal itu perlu dilakukan? Sertakan alasan anda !

Jawab :

Untuk memanggil konstruktor di superclass dari subclass, dapat menggunakan `super()` di dalam konstruktor subclass. Pemanggilan konstruktor superclass di konstruktor subclass diperlukan apabila kita ingin menggunakan atribut atau method yang telah didefinisikan di superclass. Dengan menggunakan konstruktor superclass, kita dapat memastikan bahwa atribut yang didefinisikan di superclass terinisialisasi dengan benar sebelum di subclass.

3. Tambahkan constructor pada class Employee dengan parameter String name! amati perubahan apa yang terjadi, jelaskan jawaban anda!

Jawab :

Penambahan konstruktor dengan parameter String name :

```
public Employee(String name) {  
    this.name = name;  
    this.salary = 0;  
    this.hireday = new Date();  
}
```

Setelah penambahan konstruktor, output akan mengalami perubahan dengan dapat menampilkan nama namun tidak dapat menampilkan jumlah gaji. Dengan adanya konstruktor baru ini, kita tidak perlu lagi memasukkan parameter yang tidak diperlukan seperti gaji, tahun, bulan, dan hari saat membuat objek Karyawan.

4. Pada Class Manager baris ke 5, setelah variable day tambahkan variable bonus! Amati apa yang terjadi dan mengapa demikian?

Jawab :

```
super(name, salary, year, month, day, bonus);  
bonus = 0;
```

Perubahan parameter dari konstruktor tersebut terjadi suatu error pada kode program. Kaarena nilai bonus yang diberikan pada konstruktor Manager daimbil dari variable bonus, yang pada saat konstruktor dipanggil masih memiliki nilai default yaitu 0. Oleh karena itu, ketika metode getSalary() dipanggil, nilai bonus yang diitung akan selalu 0, karena tidak pernah diberikan nilai baru melalui setter setBonus().

5. Untuk apa digunakan keyword this pada class manager dan employee? Hapus keyword this dan amati apa yang terjadi?

Jawab :

Keyword this digunakan untuk merujuk ke variabel atau metode pada objek saat ini. Pada kelas Employee, this digunakan untuk merujuk ke variabel name, salary, dan hireday pada objek saat ini. Pada kelas Manager, this tidak digunakan karena tidak ada variabel yang memiliki nama yang sama dengan parameter atau variabel lokal di dalam metode. Menghapus keyword this pada kelas Employee dan Manager tidak akanberpengaruh signifikan terhadap output program, tetapi dapat membuat kode menjadi kurang jelas dan sulit dibaca atau kurang terstruktur.

6. Tambahkan constructor pada class Employee dengan parameter Bertipe data string bernama name yang nantinya bila constructor ini akan dipanggil akan menginisialisasi variable name! Amati perubahannya pada class anak dan jelaskan! Benahi bila terjadi kesalahan!

Jawab :

```
public Employee(String name) {  
    this.name = name;
```

```
this.salary = 0;  
this.hireday = new Date();
```

Laporan Praktikum Pemrograman Lanjut SI 2



**LABORATORIUM PEMBELAJARAN
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA**

Dengan menambahkan constructor pada kelas Employee yang memiliki parameter bertipe data String bernama name, maka kelas anak (Manager) juga harus disesuaikan dengan menambahkan parameter name pada constructor-nya. Hal ini karena kelas anak akan memanggil constructor dari kelas induknya menggunakan keyword "super" dan harus mengirimkan argumen yang diperlukan oleh constructor tersebut. Penambahan di kelas manager

7. Pada bab sebelumnya anda telah belajar mengenai konsep encapsulation, jelaskan mengapa pada super class menggunakan modifier protected? Apa yang terjadi jika modifier anda ubah menjadi private atau public? Jelaskan !

Jawab :

Untuk memberikan akses ke field atau method pada kelas turunan, modifier protected digunakan pada super class. Dengan begitu, kelas turunan bisa mengakses field atau method yang terproteksi tersebut tanpa memerlukan keyword super. Jika modifier diubah menjadi private, maka field atau method tersebut hanya bisa diakses oleh kelas itu sendiri dan tidak bisa diakses oleh kelas turunannya. Namun, jika modifier diganti menjadi public, maka field atau method tersebut dapat diakses oleh kelas mana pun, bahkan dari luar package, yang dapat menimbulkan masalah keamanan dan penggunaan yang tidak semestinya. Oleh karena itu, modifier protected digunakan sebagai salah satu cara untuk mengimplementasikan encapsulation pada program.

8. Ubahlah acces modifier method pada kelas employee menjadi :

- a. Private
- b. Protected

Amati perubahan apa yang terjadi? Jelaskan jawaban anda dengan detail!

Jawab :

- a. Apabila access modifier method pada kelas Employee diubah menjadi private, method tersebut hanya dapat diakses oleh kelas itu sendiri dan tidak dapat diakses oleh kelas

turunan maupun kelas lain di luar package. Hal ini dapat menimbulkan masalah pada pengembangan program yang lebih kompleks karena kelas turunan atau kelas lain tidak dapat mengakses atau memodifikasi data pada kelas Employee.

Laporan Praktikum Pemrograman Lanjut SI 3



LABORATORIUM PEMBELAJARAN
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

b. Apabila access modifier method pada kelas Employee diubah menjadi protected, maka method tersebut bisa diakses oleh kelas turunan, tetapi tidak bisa diakses oleh kelas di luar package. Dalam hal ini, kelas turunan dapat mengakses dan mengubah data pada kelas Employee tanpa mengkhawatirkan akses oleh kelas di luar package yang mungkin dapat mengganggu keamanan data. Selain itu, kelas turunan juga tidak perlu menggunakan keyword super untuk mengakses method tersebut.

2. Tugas Praktikum

2.1 Source code

Kelas Manusia :

```
public class Manusia {  
    private String nama;  
    private boolean jenisKelamin;  
    private String nik;  
    private boolean menikah;  
  
    public Manusia(String nama, boolean jenisKelamin, String  
    nik, boolean menikah) {  
        this.nama = nama;  
        this.jenisKelamin = jenisKelamin;  
        this.nik = nik;  
        this.menikah = menikah;  
    }  
  
    public String getNama() {  
        return nama;  
    }  
  
    public void setNama(String nama) {  
        this.nama = nama;  
    }  
  
    public boolean isJenisKelamin() {  
        return jenisKelamin;  
    }  
  
    public void setJenisKelamin(boolean jenisKelamin) {  
        this.jenisKelamin = jenisKelamin;  
    }  
  
    public String getNik() {  
        return nik;  
    }  
  
    public void setNik(String nik) {  
        this.nik = nik;  
    }  
  
    public boolean isMenikah() {
```

Laporan Praktikum Pemrograman Lanjut SI 4



LABORATORIUM PEMBELAJARAN
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```

return menikah;
}

public void setMenikah(boolean menikah) {
    this.menikah = menikah;
}

public double getTunjangan() {
    if (menikah) {
        if (jenisKelamin) {
            return 25.0;
        } else {
            return 20.0;
        }
    } else {
        return 15;
    }
}

public double getPendapatan() {
    return getTunjangan();
}

public String toString() {
    String jk = (jenisKelamin) ? "Laki-laki" : "Perempuan";
    return "Nama: " + nama + "\n" + "NIK: " + nik + "\n" + "Jenis
Kelamin: " + jk + "\n" +
    "Pendapatan: $" + getPendapatan();
}
}

```

Kelas MahasiswaFILKOM :

```

public class MahasiswaFILKOM extends Manusia {
    private String nim;
    private double ipk;

    public MahasiswaFILKOM(String nama, boolean jenisKelamin,
String nik, boolean menikah, String nim, double ipk) {
        super(nama, jenisKelamin, nik, menikah);
        this.nim = nim;
        this.ipk = ipk;
    }

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public double getIpk() {
        return ipk;
    }

    public void setIpk(double ipk) {
        this.ipk = ipk;
    }
}

```

**FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA**

```
}

public String getStatus() {
    String prodi;
    int angkatan = Integer.parseInt(nim.substring(0, 2)); int
    kodeProdi = Integer.parseInt(nim.substring(6, 7)); switch
    (kodeProdi) {
        case 2:
            prodi = "Teknik Informatika";
            break;
        case 3:
            prodi = "Teknik Komputer";
            break;
        case 4:
            prodi = "Sistem Informasi";
            break;
        case 6:
            prodi = "Pendidikan Teknologi Informasi"; break;
        case 7:
            prodi = "Teknologi Informasi";
            break;
        default:
            prodi = "Tidak Diketahui";
    }
    return prodi + ", " + "20" + angkatan;
}

public double getBeasiswa() {
    double beasiswa = 0.0;
    if (ipk < 3.0) {
        return 0;
    } else if (ipk >= 3.0 && ipk <= 3.5) {
        return 50;
    } else {
        return 75;
    }
}

@Override
public String toString() {
    return super.toString() +
        "\nNIM: " + nim +
        "\nIPK: " + ipk +
        "\nStatus: " + getStatus();
}
}

Kelas Pekerja :
import java.time.LocalDate;
import java.time.Period;

public class Pekerja extends Manusia {
    private double gaji;
    private LocalDate tahunMasuk;
    private int jumlahAnak;
}
```

```
public Pekerja(String nama, boolean jenisKelamin, String
nik, boolean menikah, double gaji, LocalDate tahunMasuk, int
jumlahAnak) {
    super(nama, jenisKelamin, nik, menikah);
    this.gaji = gaji;
    this.tahunMasuk = tahunMasuk;
    this.jumlahAnak = jumlahAnak;
}

public double getGaji() {
    return gaji;
}

public void setGaji(double gaji) {
    this.gaji = gaji;
}

public LocalDate getTahunMasuk() {
    return tahunMasuk;
}

public void setTahunMasuk(LocalDate tahunMasuk) {
    this.tahunMasuk = tahunMasuk;
}

public int getJumlahAnak() {
    return jumlahAnak;
}

public void setJumlahAnak(int jumlahAnak) {
    this.jumlahAnak = jumlahAnak;
}

public double getBonus() {
    int lamaBekerja = LocalDate.now().getYear() -
tahunMasuk.getYear();
    if (lamaBekerja <= 5) { return 0.05 * gaji; } else if
(lamaBekerja <= 10) { return 0.1 * gaji; } else {
return 0.15 * gaji;
}
}

@Override
public double getTunjangan() {
    return super.getTunjangan() + (20.0 * jumlahAnak);
}

@Override
public double getPendapatan() {
    return gaji + getBonus() + getTunjangan();
}

@Override
public String toString() {
    return super.toString() + "\n" +
```



```
"Tahun Masuk: " + tahunMasuk + "\n" + "Jumlah Anak: " +  
jumlahAnak + "\n" + "Gaji: " + gaji + "\n" + "Bonus: " +  
getBonus() + "\n" +  
"Total Pendapatan: " + getPendapatan();  
}
```

Kelas Manager :

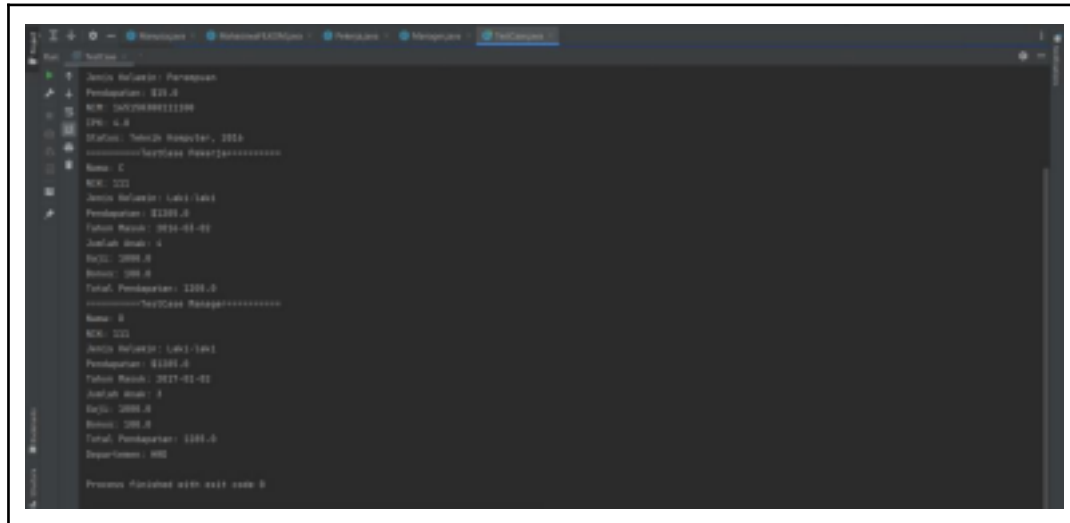
```
import java.time.LocalDate;  
  
public class Manager extends Pekerja {  
    private String departemen;  
  
    public Manager(String nama, boolean jenisKelamin, String  
    nik, boolean menikah,  
    double gaji, LocalDate tahunMasuk, int jumlahAnak,  
    String departemen) {  
  
        super(nama, jenisKelamin, nik, menikah, gaji,  
        tahunMasuk, jumlahAnak);  
        this.departemen = departemen;  
    }  
  
    public String getDepartemen() {  
        return departemen;  
    }  
  
    public void setDepartemen(String departemen) {  
        this.departemen = departemen;  
    }  
  
    @Override  
    public double getTunjangan() {  
        return super.getTunjangan() + (0.1 * getGaji());  
    }  
  
    @Override  
    public String toString() {  
        return super.toString() +  
        "\nDepartemen: " + departemen;  
    }  
}
```

Kelas TestCase :

```
import java.time.LocalDate;  
  
public class TestCase {  
    public static void main(String[] args) {  
        System.out.println("=====TestCase  
Manusia=====");  
        Manusia a = new Manusia("A", true, "111", true);  
        System.out.println(a);  
        System.out.println("=====TestCase  
MahasiswaFilkom=====");  
        MahasiswaFILKOM b = new MahasiswaFILKOM("B", false,  
        "111", false, "165150300111100", 4.0);  
    }  
}
```

```
System.out.println(b);  
System.out.println("=====Test Case  
Pekerja=====");  
Pekerja c = new Pekerja("C", true, "111", true, 1000,  
LocalDate.of(2016, 3, 2), 4);  
System.out.println(c);  
System.out.println("=====Test Case  
Manager=====");  
Manager d = new Manager("D", true, "111", true, 1000,  
LocalDate.of(2017, 1, 2), 3, "HRD");  
System.out.println(d);  
}  
}
```

2.2 Screenshot hasil



```
Run: TestCases  
+ JavaSE-8: Parametrisasi  
+ Parametrisasi: $10.0  
+ NIM: 1002000011100  
+ CPU: 4.0  
+ Status: Tested: Passed, 2018  
+ =====Test Case Pekerja=====  
+ Name: C  
+ NIM: 111  
+ JavaSE-8: Lab1-Lab1  
+ Parametrisasi: $1000.0  
+ Tahun Masuk: 2016-03-02  
+ Jumlah Revisi: 4  
+ Revisi: 1000.0  
+ Bonus: 100.0  
+ Total Parametrisasi: $1000.0  
+ =====Test Case Manager=====  
+ Name: D  
+ NIM: 111  
+ JavaSE-8: Lab1-Lab1  
+ Parametrisasi: $1000.0  
+ Tahun Masuk: 2017-01-02  
+ Jumlah Revisi: 3  
+ Revisi: 1000.0  
+ Bonus: 100.0  
+ Total Parametrisasi: $1000.0  
+ Departemen: HRD  
+ Process finished with exit code 0
```