UAS 2022

1. Buatlah sebuah kelas yang berisi 2 buah metode. Metode pertama bersifat statis digunakan untuk menghitung berat badan ideal (kg) yang merupakan fungsi dari tinggi badan (cm) dengan formula: berat = 0.9*(tinggi-100), sedangkan metode kedua merupakan metode non statis yang digunakan untuk menghitung kalori yang dibutuhkan perhari yang merupakan fungsi dari usia (tahun) dengan formula: kalori = 0.8*berat+0.2*usia.
   a. Buatlah Class yang berisi metode yang dimaksud serta buatlah pula kelas untuk menguji kelas tersebut di atas!

```
public class BBIdeal {
    public static double hitungBeratIdeal(double tinggi) {
        return 0.9 * (tinggi - 100);
    }

    public double hitungKalori(int usia, double berat) {
        return 0.8 * berat + 0.2 * usia;
    }
}

public class Main {
    public static void main(String[] args) {
        double beratIdeal = BBIdeal.hitungBeratIdeal(170);
        System.out.println(beratIdeal);

        BBIdeal body = new BBIdeal();
        double kalori = body.hitungKalori(30, 65);
        System.out.println(kalori);
    }
}
```

   Output:
   63.0
   58.0

   b. Jelaskan perbedaan metode yang menggunakan statis dan yang non statis!
      i. Metode statis dapat diakses langsung dari nama kelas tanpa membuat instance kelas.
      ii. Metode non-statis hanya bisa diakses melalui instance kelas.
      iii. Metode statis tidak dapat mengakses variabel instance, hanya variabel statis.
      iv. Metode non-statis dapat mengakses variabel instance maupun statis.

---

2. Buatlah sebuah kelas bersarang atau nested class, kelas terluar diberi nama Outer. Di dalam kelas Outer terdapat sebuah kelas yang diberi nama Inner1. Selanjutnya, di dalam kelas Inner1 terdapat sebuah kelas yang diberi nama Inner2. Di dalam kelas Inner2 terdapat metode static yakni Metode Tambah yang digunakan untuk menjumlahkan 2 bilangan bulat. Buatlah kelas ujinya untuk memanggil metode Tambah dari dalam kelas Inner2 tersebut!

```
public class Outer {
    class Inner1 {
```

```
        class Inner2 {
            public static int Tambah(int a, int b) {
                return a + b;
            }
        }
    }

    public static void main(String[] args) {
        int hasil = Outer.Inner1.Inner2.Tambah(5, 3);
        System.out.println(hasil);
    }
}
```

Output: 8

---

3. Interface digunakan untuk memberikan hubungan antar class yang tidak memiliki hubungan pewarisan. Sebuah perusahaan otomotif memproduksi 3 jenis kendaraan yaitu Mobil, Motor, dan Listrik. Harga jual masing-masing kendaraan ditentukan berdasarkan spesifikasi utama kendaraan antara lain: (1) Volume cylinder (V), (2) Jenis bahan bakar (F), dan (3) Pajak penjualan (P). Adapun spesifikasi dan besarnya pajak penjualan untuk setiap kelas kendaraan disajikan pada tabel berikut:

Kelas Kendaraan	Volume Cylinder (V)	Jenis Bahan Bakar (F)	Pajak Penjualan P (%)
Mobil	2500	Solar	10
Motor	150	Bensin	5
Listrik	1	Listrik	0

Untuk menentukan harga jual kendaraan digunakan formula sebagai berikut:

a. Untuk kelas Mobil dan Motor: HargaKendaraan = (1+P)*(k1*V+k2+k3*F)
b. Untuk kelas Listrik: HargaKendaraan = k1*V+k2+k3*F

k1, k2, dan k3 merupakan konstanta dengan nilai k1 = 2000000, k2 = 20000000, dan k3 = 10000000. Sedangkan F, P merupakan variabel. Nilai P = 0.1 untuk kelas Mobil dan P = 0.05 untuk kelas Motor, nilai F = 1 untuk bensin dan listrik, F = 2.5 untuk solar.
Mobil, Motor, dan Listrik tidak memiliki hubungan pewarisan akan tetapi Ketiga kelas tersebut akan saling berhubungan dengan menggunakan interface bernama Kendaraan.
Buatlah sebuah interface Kendaraan yang dapat diimplementasikan pada ketiga kelas tersebut sehingga variable objek dari interface Kendaraan tersebut dapat memanggil 1 method yang sama yakni HargaKendaraan yang berperilaku sesuai objek yang dipanggil, buatlah interface, class dan class pengujinya!

```
public interface Kendaraan {
    public double HargaKendaraan();
}

class Mobil implements Kendaraan {
    double v;
    double f;
    double p;
```

```
    Mobil(double v, double f, double p) {
        this.v = v;
        this.f = f;
        this.p = p;
    }

    public double HargaKendaraan() {
        double k1 = 2000000;
        double k2 = 20000000;
        double k3 = 10000000;
        return (1+p)*(k1*v + k2 + k3*f);
    }
}

class Motor implements Kendaraan {
    double v;
    double f;
    double p;

    Motor(double v, double f, double p) {
        this.v = v;
        this.f = f;
        this.p = p;
    }

    public double HargaKendaraan() {
        double k1 = 2000000;
        double k2 = 20000000;
        double k3 = 10000000;
        return (1+p)*(k1*v + k2 + k3*f);
    }
}

class Listrik implements Kendaraan {
    double v;
    double f;

    Listrik(double v, double f) {
        this.v = v;
        this.f = f;
    }

    public double HargaKendaraan() {
        double k1 = 2000000;
        double k2 = 20000000;
        double k3 = 10000000;
        return k1*v + k2 + k3*f;
    }
}

import java.text.DecimalFormat;
public class Main {
    public static void main(String[] args) {
```

```
        DecimalFormat formatter = new DecimalFormat("#,###");

        Kendaraan mobil = new Mobil(2500, 2.5, 0.1);
        double hargaMobil = mobil.HargaKendaraan();
        System.out.print("Harga Mobil : ");
        System.out.println(formatter.format(hargaMobil));

        Kendaraan motor = new Motor(150, 1, 0.05);
        double hargaMotor = motor.HargaKendaraan();
        System.out.print("Harga Motor : ");
        System.out.println(formatter.format(hargaMotor));

        Kendaraan listrik = new Listrik(1, 1);
        double hargaListrik = listrik.HargaKendaraan();
        System.out.print("Harga Listrik : ");
        System.out.println(formatter.format(hargaListrik));
    }
}
```

Output:
Harga Mobil : 5,549,500,000
Harga Motor : 346,500,000
Harga Listrik : 32,000,000

UAS 2021

1. Jelaskan secara singkat mengenai konsep paket/package dalam object oriented programming!
   Secara singkat, paket (package) dalam pemrograman berorientasi objek memiliki beberapa fungsi utama:
   a. Menyediakan namespace untuk mengelompokkan kelas-kelas dan interface yang berhubungan menjadi satu kesatuan logika. Hal ini untuk menghindari konflik nama kelas antar paket.
   b. Meningkatkan modularitas dan enkapsulasi kode program. Paket memungkinkan pengembang untuk membagi kode program yang besar menjadi modul-modul yang lebih kecil dan independen.
   c. Mengatur visibilitas (aksesibilitas) dari kelas, interface, dan anggota kelas (method & variabel) agar terkontrol dengan baik. Kita bisa menentukan elemen mana saja yang bisa diakses oleh paket lain melalui pengaturan akses.

---

2. Jelaskan konsep Polymorphism pada Object Oriented Programming , serta berikan contohnya!
   Polymorphism memungkinkan objek memiliki banyak bentuk (form). Ada 2 jenis:
   a. Compile time: Method overload. perilaku method berbeda berdasarkan parameter.
   b. Run time: Inheritance dan overriding method. Objek bertingkah berbeda pada saat runtime.
   Contoh:

```
public class Kalkulator {
    public int tambah(int a, int b){
```

```
        return a + b;
    }

    public double tambah(double a, double b){
        return a + b;
    }
}

public class Hewan {
    public void suara() {}
}

public class Kucing extends Hewan {
    @Override
    public void suara() {
        System.out.println("meong");
    }
}
```

---

3. Buatlah class Balok dan Bola yang meng-extends kelas abstrak berikut dan buatlah kelas ujinya:

```
abstract class bangun_ruang {
    final double phi = 3.14;
    double panjang, lebar, tinggi, radius;

    public abstract double Volume();
    public abstract double Luas_Permukaan();
    public abstract double Diagonal();
    public abstract double Diameter();
}

class Balok extends bangun_ruang {
    double panjang;
    double lebar;
    double tinggi;

    public Balok(double panjang, double lebar, double tinggi) {
        this.panjang = panjang;
        this.lebar = lebar;
        this.tinggi = tinggi;
    }

    @Override
    public double Volume() {
        return panjang * lebar * tinggi;
    }

    @Override
    public double Luas_Permukaan() {
        return 2 * (panjang*lebar + panjang*tinggi + lebar*tinggi);
    }

    @Override
```

```
    public double Diagonal() {
            return Math.sqrt(Math.pow(panjang, 2) + Math.pow(lebar, 2) +
Math.pow(tinggi, 2));
    }

    @Override
    public double Diameter() {
        return 0;
    }
}

class Bola extends bangun_ruang {
    double radius;

    public Bola(double radius) {
        this.radius = radius;
    }

    @Override
    public double Volume() {
        return 4/3 * phi * Math.pow(radius, 3);
    }

    @Override
    public double Luas_Permukaan() {
        return 4 * phi * Math.pow(radius, 2);
    }

    @Override
    public double Diagonal() {
        return 0;
    }

    @Override
    public double Diameter() {
        return 2 * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        Balok balok = new Balok(4, 5, 8);
        System.out.println(balok.Volume());

        Bola bola = new Bola(10);
        System.out.println(bola.Luas_Permukaan());
    }
}
```

Output:
160.0
1256.0

4. Jelaskan fungsi dari eksepsi dan berikan contoh sederhananya!

Fungsi eksepsi (exception) dalam pemrograman Java adalah:

a. Untuk menangani kesalahan (error) yang terjadi saat program dijalankan agar program tetap berjalan dan terkendali alih-alih langsung berhenti karena crash.
b. Memisahkan alur kesalahan dari alur normal program sehingga kode program lebih terstruktur dan mudah dibaca.
c. Memberikan pesan kesalahan yang deskriptif kepada user maupun developer.

Contoh:

```
public class ContohException {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3};
        try {
            System.out.println(numbers[5]); // akses index yang tidak ada
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Terjadi error: " + e.getMessage());
        }
    }
}
```

UAS FMIPA 2022

1. Jelaskan apa yang dimaksud dengan:
   a. Interface: kerangka/blueprint untuk mendefinisikan perilaku suatu objek. Interface berisi kumpulan method abstract yang wajib diimplementasi kelas turunannya.
   b. Overloading method: kemampuan sebuah method untuk memiliki nama yang sama tapi parameter yang berbeda. Tujuannya adalah untuk memudahkan penggunaan ulang kode.
   c. Polymorphism: *ada diatas*

---

2. Jelaskan perbedaan interface dan abstract class jelaskan!
   a. Method, Interface hanya boleh memiliki deklarasi method abstract. Sedangkan abstract class bisa memiliki method abstract dan non-abstract (method konkrit).
   b. Fields, Interface tidak boleh memiliki fields/attributes. Sedangkan abstract class boleh memiliki fields.
   c. Inheritance, Interface mendukung multiple inheritance. Class hanya single inheritance.

---

3. Buatlah sebuah contoh mengenai polymorphisme! *Ada diatas*

Materi

- ➔ Aturan Interface :
  - Implementasi menggunakan kata kunci 'implements' pada pembuatan class
  - Tiap method tidak memiliki body
  - Interface bisa di implementasi sebanyak mungkin pada suatu class.
- ➔ Abstrak: Kelas dalam pemrograman berorientasi objek yang tidak dapat diinstansiasi, dan seringkali memiliki metode tanpa implementasi (abstrak) yang harus diimplementasikan oleh kelas turunannya.
- ➔ Aturan Abstrak :
  - Objek tidak dapat dibuat langsung dari kelas abstrak
  - Kelas turunan yang meng-extends kelas abstrak harus mengimplementasi semua method dalam kelas abstrak
  - Method tanpa implementasi dideklarasikan dengan kata kunci 'abstract' dan tidak memiliki body
- ➔ Inner Class: konsep dalam pemrograman berorientasi objek di mana sebuah kelas dideklarasikan di dalam kelas lain.
- ➔ Single Inheritance: Suatu konsep dalam pemrograman berorientasi objek di mana sebuah kelas dapat mewarisi sifat-sifat dari satu kelas induk atau kelas dasar.
- ➔ Package: mengelompokkan kelas-kelas yang saling berhubungan menjadi satu kesatuan. Berfungsi untuk namespace, enkapsulasi, dan modularisasi program Java.
- ➔ Exception: menangani error yang terjadi pada saat program dijalankan agar alur program tetap terkontrol. exception digunakan untuk memisahkan alur kesalahan dari program utama.
- ➔ Thread: membuat alur eksekusi program secara paralel/bersamaan. Thread memungkinkan beberapa proses dijalankan secara asynchronous pada waktu yang bersamaan. Hal ini berguna untuk meningkatkan performa aplikasi Java, misalnya ketika mengakses UI dan proses backend secara paralel.
- ➔ Perbedaan Single Thread dan Multi Thread:
  - Single Thread:
    - Hanya ada satu alur eksekusi kode.
    - Semua task dijalankan secara berurutan dan blocking. Task berikutnya harus menunggu task sebelumnya selesai.
    - Lebih mudah diimplementasikan dan dikelola.
    - Performa umumnya lebih rendah karena tidak ada paralelisme.
  - Multi Thread:
    - Ada beberapa alur eksekusi kode yang berjalan paralel.
    - Task bisa dijalankan secara asynchronous dan non-blocking. Task tidak perlu menunggu task lain selesai.
    - Memerlukan sinkronisasi dan manajemen resource yang lebih kompleks.
    - Performa biasanya lebih tinggi karena adanya paralelisme dalam eksekusi kode.
    - Cocok untuk aplikasi seperti UI dan backend yang bisa dijalankan secara paralel.
  - Jadi intinya multi threading memungkinkan eksekusi kode secara paralel sehingga performa aplikasi Java bisa lebih optimal. Namun juga memerlukan desain dan implementasi yang lebih cermat.