Java: Inheritance

IF2210 – Semester II 2022/2023

by: Yohanes Nugroho; rev: AI, SA, YW, SAR

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 Lingkaran extends Bangun {
  // ...
}
```

Yang terjadi saat inheritance

- Atribut yang diturunkan dapat langsung digunakan.
- Jika dideklarasikan atribut dengan nama yang sama dengan atribut pada *superclass*, atribut di *superclass* menjadi **tersembunyi**. (*not recommended*)
- Dapat dideklarasikan atribut baru yang tidak ada di *superclass*.
- *Method* yang diturunkan dapat langsung digunakan.
- Jika dideskripsikan *method* dengan nama dan *signature* yang sama dengan *method* pada *superclass*, *method* pada *superclass* di-**override**.
- Jika dideskripsikan *method* statik dengan nama dan *signature* yang sama dengan method statik pada *superclass*, method statik pada *superclass* menjadi **tersembunyi**.
- Dapat dideklarasikan *method* baru yang tidak ada di *superclass*.
- Dapat dibuat konstruktor yang memanggil konstruktor *superclass*.

Polymorphism

- Java menganut *static polymorphism*—sifat sebuah objek adalah berdasarkan *reference* yang ditunjuknya, bukan tipe pada saat deklarasi.
- Contoh:
  - `Lingkaran` dan `Segitiga` diturunkan dari `Bangun`. `Bangun` memiliki *method* `getLuas()` yang di-*override* 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 luas lingkaran */
System.out.println(b.getLuas());

b = s;
/* mencetak luas segitiga */
System.out.println(b.getLuas());
```

- Di C++, pemanggilan *method* `getLuas()` bergantung pada cara memanggilnya.

Hubungan is-a

- Jika kelas `Lingkaran` adalah turunan dari `Bangun`, maka sebuah `Lingkaran` adalah sebuah `Bangun` juga (*Lingkaran is a Bangun*).
- Maka jika sebuah *method* menerima parameter bertipe `Bangun`, ia dapat di-*supply* dengan `Lingkaran`. Contoh:

```
void doSomethingTo(Bangun b) { ... }
Lingkaran l = new Lingkaran(5);
doSomethingTo(l);
```

Method dan kelas final

- Sebuah implementasi *method* yang ditandai *keyword* `final` tidak dapat di-*override*.
  - Beberapa *method* di kelas `Object` adalah `final`, contohnya: `getClass()`, `wait()`/`notify()` (untuk pemrograman *concurrent*).
- Sebuah kelas yang ditandai *keyword* `final` tidak dapat diturunkan.
  - Contoh: kelas `System`, `String`.

Kelas abstrak

- Kelas abstrak adalah kelas yang dideklarasikan dengan *keyword* `abstract`.
  - Kelas abstrak dapat berisi *method* abstrak.
- Kelas abstrak tidak dapat diinstansiasi.
- Kelas abstrak dapat diturunkan.
  - Jika kelas turunannya tidak mengimplementasikan semua *method* abstrak yang diturunkan, kelas turunan tsb. harus dideklarasikan `abstract` juga.

Method abstrak

- Method abstrak adalah *method* yang dideklarasikan tanpa implementasi, sbb:
  - Menggunakan *keyword* `abstract`,
  - Tidak menggunakan `{ }`,
  - Diakhiri titik koma (`;`).

  Contoh: `abstract float getLuas();`
- Jika sebuah kelas memiliki *method* abstrak, kelas tersebut harus abstrak juga.

Contoh kelas Abstrak dan Turunannya

```
abstract class Bangun {
  void test() {
    System.out.println("Luas" + getLuas());
  }
  abstract float getLuas();
}

class Lingkaran extends Bangun {
  float r;

  @override /* anotasi ini tidak wajib */
  float getLuas() { return Math.PI * r * r;}
}
```

Keyword super

- Menginvokasi *behavior* dari *parent class*.
- Dapat digunakan untuk:
  - memanggil konstruktor *parent* dari konstruktor turunannya,
  - memanggil *method* milik *parent* dari *method* milik turunannya.

Memanggil konstruktor parent

- Gunakan `super(...)` dengan parameter yang sesuai.
  - Parameter boleh kosong, seperti ini: `super()`
- Harus merupakan *statement* pertama dalam konstruktor anak.
  - Harus menginvokasi `super` sebelum melakukan operasi yang lain.
- Sifat `super` pada konstruktor:
  - konstruktor *default* (*nullary constructor*) *parent* akan selalu dipanggil jika `super(...)` tidak dipanggil.
- Terjadi *constructor chaining*: konstruktor semua kelas "leluhur"-nya dipanggil.

Contoh

```
class Person {
  // ...
  Person(String name) {
    this.name = name;
  }
}

class Student extends Person {
  // ...
  Student(String name, String studentId) {
    super(name);
    this.studentId = studentId;
  }
}
```

Memanggil method parent

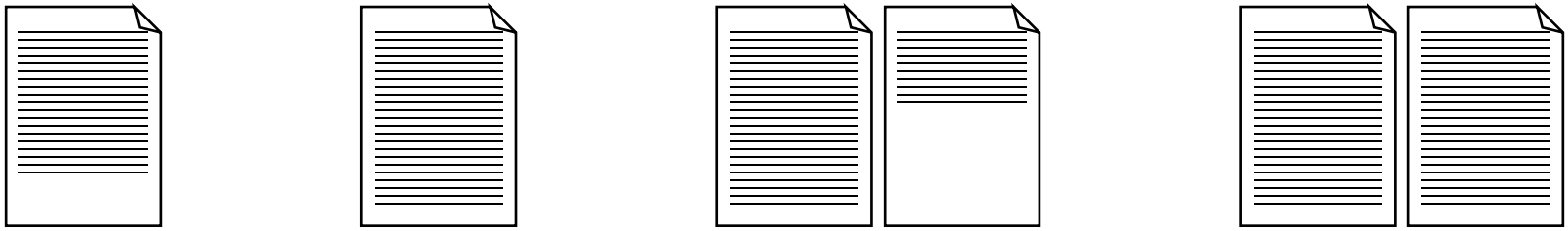
- Jika suatu *method* meng-*override method parent* dan ingin memanggil implementasi *parent*, gunakan sintaks seperti pada contoh berikut:

```
class Person {
  // ...
  void print() { System.out.println("name: " + name); }
}

class Student extends Person {
  // ...
  void print() {
    super.print();
    System.out.println("id: " + studentId);
  }
}
```

Tugas Baca

- Gast, H. (2015). *How to use objects: code and concepts*. Addison-Wesley Professional.
- Section 1.4, 1.6-1.8
- Buat summary min. 1 halaman, max. 3 halaman
  - excluding header e.g. nama, NIM
  - excluding baris kosong e.g. Enter 2×)
- Pengumpulan: Edunex



Not OK OK OK OK