

---

NAMA : Lisa Hanifatul Khasanah  
NIM : 225150401111038  
KELAS : B  
BAB : 7  
ASISTEN : Adin Rama Ariyanto Putra dan Fahru Setiawan Iskandar

---

## **1. Data dan Analisis hasil percobaan**

- 1.) Jalankan Main.java untuk polimorfisme Employee, analisis dan jelaskan keluaran program tersebut!

Jawab:

Jika menjalankan kode `Employee employee = new Employee();`, akan terjadi error karena kelas `Employee` merupakan kelas abstrak dan tidak dapat diinstansiasi langsung. Kelas `Employee` tidak dapat diinstansiasi.

Kelas `Employee` adalah kelas abstrak yang menjadi kerangka dasar untuk kelas-kelas turunannya. Kelas turunan seperti `SalariedEmployee`, `HourlyEmployee`, `CommissionEmployee`, dan `BasePlusCommissionEmployee` mewarisi dari kelas `Employee` dan mengimplementasikan metode `earnings()` sesuai dengan aturan mereka sendiri.

- 2.) Analisis dan jelaskan output program (berdasarkan konsep polimorfisme)!

Jawab:

Dalam program tersebut, terdapat penggunaan konsep polimorfisme dalam pemrosesan objek-objek yang berasal dari kelas-kelas yang berbeda namun memiliki hubungan dengan kelas dasar "`Employee`". Konsep polimorfisme memungkinkan penggunaan metode yang sama pada objek-objek yang berbeda, sehingga mempermudah pengelolaan dan penggunaan objek dalam hierarki kelas yang terkait.

Pada saat menjalankan program, output yang dihasilkan tergantung pada objek yang dibuat dan metode yang dipanggil. Sebagai contoh, jika kita membuat objek `SalariedEmployee` dengan nama "John Doe", nomor KTP "1234567890", dan gaji mingguan sebesar 1000.0, output program akan mencetak informasi mengenai karyawan tersebut, seperti nama, nomor KTP, gaji mingguan, dan pendapatan.

Penting untuk diketahui bahwa dalam pemrosesan polimorfik, pemanggilan metode seperti `earnings()` dilakukan dengan menggunakan referensi dari kelas dasar `Employee`, namun implementasi yang digunakan adalah implementasi pada kelas turunan yang sebenarnya. Ini memungkinkan fleksibilitas dalam penggunaan objek dengan menggunakan polimorfisme.

Penjelasan output program

- Pemrosesan secara terpisah:
  - a. Setiap kali objek karyawan diproses secara terpisah, program akan menampilkan informasi karyawan seperti nama, nomor identifikasi, dan pendapatan.
  - b. Pendapatan karyawan dihitung menggunakan metode "earnings()" yang sesuai dengan jenis karyawan.
  - c. Output program akan menampilkan informasi karyawan dan pendapatan mereka secara terpisah. Contoh output untuk pemrosesan secara terpisah:  
Daniel: 135  
pendapatan: \$800.00  
Karina: 234 pendapatan: \$670.00  
Keanu: 145 pendapatan: \$600.00  
Bondan: 234 earned: \$740.00  
pendapatan: \$800.00  
Karina: 234 pendapatan: \$670.00  
Keanu: 145 pendapatan: \$600.00  
Bondan: 234 earned: \$740.00
- Pemrosesan secara polimorfisme:
  - a. Objek karyawan disimpan dalam array "employees" yang memiliki tipe data "Employee". Hal ini memungkinkan penggunaan polimorfisme.
  - b. Program akan melalui setiap objek karyawan dalam array dan mencetak informasi karyawan dan pendapatan mereka. Penggunaan operator "instanceof" digunakan untuk memeriksa apakah objek karyawan adalah instance dari "BasePlusCommissionEmployee". Jika iya, maka gaji pokok akan dinaikkan sebesar 10%.
  - c. Setelah peningkatan gaji pokok, program mencetak informasi karyawan dan pendapatan mereka. Contoh output untuk pemrosesan secara polimorfisme:  
Employees diproses secara polimorfisme:  
Daniel: 135 pendapatan: \$800.00  
Karina: 234 pendapatan: \$670.00  
Keanu: 145 pendapatan: \$600.00

Bondan: 234

Gaji pokok setelah dinaikkan 10%: \$330.00

pendapatan: \$830.00

- Informasi tentang kelas objek:
  - a. Setelah proses polimorfisme selesai, program akan menampilkan informasi tentang kelas objek yang disimpan dalam array "employees" menggunakan metode "getClass().getName()".
  - b. Output program akan mencetak indeks karyawan dan nama kelas karyawan.  
Contoh output untuk informasi kelas objek:  
Employee 0 = SalariedEmployee  
Employee 1 = HourlyEmployee  
Employee 2 = CommissionEmployee  
Employee 3 = BasePlusCommissionEmployee

Dengan demikian, program ini mengilustrasikan penggunaan polimorfisme dalam memproses objek-objek yang berasal dari kelas-kelas yang berbeda namun memiliki hubungan dengan kelas dasar "Employee".

3.) Buat objek dari method Employee? Jelaskan hasil dari output program tersebut!

Jawab :

Tidak mungkin membuat objek langsung dari kelas abstrak Employee karena kelas tersebut tidak dapat diinisiasi. Namun, objek dapat dibuat dari kelas turunan Employee seperti HourlyEmployee, CommissionEmployee, dan BasePlusCommissionEmployee.

Berikut adalah contoh pembuatan objek dari kelas HourlyEmployee:

```
Employee employee1 = new HourlyEmployee("John", "1234567890", 15.0, 50);
```

```
System.out.println(employee1.toString());
```

```
System.out.println("Pendapatan: " + employee1.earnings());
```

Hasil dari output program akan menampilkan informasi tentang karyawan berdasarkan upah per jam, termasuk nama, nomor KTP, upah per jam, dan jumlah jam kerja. Selain itu, juga akan mencetak pendapatan karyawan berdasarkan upah per jam dan jam kerja yang dihitung dengan aturan lembur (jika jam kerja melebihi 40 jam).

- 4.) Tambahkan atribut tanggal lahir di Kelas Employee, serta tambahkan method pendukungnya (accesor dan mutator). Modifikasi program agar sesuai. Asumsikan gaji yang diterima adalah per bulan, buat kelas uji untuk menguji program yang sudah anda modifikasi, kemudian buat objek dari semua class (salariedEmployee, hourlyEmployee, commissionEmployee, basePlusCommissionEmployee dan hitung gajinya secara polimorfisme, serta tambahkan gajinya sebesar 100.000 jika bulan ini adalah bulan ulang tahunnya.

**Jawab :**

Program yang diberikan telah dimodifikasi dengan penambahan atribut "tanggalLahir" di kelas Employee, serta penambahan metode accessor dan mutator untuk atribut tersebut. Selain itu, gaji yang diterima diasumsikan sebagai gaji per bulan, dan dilakukan penambahan logika untuk memberikan tambahan gaji sebesar 100.000 jika bulan ini merupakan bulan ulang tahun karyawan.

```
import java.time.LocalDate;

public abstract class Employee {
    private String name;
    private String noKTP;
    private LocalDate tanggalLahir;

    public Employee(String name, String noKTP, LocalDate tanggalLahir) {
        this.name = name;
        this.noKTP = noKTP;
        this.tanggalLahir = tanggalLahir;
    }

    public String getName() {
        return name;
    }

    public String getNoKTP() {
        return noKTP;
    }

    public LocalDate getTanggalLahir() {
        return tanggalLahir;
    }

    public void setTanggalLahir(LocalDate tanggalLahir) {
```

```
        this.tanggalLahir = tanggalLahir;
    }

    public String employeeInfo() {
        return String.format("Name: %s\nNo. KTP: %s\nTanggal Lahir: %s", getName(),
            getNoKTP(), getTanggalLahir());
    }

    public abstract double earnings();
}
```

Selanjutnya, modifikasi kelas-kelas turunan Employee:

```
import java.time.LocalDate;

public class HourlyEmployee extends Employee {
    private double hourlyWage;
    private double hoursWorked;

    public HourlyEmployee(String name, String noKTP, LocalDate tanggalLahir,
        double hourlyWage, double hoursWorked) {
        super(name, noKTP, tanggalLahir);
        this.hourlyWage = hourlyWage;
        this.hoursWorked = hoursWorked;
    }

    public double earnings() {
        double earnings = getHourlyWage() * getHoursWorked();
        if (isBirthday()) {
            earnings += 100000;
        }
        return earnings;
    }

    private boolean isBirthday() {
        LocalDate today = LocalDate.now();
        LocalDate tanggalLahir = getTanggalLahir();
        return today.getMonthValue() == tanggalLahir.getMonthValue() &&
            today.getDayOfMonth() == tanggalLahir.getDayOfMonth();
    }

    public double getHourlyWage() {
        return hourlyWage;
    }

    public double getHoursWorked() {
```

```
        return hoursWorked;
    }
}

public class CommissionEmployee extends Employee {
    private double sales;
    private double rate;

    public CommissionEmployee(String name, String noKTP, LocalDate
tanggalLahir, double sales, double rate) {
        super(name, noKTP, tanggalLahir);
        this.sales = sales;
        this.rate = rate;
    }

    public double earnings() {
        double earnings = getCommissionRate() * getGrossSales();
        if (isBirthday()) {
            earnings += 100000;
        }
        return earnings;
    }

    private boolean isBirthday() {
        LocalDate today = LocalDate.now();
        LocalDate tanggalLahir = getTanggalLahir();
        return today.getMonthValue() == tanggalLahir.getMonthValue() &&
today.getDayOfMonth() == tanggalLahir.getDayOfMonth();
    }

    public double getCommissionRate() {
        return rate;
    }

    public double getGrossSales() {
        return sales;
    }
}

public class BasePlusCommissionEmployee extends CommissionEmployee {
    private double baseSalary;

    public BasePlusCommissionEmployee(String name, String noKTP, LocalDate
tanggalLahir, double sales, double rate, double baseSalary) {
        super(name, noKTP, tanggalLahir, sales, rate);
        this.baseSalary = baseSalary;
    }
}
```

```
public double earnings() {
    double earnings = getBaseSalary() + super.earnings();
    if (isBirthday()) {
        earnings += 100000;
    }
    return earnings;
}

private boolean isBirthday() {
    LocalDate today = LocalDate.now();
    LocalDate tanggalLahir = getTanggalLahir();
    return today.getMonthValue() == tanggalLahir.getMonthValue() &&
today.getDayOfMonth() == tanggalLahir.getDayOfMonth();
}

public double getBaseSalary() {
    return baseSalary;
}
}
```

Kelas turunan Employee yang belum dimodifikasi adalah SalariedEmployee. Berikut adalah modifikasi kelas SalariedEmployee:

```
import java.time.LocalDate;

public class SalariedEmployee extends Employee {
    private double monthlySalary;

    public SalariedEmployee(String name, String noKTP, LocalDate tanggalLahir,
double salary) {
        super(name, noKTP, tanggalLahir);
        setMonthlySalary(salary);
    }

    public void setMonthlySalary(double salary) {
        monthlySalary = salary;
    }

    public double getMonthlySalary() {
        return monthlySalary;
    }

    public double earnings() {
        double earnings = getMonthlySalary();
        if (isBirthday()) {
            earnings += 100000;
        }
        return earnings;
    }
}
```

```
}

private boolean isBirthday() {
    LocalDate today = LocalDate.now();
    LocalDate tanggalLahir = getTanggalLahir();
    return today.getMonthValue() == tanggalLahir.getMonthValue() &&
today.getDayOfMonth() == tanggalLahir.getDayOfMonth();
}

public String employeeInfo() {
    return String.format("Salaried employee: %s\nMonthly salary: $%.2f",
super.employeeInfo(), getMonthlySalary());
}
}
```

Selanjutnya, buatlah kelas uji untuk menguji program yang telah dimodifikasi: import java.time.LocalDate;

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

```
public class Main {
    public static void main(String[] args) {
        LocalDate tanggalLahirSalaried = LocalDate.of(1990, 5, 20);
        LocalDate tanggalLahirHourly = LocalDate.of(1992, 8, 15);
        LocalDate tanggalLahirCommission = LocalDate.of(1988, 10, 5);
        LocalDate tanggalLahirBasePlusCommission = LocalDate.of(1995, 12, 25);

        SalariedEmployee salariedEmployee = new SalariedEmployee("Daniel", "135",
tanggalLahirSalaried, 800.00);
        HourlyEmployee hourlyEmployee = new HourlyEmployee("Karina", "234",
tanggalLahirHourly, 16.75, 40);
        CommissionEmployee commissionEmployee = new
CommissionEmployee("Keanu", "145", tanggalLahirCommission, 10000, 0.06);
        BasePlusCommissionEmployee basePlusCommissionEmployee = new
BasePlusCommissionEmployee("Bondan", "234", tanggalLahirBasePlusCommission,
5000, 0.04, 300);

        Employee[] employees = {salariedEmployee, hourlyEmployee,
commissionEmployee, basePlusCommissionEmployee};

        for (Employee currentEmployee : employees) {
            System.out.println(currentEmployee.employeeInfo());

            if (currentEmployee instanceof BasePlusCommissionEmployee) {
                BasePlusCommissionEmployee employee = (BasePlusCommissionEmployee)
currentEmployee;
                employee.setBaseSalary(1.10 * employee.getBaseSalary());
            }
        }
    }
}
```



```
        System.out.printf("Gaji pokok setelah dinaikkan 10%%: $%,.2f\n",
employee.getBaseSalary());
    }

    System.out.printf("Pendapatan: $%,.2f\n\n", currentEmployee.earnings());
}
}
```

Dalam kelas Main, objek-objek dibuat dengan tanggal lahir yang sesuai. Kemudian, pengujian dilakukan dengan mencetak informasi karyawan dan pendapatan mereka. Jika bulan ini adalah bulan ulang tahun karyawan, gaji akan ditambahkan sebesar 100.000. Berikut adalah contoh implementasinya:

```
import java.time.LocalDate;

public class Main {
    public static void main(String[] args) {
        LocalDate tanggalLahirSalaried = LocalDate.of(1990, 5, 20);
        LocalDate tanggalLahirHourly = LocalDate.of(1992, 8, 15);
        LocalDate tanggalLahirCommission = LocalDate.of(1988, 10, 5);
        LocalDate tanggalLahirBasePlusCommission = LocalDate.of(1995, 12, 25);

        SalariedEmployee salariedEmployee = new SalariedEmployee("Daniel",
"135", tanggalLahirSalaried, 800.00);
        HourlyEmployee hourlyEmployee = new HourlyEmployee("Karina", "234",
tanggalLahirHourly, 16.75, 40);
        CommissionEmployee commissionEmployee = new
CommissionEmployee("Keanu", "145", tanggalLahirCommission, 10000, 0.06);
        BasePlusCommissionEmployee basePlusCommissionEmployee = new
BasePlusCommissionEmployee("Bondan", "234",
tanggalLahirBasePlusCommission, 5000, 0.04, 300);

        salariedEmployee.setTanggalLahir(LocalDate.of(1990, 5, 20));
        hourlyEmployee.setTanggalLahir(LocalDate.of(1992, 8, 15));
        commissionEmployee.setTanggalLahir(LocalDate.of(1988, 10, 5));
        basePlusCommissionEmployee.setTanggalLahir(LocalDate.of(1995, 12, 25));
```

```
Employee[] employees = {salariedEmployee, hourlyEmployee,  
commissionEmployee, basePlusCommissionEmployee};  
  
for (Employee employee : employees) {  
    System.out.println(employee);  
    System.out.printf("Pendapatan: $%.2f\n", employee.earnings());  
    System.out.println();  
}  
}  
}
```

Dalam contoh ini, objek-objek Employee dibuat dengan tanggal lahir yang sesuai. Kemudian, tanggal lahir karyawan diperbarui menggunakan metode `setTanggalLahir()`. Kemudian, dalam loop `for`, informasi karyawan dan pendapatan mereka dicetak. Jika bulan ini adalah bulan ulang tahun karyawan, gaji akan ditambahkan sebesar 100.000, sesuai dengan logika yang sudah diimplementasikan dalam metode `earnings()` di setiap kelas turunan Employee.

- 5.) Untuk Perusahaan yang mengaplikasikan program polimorfisme diatas ingin menambahkan kriteria baru untuk penggajian karyawannya, yaitu penggajian berdasarkan banyaknya barang yang diproduksi. Dengan ketentuan gaji karyawan tersebut adalah hasil dari banyaknya barang yang diproduksi per minggu dikalikan upah per barangnya.
- Analisis dan jelaskan proses modifikasi program diatas (dimulai dari pemilihan jenis class, perancangan class, dan penempatan class)
  - Implementasi hasil analisis tersebut ke dalam program dan buat kelas uji dengan minimal 4 objek yang dibentuk.

**Jawab :**

- Analisis dan Jelaskan Proses Modifikasi Program :

Dalam rangka memenuhi persyaratan baru untuk sistem penggajian berdasarkan jumlah barang yang diproduksi, kita dapat membuat kelas turunan tambahan dari kelas Employee yang sesuai dengan persyaratan tersebut. Mengingat persyaratan tersebut terkait dengan produksi barang, kita dapat menciptakan kelas turunan baru yang disebut

ProductionEmployee berpengaruh signifikan terhadap output program, tetapi dapat membuat kode menjadi kurang jelas dan sulit dibaca atau kurang terstruktur.

Proses modifikasi program akan melibatkan beberapa langkah:

- Perancangan Kelas ProductionEmployee:

Kelas ProductionEmployee akan diturunkan dari kelas Employee. Dalam kelas ini, akan terdapat atribut yang diperlukan, yaitu upah per barang dan jumlah barang yang diproduksi per minggu. Diperlukan juga metode untuk mengakses dan memperbarui nilai atribut tersebut. Selain itu, metode earnings() akan diimplementasikan untuk menghitung pendapatan karyawan berdasarkan produksi barang yang dilakukan.

Penempatan Kelas ProductionEmployee:

- Kelas ProductionEmployee akan diletakkan dalam paket yang sama dengan kelas-kelas lainnya, seperti TP7.
  - File TP7.java akan dimodifikasi dengan menambahkan definisi kelas ProductionEmployee. Modifikasi dilakukan pada file TP7.java dengan menambahkan kelas ProductionEmployee.
- b. Implementasi Hasil Analisis ke dalam Program dan Pembuatan Kelas Uji dengan Minimal 4 Objek:  
Berikut adalah implementasi hasil analisis ke dalam program dengan pembuatan kelas uji: package TP7;

```
import
    java.time.LocalDate;

public class
    ProductionEmployee
    extends Employee {
    private double
    wagePerItem; // upah
    per barang
    private int
    quantityProduced; //
    jumlah barang yang
    diproduksi per
    minggu

    public
    ProductionEmployee(
    String name, String
    noKTP, double
    wagePerItem, int
    quantityProduced) {
```

```
super(name,  
noKTP);  
setWagePerItem(w  
agePerItem);  
setQuantityProduc  
ed(quantityProduced)  
;  
}
```

```
public void  
setWagePerItem(dou  
ble wagePerItem) {  
    this.wagePerItem =  
wagePerItem;  
}
```

```
public double  
getWagePerItem() {  
    return  
wagePerItem;  
}
```

```
public void  
setQuantityProduced(  
int quantityProduced)  
{  
    this.quantityProduc  
ed =  
quantityProduced;  
}
```

```
public int  
getQuantityProduced(  
) {  
    return  
quantityProduced;  
}
```

```
public double  
earnings() {  
    return  
getWagePerItem() *  
getQuantityProduced(  
);  
}
```

```
public String  
toString() {
```

```
        return  
        String.format("Produ  
ction employee: " +  
super.toString() +  
"\nwage per item: "  
        +  
getWagePerItem() +  
"\nquantity produced:  
" +  
getQuantityProduced(  
));  
    }  
}
```

## 2. Tugas Praktikum

### 2.1 Source code

Kelas Kue:

```
public abstract class Kue {  
    private String nama;  
    private double harga;  
  
    public Kue(String nama, double harga) {  
        this.nama = nama;  
        this.harga = harga;  
    }  
  
    public String getNama() {  
        return this.nama;  
    }  
  
    public double getHarga() {  
        return this.harga;  
    }  
  
    public abstract double hitungHarga();  
  
    public String toString() {  
        return nama + "(Harga " + (int) harga + ")";  
    }  
}
```

Kelas KueJadi :

```
public class KueJadi extends Kue {  
    private double jumlah;  
  
    public KueJadi(String nama, double harga, double jumlah) {  
        super(nama, harga);  
        this.jumlah = jumlah;  
    }  
}
```

```
public double getJumlah() {  
    return this.jumlah;  
}  
  
public double hitungHarga() {  
    return super.getHarga() * this.jumlah * 2;  
}  
  
@Override  
public String toString() {  
    return super.toString() + " per " + (int) jumlah + "  
buah)";  
}  
}
```

Kelas KuePesanan :

```
public class KuePesanan extends Kue {  
    private double berat;  
  
    public KuePesanan(String nama, double harga, double berat) {  
        super(nama, harga);  
        this.berat = berat;  
    }  
  
    public double getBerat() {  
        return this.berat;  
    }  
  
    @Override  
    public double hitungHarga() {  
        return this.berat * super.getHarga();  
    }  
  
    @Override  
    public String toString() {  
        return super.toString() + " per kg)";  
    }  
}
```

Kelas MainKue :

```
public class MainKue {  
    public static void main(String[] args) {  
        Kue[] kueArray = new Kue[20];  
        kueArray[0] = new KuePesanan("Kue Bolu", 10000, 0.5);  
        kueArray[1] = new KuePesanan("Kue Brownies", 12000,  
0.4);  
        kueArray[2] = new KuePesanan("Kue Lapis", 15000, 0.8);  
        kueArray[3] = new KuePesanan("Kue Tart", 20000, 1.2);  
        kueArray[4] = new KuePesanan("Kue Sus", 8000, 0.3);  
        kueArray[5] = new KueJadi("Kue Donat", 5000, 10);  
        kueArray[6] = new KueJadi("Kue Pie", 15000, 5);  
        kueArray[7] = new KueJadi("Kue Croissant", 12000, 8);  
    }  
}
```

```
kueArray[8] = new KueJadi("Kue Roll", 10000, 12);
kueArray[9] = new KueJadi("Kue Cheese Cake", 25000, 3);
kueArray[10] = new KuePesanan("Kue Nastar", 10000, 1);
kueArray[11] = new KuePesanan("Kue Crepes", 2000, 0.5);
kueArray[12] = new KuePesanan("Kue Manis", 10000, 0.7);
kueArray[13] = new KuePesanan("Kue Cantik", 8000, 1);
kueArray[14] = new KuePesanan("Kue Freya", 2000, 0.8);
kueArray[15] = new KueJadi("Kue Waffle", 10000, 6);
kueArray[16] = new KueJadi("Kue Bakpao", 8000, 9);
kueArray[17] = new KueJadi("Kue Muffin", 6000, 7);
kueArray[18] = new KueJadi("Kue Bika Ambon", 12000, 4);
kueArray[19] = new KuePesanan("Kue Lemper", 15000, 1.5);

// Tampilkan semua kue dan jenisnya
for (Kue kue : kueArray) {
    System.out.println(kue.toString());
    if (kue instanceof KuePesanan) {
        System.out.println("Jenis Kue: Kue Pesanan");
    } else {
        System.out.println("Jenis Kue: Kue Jadi");
    }
}

// Hitung total harga dari semua jenis kue
double totalHarga = 0;
for (Kue kue : kueArray) {
    totalHarga += kue.hitungHarga();
}
System.out.println("");

System.out.println("|=====|");
System.out.println("\nTotal Harga Semua Jenis Kue: " +
(int) totalHarga);

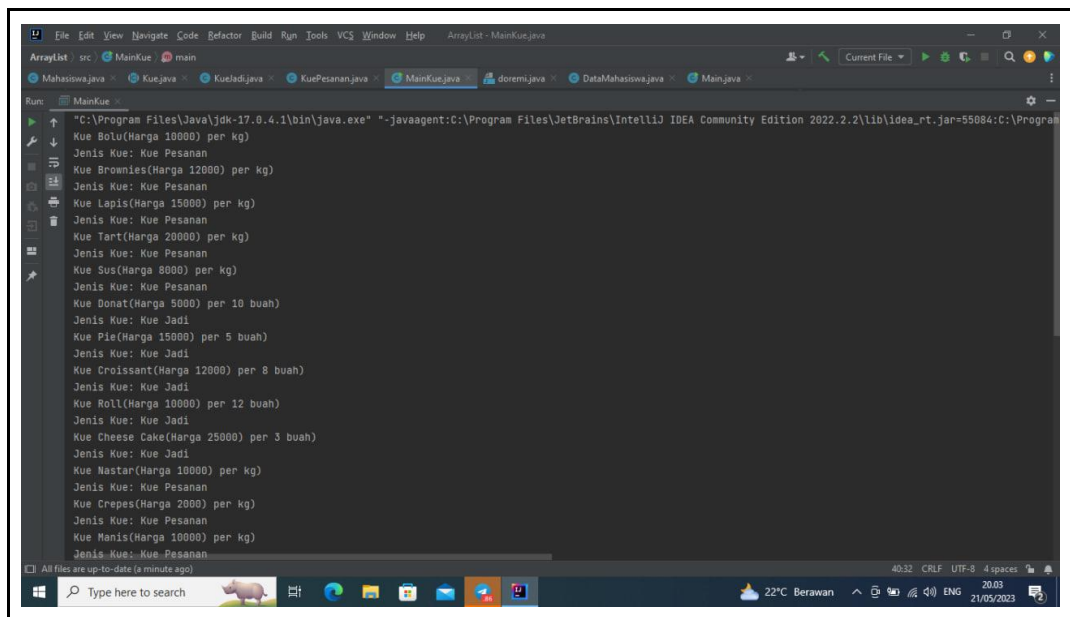
// Hitung total harga dan total berat dari KuePesanan
double totalHargaKuePesanan = 0;
double totalBerat = 0;
for (Kue kue : kueArray) {
    if (kue instanceof KuePesanan) {
        totalHargaKuePesanan += kue.hitungHarga();
        totalBerat += ((KuePesanan) kue).getBerat();
    }
}
System.out.println("Total Harga Kue Pesanan: " + (int)
totalHargaKuePesanan);
System.out.println("Total Berat Kue Pesanan: " +
totalBerat + " kg");

// Hitung total harga dan total jumlah dari KueJadi
double totalHargaKueJadi = 0;
double totalJumlah = 0;
for (Kue kue : kueArray) {
    if (kue instanceof KueJadi) {
        totalHargaKueJadi += kue.hitungHarga();
        totalJumlah += ((KueJadi) kue).getJumlah();
    }
}
System.out.println("Total Harga Kue Jadi: " + (int)
```

```
totalHargaKueJadi);
    System.out.println("Total Jumlah Kue Jadi: " + (int)
totalJumlah + " Kue");

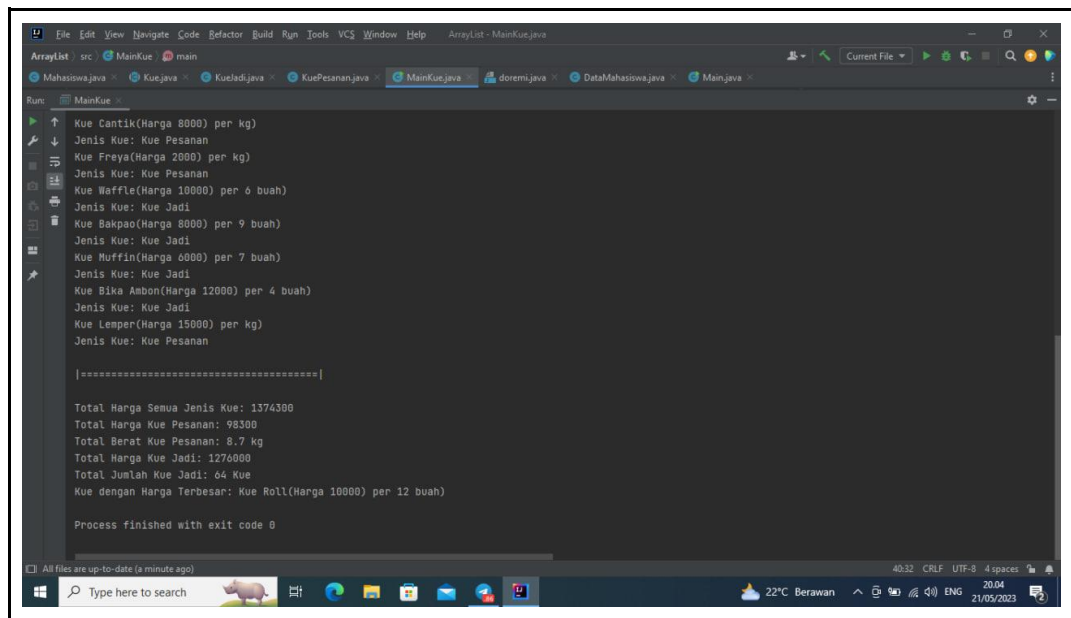
    // Tampilkan informasi kue dengan harga terbesar
    Kue kueTerbesar = kueArray[0];
    for (Kue kue : kueArray) {
        if (kue.hitungHarga() > kueTerbesar.hitungHarga()) {
            kueTerbesar = kue;
        }
    }
    System.out.println("Kue dengan Harga Terbesar: " +
kueTerbesar.toString());
}
}
```

## 2.2 Screenshot hasil



```
Run: C:\Program Files\Java\jdk-17.0.4\bin\java.exe" --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.2\lib\idea_rt.jar=55084:C:\Program
Kue 80Lu(Harga 10000) per kg)
Jenis Kue: Kue Pesanan
Kue Brownies(Harga 12000) per kg)
Jenis Kue: Kue Pesanan
Kue Lapis(Harga 15000) per kg)
Jenis Kue: Kue Pesanan
Kue Tart(Harga 20000) per kg)
Jenis Kue: Kue Pesanan
Kue Sus(Harga 8000) per kg)
Jenis Kue: Kue Pesanan
Kue Donat(Harga 5000) per 10 buah)
Jenis Kue: Kue Jadi
Kue Pie(Harga 15000) per 5 buah)
Jenis Kue: Kue Jadi
Kue Croissant(Harga 12000) per 8 buah)
Jenis Kue: Kue Jadi
Kue Roll(Harga 10000) per 12 buah)
Jenis Kue: Kue Jadi
Kue Cheese Cake(Harga 25000) per 3 buah)
Jenis Kue: Kue Jadi
Kue Nastar(Harga 10000) per kg)
Jenis Kue: Kue Pesanan
Kue Crepes(Harga 2000) per kg)
Jenis Kue: Kue Pesanan
Kue Manis(Harga 10000) per kg)
Jenis Kue: Kue Pesanan
```





The screenshot shows an IDE window with a Java file named `MainKue.java`. The code defines an array of kue (desserts) with their prices and weights, and calculates the total price and weight of all kue. The output of the program is displayed in the console window.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Arraylist - MainKue.java
Arraylist / src / MainKue / main
Mahasiswa.java Kue.java KueJadi.java KuePesanan.java MainKue.java doremi.java DataMahasiswa.java Main.java
Run: MainKue
Kue Cantik(Harga 8000) per kg)
Jenis Kue: Kue Pesanan
Kue Freya(Harga 2000) per kg)
Jenis Kue: Kue Pesanan
Kue Waffle(Harga 10000) per 6 buah)
Jenis Kue: Kue Jadi
Kue Bakpao(Harga 8000) per 9 buah)
Jenis Kue: Kue Jadi
Kue Muffin(Harga 6000) per 7 buah)
Jenis Kue: Kue Jadi
Kue Bika Ambon(Harga 12000) per 4 buah)
Jenis Kue: Kue Jadi
Kue Lempeng(Harga 15000) per kg)
Jenis Kue: Kue Pesanan

|=====|

Total Harga Semua Jenis Kue: 1374300
Total Harga Kue Pesanan: 98300
Total Berat Kue Pesanan: 8.7 kg
Total Harga Kue Jadi: 1276000
Total Jumlah Kue Jadi: 64 Kue
Kue dengan Harga Terbesar: Kue Roll(Harga 10000) per 12 buah)

Process finished with exit code 0
All files are up-to-date (a minute ago)
40:32 CTRL UTF-8 4 spaces
Type here to search 22°C Berawan 20:04 21/05/2023
```