Soal 1. Silakan Anda perhatikan nomor yang ada di sebelah kanan beberapa baris kode di program utama. Silakan Anda isi dengan **[tidak boleh]** jika baris kode di sebelah kiri mengandung ekspresi yang tidak diperbolehkan. Kemudian, jika ekspresi diperbolehkan (tidak ada masalah), silakan tulis outputnya. **Asumsi: tidak ada kesalahan kompilasi, dan kesalahan ekspresi tidak mempengaruhi output pada ekspresi lain**.

```
public interface IA {
    public int methodA1();
    public int methodA2();
}
```

```
public interface IB {
    public int methodB1();
    public int methodB2();
}
```

```
public class A implements IA {
    private final int VAL = 1;

    public int methodA1() {
        return VAL;
    }
    public int methodA2() {
        return VAL+1;
    }
    public int methodA3() {
        return VAL+2;
    }
}
```

```
public class B implements IA, IB {
    private final int VAL = 10;

    public int methodA1() {
        return VAL;
    }
    public int methodA2() {
        return VAL+1;
    }
    public int methodB1() {
        return VAL+2;
    }
    public int methodB2() {
        return VAL+3;
    }
}
```

```
public class C implements IB {
    private final int VAL = 100;

    public int methodB1() {
        return VAL;
    }
    public int methodB2() {
        return VAL+1;
    }
    public int methodC1() {
        return VAL+2;
    }
}
```

```
public class Main {
    public static void main(String[] args) {

        A oa = new A();
        B ob = new B();

        System.out.println(oa.methodA1());   //(1)
        System.out.println(oa.methodA2());   //(2)
        System.out.println(oa.methodA3());   //(3)
```

```
        IA iia = oa;

        System.out.println(iia.methodA2());  //(4)
        System.out.println(oa.methodA2());   //(5)
        System.out.println(iia.methodA3());  //(6)

        IB iib = new C();

        System.out.println(iib.methodB1());  //(7)
        System.out.println(iib.methodC1());  //(8)

        IB iib2 = new B();
        IA iia2 = new B();
        System.out.println(iib2.methodB1()); //(9)
        System.out.println(iib2.methodB2()); //(10)
        System.out.println(iib2.methodA1()); //(11)
        System.out.println(iia2.methodA1()); //(12)
        System.out.println(iia2.methodA2()); //(13)
        System.out.println(iia2.methodB2()); //(14)


        IA iia3 = new A();
        System.out.println(iia3. methodA2()); //(15)
        iia3 = new B();
        System.out.println(iia3. methodA2()); //(16)
        iia3 = ob;
        System.out.println(iia3. methodA2()); //(17)
        System.out.println(iia3. methodB2()); //(18)


    }
}
```

Soal 2. Perhatikan desain kelas pada potongan program berikut. Pertama, kita membuat sebuah kelas **Pizza** yang merepresentasikan sebuah jenis makanan, yaitu Pizza:

```
class Pizza {
      private double calories;
      private String taste;
      private char size; //S, M, L

      public Pizza(double calories, String taste, char size) {
            this.calories = calories;
            this.taste = taste;
            this.size = size;
      }

      public double getCals() {return calories;}
      public String getTaste() {return taste;}
      public char getSize() {return size;}
}
```

Selanjutnya, kita ingin membuat sebuah kelas **PizzaCollection** yang dapat menyimpan banyak **Pizza** dan menghitung **total kalori** yang ada pada kumpulan **Pizza** tersebut:

```
public class PizzaCollection {
	private ArrayList<Pizza> pizzaList;

	public PizzaCollection() {
		pizzaList = new ArrayList<Pizza>();
	}

	public void addPizza(Pizza p) {
		pizzaList.add(p);
	}

	public double totalCalories () {
		double sum = 0;
		for (int i = 0; i < pizzaList.size(); i++) {
			Pizza p = pizzaList.get(i);
			sum += p.getCals();
		}
		return sum;
	}
}
```

Untuk saat ini, makanan yang tersedia hanya **Pizza** saja. Namun, di masa mendatang, mungkin saja terdapat makanan-makanan baru yang juga perlu ditampung. Misal, terdapat sebuah jenis makanan baru yaitu **Sunkist**.

```
class Sunkist {
	private double calories;
	private String taste;

	public Sunkist(double calories, String taste) {
		this.calories = calories;
		this.taste = taste;
	}

	public double getCals() {return calories;}
	public String getTaste() {return taste;}
	public String getOrigin() {return "California";}
}
```

Tentunya, untuk menyimpan **Sunkist**, kita tidak bisa menggunakan kelas **PizzaCollection** karena kelas **PizzaCollection** diimplementasikan untuk menyimpan **Pizza** saja.

Solusinya ? kita membuat kelas baru yaitu kelas **SunkistCollection** yang serupa dengan **PizzaCollection**.

Apakah ini adalah **solusi yang baik** ?

Jika belum baik, silakan Anda kemukakan alasannya dan **desain ulang kelas-kelas tersebut menjadi lebih baik.**

Goal : Kita ingin membuat sebuah kelas yang dapat menampung "*segala macam makanan*" (*collection of edible things*)

Soal 3. **Latihan Merancang Kelas-kelas dengan melibatkan interface**

Mesin Cuci

Anda diminta untuk membuat program yang dapat memodelkan bagaimana **MesinCuci** bekerja. MesinCuci adalah sebuah benda **elektronik** yang dapat "*dihidupkan*" dan "*dimatikan*". Benda elektronik tidak hanya MesinCuci, tetapi juga radio, televisi, dan yang lainnya.

MesinCuci mempunyai daftar barang-barang yang ingin dicuci. Daftar ini bisa ditambah dan bisa dikurangi secara dinamis tergantung kebutuhan si pengguna MesinCuci. Akan tetapi, MesinCuci mempunyai kapasitas maksimal. Jika MesinCuci sudah mencapai kapasitas maksimal, maka MesinCuci tidak akan menerima barang tambahan untuk dicuci. MesinCuci dapat mencuci semua benda-benda *"yang dapat dicuci"*, seperti **Pakaian** dan **Piring**. Mungkin, suatu saat nanti, jenis benda-benda yang dapat dicuci oleh MesinCuci dapat bertambah, misal **Karpet**.

Beberapa benda yang dicuci merupakan benda *"yang dapat dipecahkan"* seperti **Piring** dan **Gelas**. Untuk saat ini, treatment yang dilakukan MesinCuci terhadap benda-benda seperti itu sama saja dengan benda yang lainnya.

Program utama hanya berisi kode testing yang menghidupkan sebuah objek MesinCuci dengan kapasitas tertentu, kemudian menyalakannya, memasukkan beberapa barang untuk dicuci, mulai mencuci barang-barang tersebut, dan terakhir mematikan aliran listrik objek MesinCuci tersebut.

Langkah pengerjaan:

1. Pahami permasalahan dan kemudian Tentukan kelas-kelas dan interface-interface yang terlibat. Anda tidak perlu mengimplementasikan semua "kata benda" atau konsep yang disebut di soal menjadi sebuah kelas. Cukup yang berkaitan dengan "mesin cuci" saja.
2. Rancanglah gambar diagram kelas dengan **notasi UML**.
3. Rancang atribut dan method dari kelas yang terlibat **secukupnya saja**. Terapkan konsep **abstraksi** di sini.
4. Implementasikan rancangan Anda ke kode dengan bahasa pemrograman Java.