

NAMA : Lisa Hanifatul Khasanah
NIM : 225150401111038
KELAS : B
BAB : 6
ASISTEN : Adin Rama Ariyanto Putra dan Fahru Setiawan Iskandar

1. Data dan Analisis hasil percobaan

1.) Jalankan code program diatas dan benahi jika menemukan kesalahan!

Tidak ada kesalahan penulisan sintaks pada kode tersebut, namun penempatan kode system out di bawah ini seharusnya disambung menjadi satu line.

```
8      System.out.println("nama boss : "+boss.getName()+",  
9      salary = "+boss.getSalary());  
10     System.out.println("nama staff : "+staff.getName()+",  
11     salary = "+staff.getSalary());  
12     }  
13 }
```

Berikut pembenarannya,

```
System.out.println("nama boss : "+boss.getName()+", salary = "+boss.getSalary());  
System.out.println("nama staff : "+staff.getName()+", salary = "+staff.getSalary());  
}  
}
```

2.) Bagaimana cara konstruktor pada subclass memanggil konstruktor di superclass nya?

Apakah hal itu perlu dilakukan? Sertakan alasan anda !

Jawab :

Untuk mengakses konstruktor superclass dari subclass, gunakan 'super' dengan parameter sesuai. Contohnya, dalam konstruktor Manager, konstruktor Employee dipanggil dengan kata kunci 'super' dan parameter name, salary, year, month, dan day. Ini diperlukan karena inisialisasi variabel tidak bisa diakses langsung oleh subclass dan subclass dapat mewarisi atribut dan metode dari superclass. Jika konstruktor superclass tidak dipanggil dalam konstruktor subclass, Java akan memanggil konstruktor superclass tanpa argumen, yang dapat menyebabkan kesalahan.

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

Jawab :

Berikut penamnanan konstruktor dengan parameter String name :

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

Kemudian ditambahkan juga objek baru untuk mengakses konstruktor baru tersebut, yaitu seperti berikut :

```
Employee staff2 = new Employee("John");  
System.out.println("nama staff : "+staff2.getName()+" , salary =  
"+staff2.getSalary());
```

Setelah penambahan konstruktor outputnya akan mengalami perubahan yaitu dapat menampilkan nama tetapi 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. Sebagai contoh, jika ingin membuat objek Karyawan dengan nama "John", kita hanya perlu memanggil konstruktor baru ini dengan satu parameter: `Karyawan john = new Karyawan("John");`. Hal ini akan memudahkan dalam pembuatan objek dan membuat kode lebih mudah dibaca.

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

Jawab :

Jika parameter dari konstruktor super diubah menjadi,

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

Maka akan terjadi error pada kode program, karena nilai bonus yang diberikan pada konstruktor Manager diambil dari variabel bonus, yang pada saat konstruktor dipanggil masih memiliki nilai default yaitu 0. Oleh karena itu, ketika metode `getSalary()` dipanggil, nilai bonus yang dihitung akan selalu 0, karena tidak pernah diberikan nilai baru melalui setter `setBonus()`. Untuk memperbaiki hal ini, kita harus menghapus `bonus = 0;` pada baris ke-6 dan memberikan nilai awal pada variabel bonus melalui parameter pada konstruktor, atau menggunakan setter `setBonus()` untuk memberikan nilai bonus pada objek Manager.

- 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 akan berpengaruh 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 :

Konstruktor yang ditambahkan,

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

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,

```
public Manager(String name) {  
    super(name);  
    bonus = 0;  
}
```

- 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.
- 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; // true = laki-laki, false =  
    perempuan  
    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() {  
        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 15.0;  
    }  
}  
  
public double getPendapatan() {  
    return getTunjangan();  
}  
  
@Override  
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 void setNim(String nim) {  
        this.nim = nim;  
    }  
  
    public String getNim() {  
        return nim;  
    }  
  
    public void setIpk(double ipk) {  
        this.ipk = ipk;  
    }  
  
    public double getIpk() {  
        return ipk;  
    }  
  
    public String getStatus() {  
        String prodi = "";  
        String angkatan = "";  
  
        int digit1 = Integer.parseInt(nim.substring(0, 2));  
        int digit7 = Integer.parseInt(nim.substring(6, 7));  
  
        switch (digit7) {  
            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";  
    }  
  
    angkatan = "20" + Integer.toString(digit1);  
  
    return prodi + ", " + angkatan;  
}  
  
public double getBeasiswa() {  
    double beasiswa = 0.0;  
  
    if (ipk >= 3.0 && ipk <= 3.5) {  
        beasiswa = 50.0;  
    } else if (ipk > 3.5 && ipk <= 4.0) {  
        beasiswa = 75.0;  
    }  
  
    return beasiswa;  
}  
  
public String toString() {  
    return super.toString() + "\nNIM: " + nim + "\nIPK: " +  
ipk + "\nStatus: " + getStatus();  
}  
}
```

Kelas Pekerja :

```
import java.time.LocalDate;  
  
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;
    }
}
```

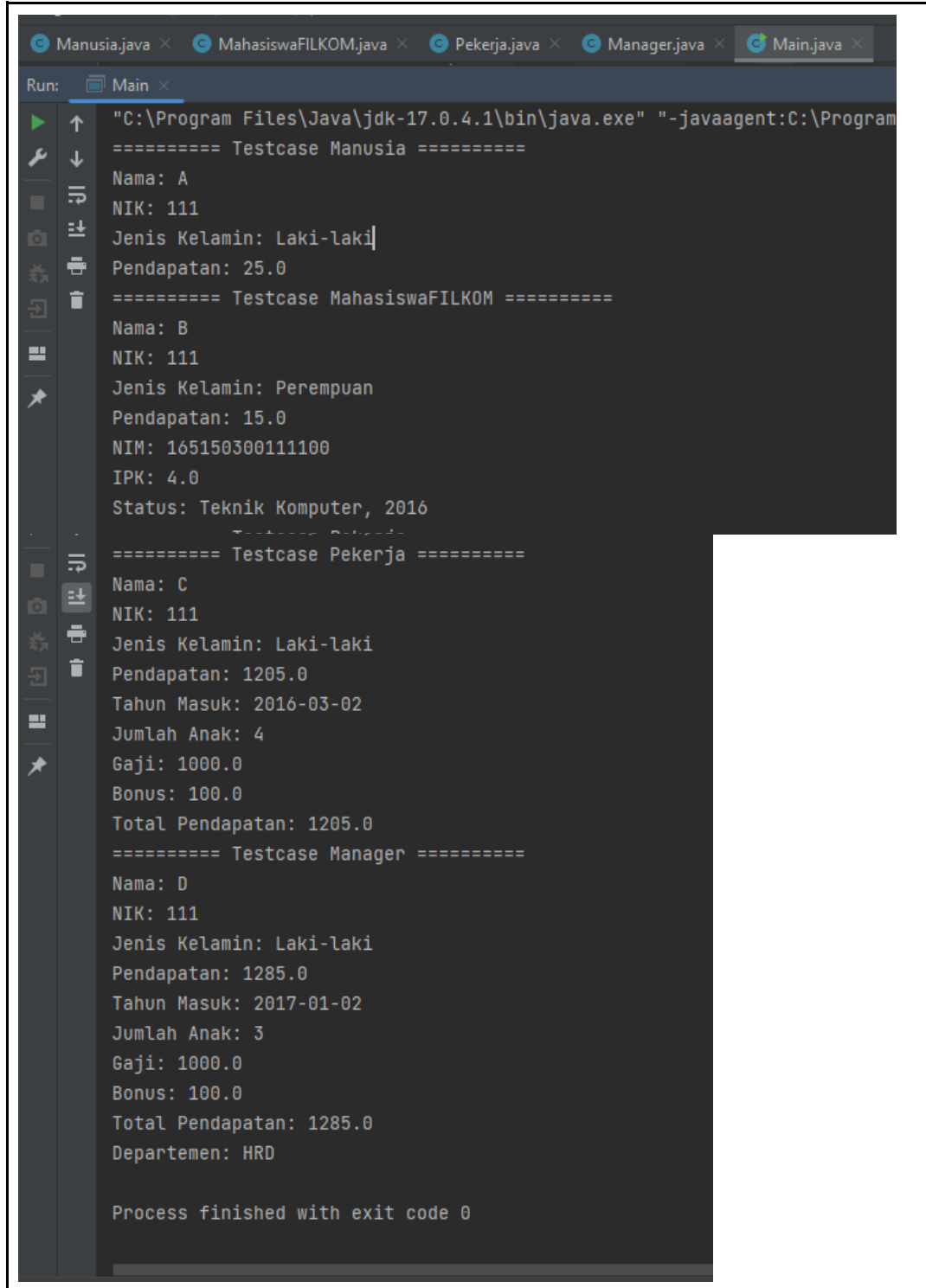
Kelas Main :

```
import java.time.LocalDate;

public class Main {
    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("===== Testcase Pekerja
        =====");
        Pekerja c = new Pekerja("C", true, "111", true, 1000,
    LocalDate.of(2016, 3, 2), 4);
        System.out.println(c);
        System.out.println("===== Testcase 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: Main x  
"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:C:\Program  
===== Testcase Manusia =====  
Nama: A  
NIK: 111  
Jenis Kelamin: Laki-laki  
Pendapatan: 25.0  
===== Testcase MahasiswaFILKOM =====  
Nama: B  
NIK: 111  
Jenis Kelamin: Perempuan  
Pendapatan: 15.0  
NIM: 165150300111100  
IPK: 4.0  
Status: Teknik Komputer, 2016  
===== Testcase Pekerja =====  
Nama: C  
NIK: 111  
Jenis Kelamin: Laki-laki  
Pendapatan: 1205.0  
Tahun Masuk: 2016-03-02  
Jumlah Anak: 4  
Gaji: 1000.0  
Bonus: 100.0  
Total Pendapatan: 1205.0  
===== Testcase Manager =====  
Nama: D  
NIK: 111  
Jenis Kelamin: Laki-laki  
Pendapatan: 1285.0  
Tahun Masuk: 2017-01-02  
Jumlah Anak: 3  
Gaji: 1000.0  
Bonus: 100.0  
Total Pendapatan: 1285.0  
Departemen: HRD  
  
Process finished with exit code 0
```