Pengenalan Bahasa JAVA
Inheritance - Interface

by: Yohanes Nugroho
rev: Achmad Imam Kistijantoro,
Saiful Akbar,
Yani Widyani

Inheritance

- Di Java hanya ada single inheritance
- Penurunan selalu bersifat "public" (tidak ada penurunan private dan protected seperti di C++)
- Kata kunci yang dipakai adalah **extends**

```
class Line3D extends Line2D {


}
```

Kelas Abstrak

- Di C++ boleh ada method virtual murni dalam sebuah kelas, di Java juga boleh ada method semacam itu
  - method virtual murni: method yang belum ada isinya
- Method virtual murni dalam Java ditandai dengan kata kunci `abstract`
- Kelas yang mengandung method virtual murni harus dideklarasikan dengan kata kunci `abstract`

Contoh kelas Abstrak dan Turunannya

```
abstract class Bangun {
   void test() {
     System.out.println("Luas"+getLu
     as()); }
   abstract int getLuas();
}
class Lingkaran extends Bangun {
   int getLuas() { return pi*r*r;}
}
```

Keyword super

- Memanggil konstruktor superclass
  - Dengan parameter yang sesuai tentunya
  - Parameter boleh kosong, seperti ini: `super()`
- Harus merupakan statement pertama dalam konstruktor anak
  - Harus mengaktifkan `super` sebelum melakukan operasi yang lain

Memanggil konstruktor parent


```
class Ortu {
   Ortu(String namaKeluarga) {
     family=namaKeluarga; }
}
class Anak  {
   Anak(String nama, String
     namaKeluarga) {
       super(namaKeluarga);
   }
}
```

Pemanggilan Method pada Parent

- Jika suatu method meng-override method parent dan ingin memanggil implementasi parent, gunakan syntax:

```
super.namamethod(parameter)
```

- Sifat super pada konstruktor
  - konstruktor default (nullary constructor) parent akan selalu dipanggil jika super() tidak dipanggil

Interface

- Java memiliki konsep interface yang tidak dimiliki C++
  - Konsep ini memungkinkan sebagian fitur multiple inheritance diimplementasikan
- Interface adalah kelas yang semua methodnya belum didefinisikan
  - Semua method kosong

Contoh interface


```
interface Draw {
   void draw();
   void draw3D();
}
```

- Interface tidak punya konstruktor, destruktor (finalizer), dan apapun, hanya punya member variabel dan deklarasi method

Implementasi interface

- Isi interface diimplementasikan oleh kelas dengan keyword **implements**
- Sebuah kelas boleh mengimplementasikan banyak interface
- Contoh:

```
class Lingkaran implements Draw {
   void draw() { /*implementasi draw*/ }
   void draw3D() { /*implementasi
     draw3D*/ }
}
```

Implementasi banyak interface


```
interface Color {
   void setColor(int color);
   int getColor();
}
class Lingkaran implements Draw, Color
  {
   void draw() { /*implementasi draw*/
     }
   void draw3D() { /*implementasi
     draw3D*/ }
   void setColor(int color);
   int getColor();
}
```

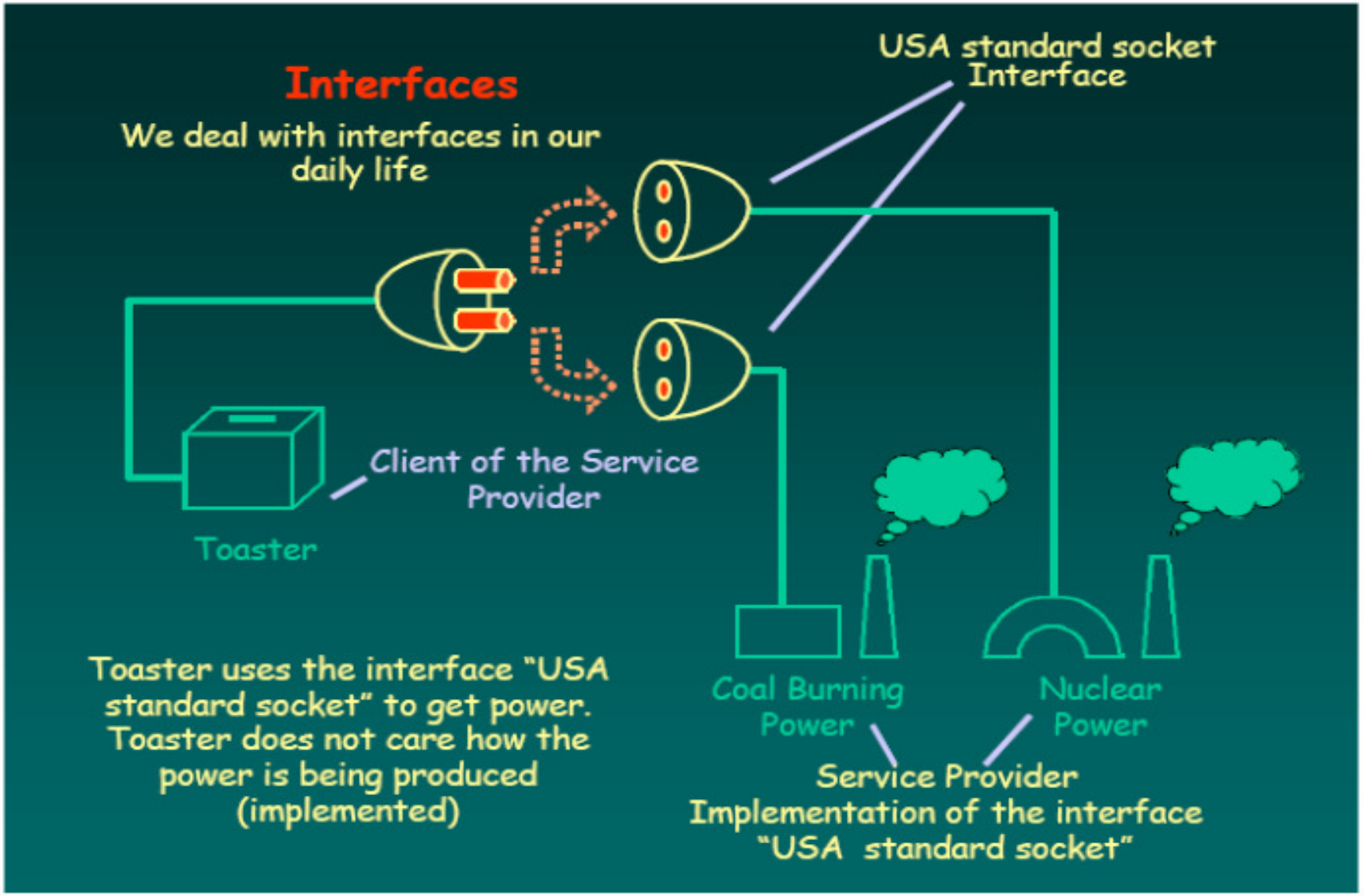
Beda kelas abstrak dengan interface

- Kelas abstrak boleh memiliki method yang sudah diimplementasikan
  - Interface harus "kosong" (tidak ada method yang terdefinisi pada interface)
- Kelas hanya boleh meng-extend (diturunkan dari) satu kelas
  - Kelas boleh mengimplementasikan banyak interface

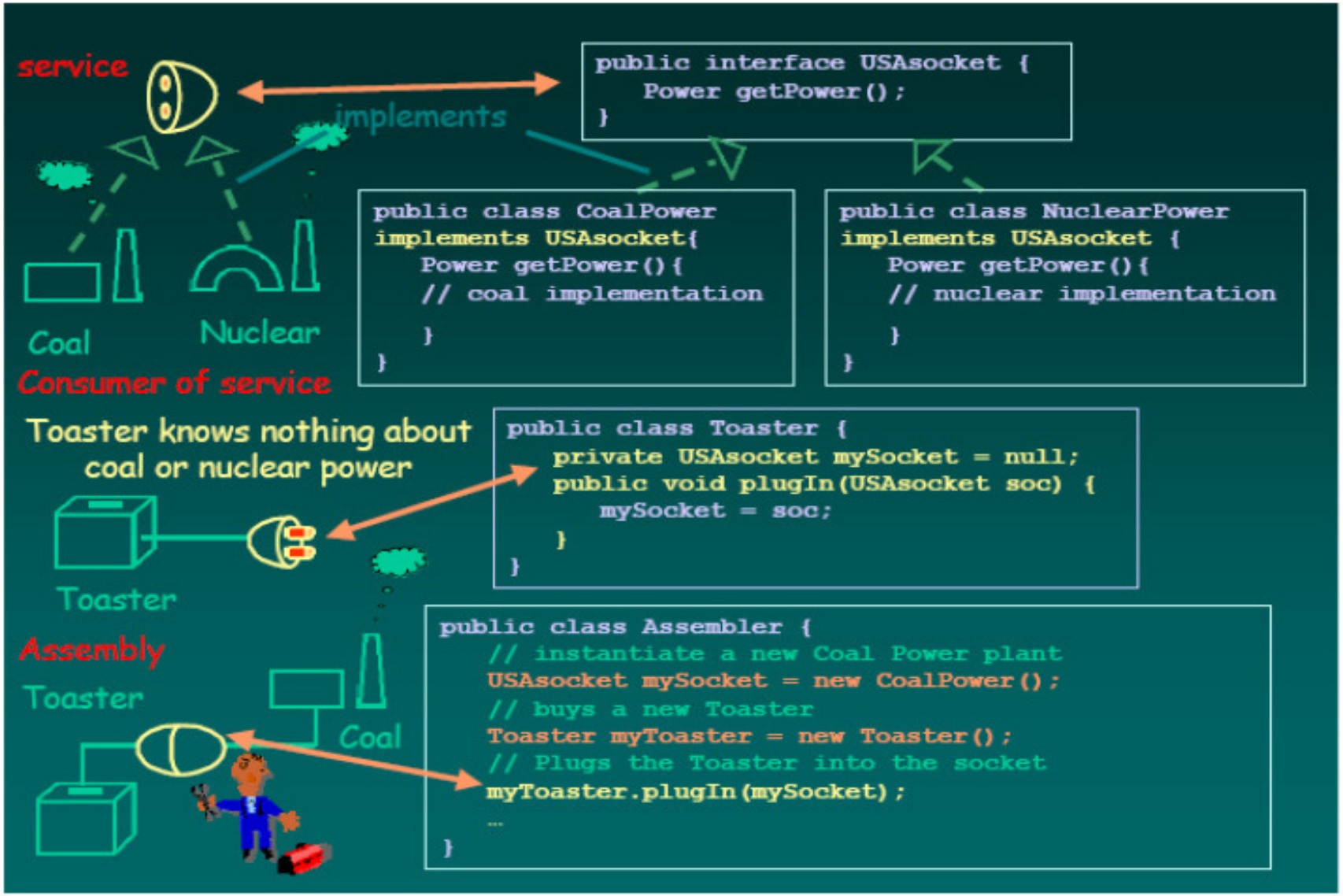
Kapan memakai kelas abstrak dan interface

- Kelas abstrak
  - Jika sudah ada algoritma yang bisa diimplementasikan di kelas tersebut
- Interface
  - Hanya memberi kontrak, misalnya Interface Measureable untuk menyatakan objek yang bisa diukur keliling dan luasnya

Contoh Pemanfaatan Interface (1)



Contoh Pemanfaatan Interface (2)



Polymorphism

- Sebuah kelas "tahu" konteksnya berdasarkan reference yang ditunjuknya
- Contoh:
  - Lingkaran dan Segitiga diturunkan dari Bangun. Bangun memiliki method luas yang mengembalikan integer yang masih abstrak (diimplementaskan oleh Lingkaran dan Segitiga)
  - Jika sebuah variabel (reference) bertipe Bangun menunjuk ke Lingkaran, maka "sifatnya" akan seperti Lingkaran, jika menunjuk ke Segitiga, maka "sifatnya" seperti Segitiga.

Contoh - Polymorphism


```
Lingkaran l = new Lingkaran(10);
Segitiga s = new Segitiga(1,2,3);
Bangun b = l;
/* mencetak lingkaran */
System.out.println(b.getLuas());
/* mencetak segitiga */
b = s;
System.out.println(b.getLuas());
```

Method/Algoritma Generik dengan Polymorphism

- Dari contoh sebelumnya bisa dibuat method printLuas:

```
void printLuas(Bangun b) {
    System.out.println(b.getLuas());
}

printLuas(s);
printLuas(l);
```