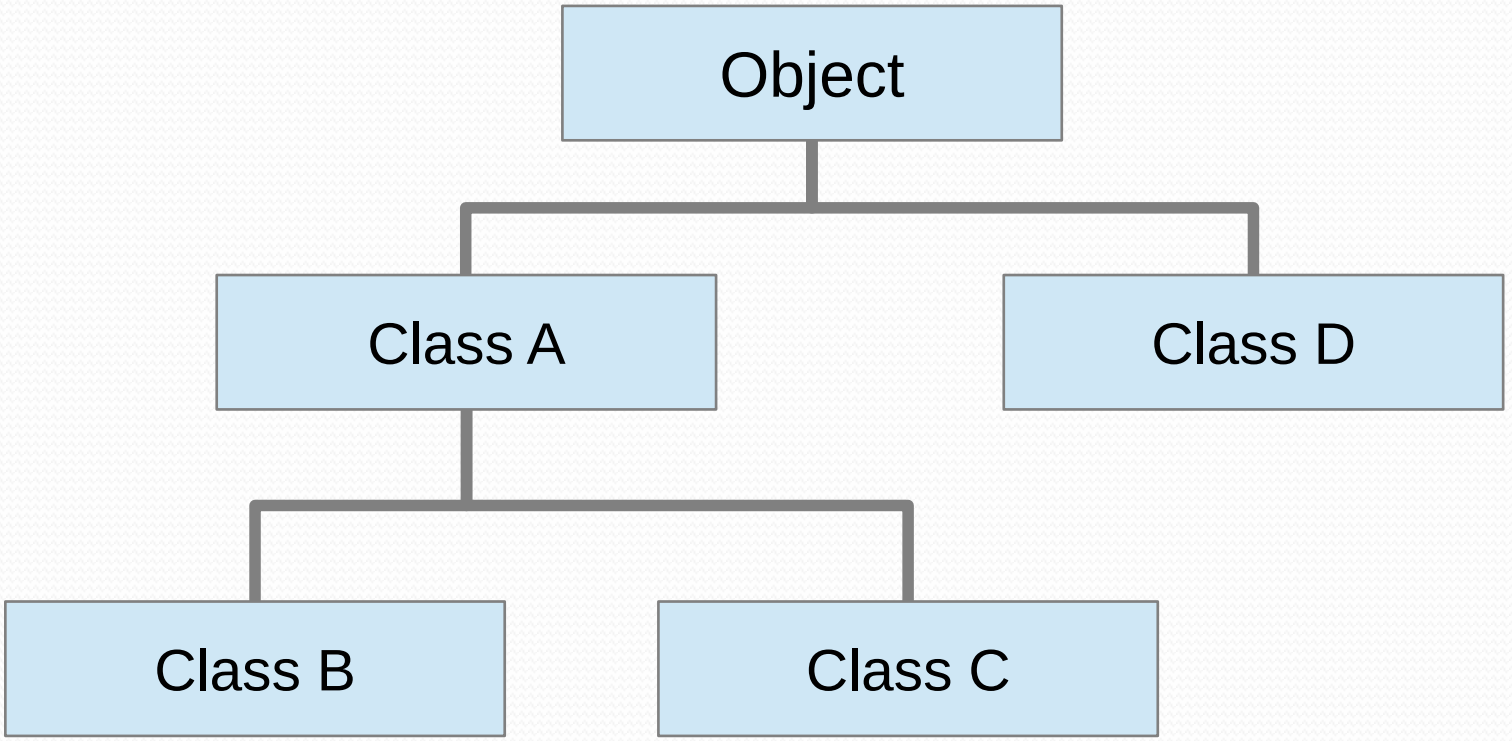
Pertemuan V

Inheritance, Abstract class, dan Interface

Pewarisan (Inheritance)

- Dalam java, semua class, termasuk dalam class yang membangun Java API, adalah subclasses dari superclass Object.
- Class utama dalam hirarki class : superclass
- Class di bawah class pokok dalam hirarki class dikenal sebagai subclass dari class tersebut.


Object
Class A
Class D
Class B
Class C

Pengertian

- Pewarisan adalah suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Ini merupakan keuntungan besar dalam pemrograman berbasis objek.
- Sehingga, anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass.

Mendefinisikan superclass dan subclass

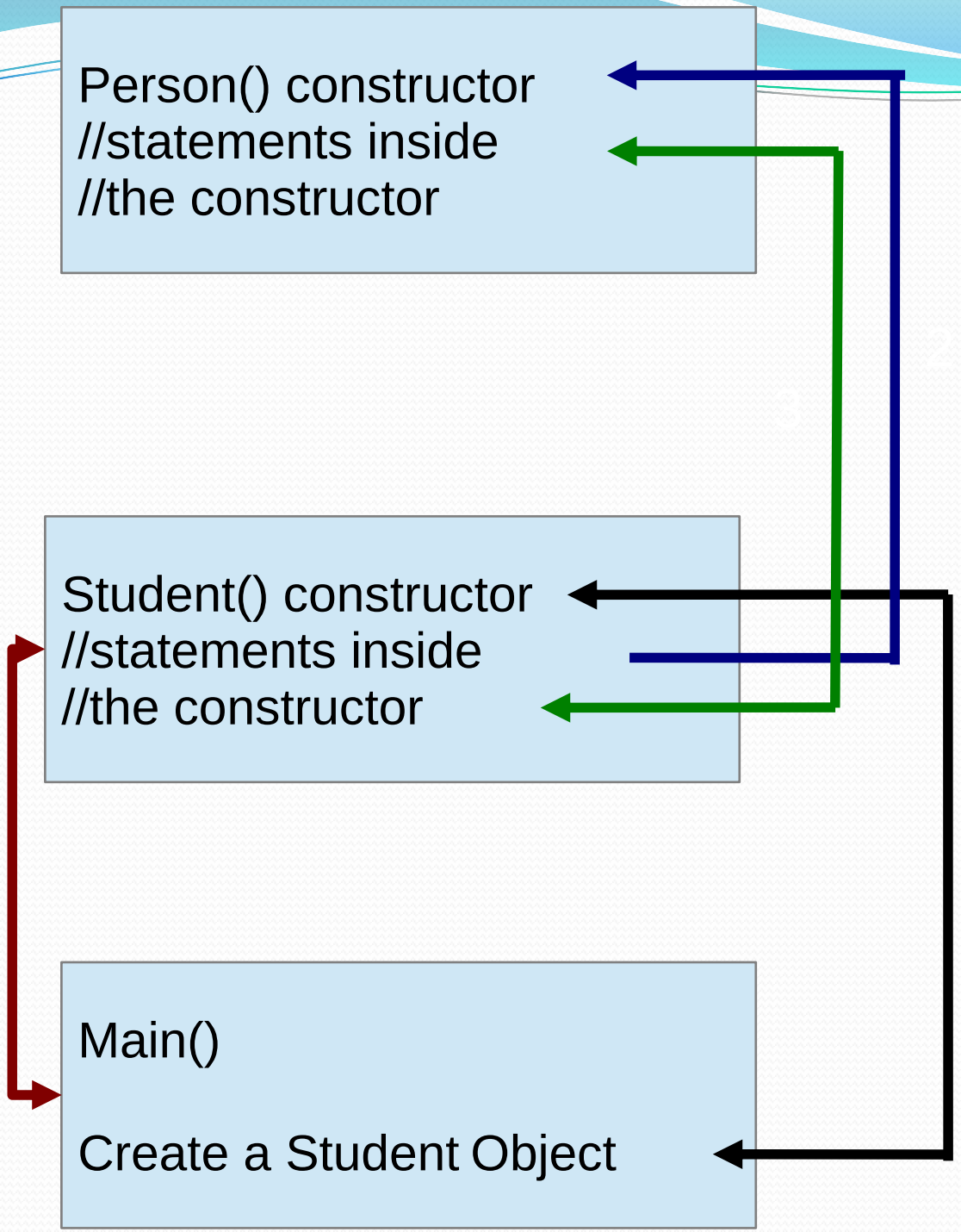
- Untuk memperoleh suatu class, kita menggunakan kata kunci `extends`.
- Sebagai contohnya, mari kita liat class induk

Person.java

  - Perhatikan bahwa atribut *name* dan *address* dideklarasikan sebagai protected.
  - Hal ini dikarenakan kita menginginkan atribut-atribut ini dapat diakses oleh subclasses dari superclasses.
  - Ingat, bahwa hanya modifier private yang tidak dapat diakses oleh subclasses-nya.

- Selanjutnya, kita membuat class lain bernama Student.
- Student juga sebagai Person, maka kita akan meng-*extends* class Person.
  - Kita dapat mewariskan semua properti dan method dari setiap class Person yang ada.

Lebih lengkap, kita lihat program Student.java


Person() constructor
//statements inside
//the constructor
Student() constructor
//statements inside
//the constructor
Main()
Create a Student Object

Kata kunci Super

- Subclass juga dapat memanggil construktor secara eksplisit dari superclass terdekatnya
- Hal ini dapat dilakukan dengan pemanggil constructor `super`.
  - Pemanggil super ini akan menghasilkan eksekusi dari superclass constructor yang bersangkutan, berdasarkan dari argumen sebelumnya.
- Lebih jauh, kita lihat program Student1.java

- Ada beberapa hal yang perlu diperhatikan ketika menggunakan pemanggil konstruktor super :
  - Pemanggil super() harus dijadikan pernyataan pertama dalam konstruktor
  - Pemanggil super() hanya dapat digunakan dalam definisi konstruktor
  - Termasuk konstruktor this() dan pemanggil super() tidak boleh terjadi dalam konstruktor yang sama

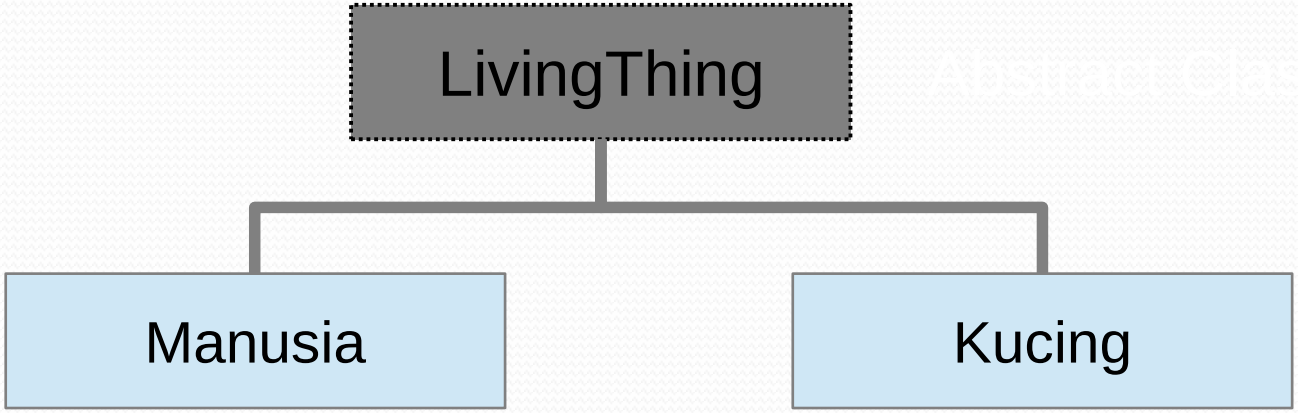
Overriding Method

- Terkadang class asal perlu mempunyai implementasi berbeda dari method yang khusus dari superclass tersebut.
- Oleh karena itu, method overriding digunakan.
- Subclass dapat mengesampingkan method yang didefinisikan dalam superclass dengan menyediakan implementasi baru dari method tersebut.
- Contoh Program : Person1.java dan Student2.java

Abstract Class

- Semisal, kita ingin membuat superclass yang mempunyai method tertentu yang berisi implementasi, dan juga beberapa method yang akan di-override oleh sub-class nya.
- Sebagai contoh kita membuat superclass `LivingThing.java`.
- Class ini mempunyai method breath, eat, sleep, walk.
- Tetapi ada beberapa method yang sifatnya tidak bisa digeneralisasi. Contoh walk.

- Tidak semua kehidupan berjalan (walk) dilakukan dengan cara sama.
- Manusia berjalan dengan dua kaki. Contoh lain kucing berjalan dengan empat kaki



Abstract Class

- Class abstract adalah sebuah class abstract adalah class yang tidak dapat di-instantiate.
- Sekali muncul di atas hirarki class pemrograman berbasis object, dan mendefinisikan keseluruhan aksi pada objek dari semua subclass-nya.
- Method dalam class abstract yang tidak mempunyai implementasi dinamakan method abstract.
  - Tinggal menulis deklarasi method tanpa tubuh class dan digunakan keyword abstract.

```
public abstract void someMethod();
```

Interface

- Interface adalah jenis khusus dari blok yang hanya berisi method signature (atau constant)
- Interface mendefinisikan sebuah (signature) dari sebuah kumpulan method tanpa tubuh.
- Sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class.
- Interface juga menunjukkan polymorphism karena program dapat memanggil method interface dan versi yang teapt dari method yang akan dieksekusi tergantung dari tipe objek yang melewatinya.

Kenapa kita menggunakan Interface?

- Jika kita ingin class yang tidak berhubungan mengimplementasikan method yang sama.
- Melalui interface, kita dapat menangkap kemiripan diantara class yang tidak berhubungan tanpa membuatnya seolah-olah class yang berhubungan.

- Contohnya :
  - Class Line, dimana berisi method yang menghitung panjang dari garis

  Dan membandingkan objek Line ke objek dari class sama
  - Class lain, MyInteger, berisi method yang membandingkan objek MyInteger ke objek dari class yang sama,
- Kedua class mempunyai method yang mirip dimana sama2 membandingkan mereka dari objek lain dalam tipe sama, namun tidak saling berhubungan.
- Interface Relation, sebagai implementasi kasus diatas

```
public interface Relation
{
  public boolean isGreater (Object a, Object b);
  public boolean isLess(Object a, Object b);
  public boolean isEqual (Object a, Object b);
}
```

- Alasan lain menggunakan Interface adalah untuk menyatakan sebuah interface pemrograman objek tanpa menyatakan classnya.
  - Kita benar2 menggunakan interface sebagai tipe data

Interface Vs Class Abstract

- Perbedaan utama adalah :
  - Method interface tidak punya tubuh,
  - Interface hanya dapat mendefinisikan konstanta
  - Interface tidak langsung mewariskan hubungan dengan class istimewa lainnya, mereka didefinisikan secara independent.

- Satu ciri dari sebuah interface dan class adalah pada tipe mereka.
- Ini artinya bahwa interface dapat digunakan dalam tempat-tempat dimana class dapat digunakan.

```
PersonInterface pi = new Person();
Person pc = new Person();
```

- Kita tidak dapat membuat instance dari interface

```
PersonInterface pi = new PersonInterface();
     // COMPILE ERROR!!
```

- Ciri umum lain, interface maupun class, sama2 dapat mendefinisikan method.

Membuat Interface

- Untuk membuat interface,

```
public interface [InterfaceName] {
    //beberapa method tanpa isi;
}
```

- Contoh Program interface, yang mendefinisikan hubungan antara dua objek yaitu sebagai berikut :
  - Relation.java
  - Line.java
- Penggunaan interface, kita gunakan keyword `implements`

Hubungan dari Interface ke Class

- Class dapat mengimplementasikan sebuah interface selama kode implementasi untuk semua method yang didefinisikan dalam interface tersedia
- Class hanya dapat meng-`extends` satu superclass. Tetapi dapat meng-`implements` banyak interface

```
Public class Person implements PersonInterface,
                    LivingThing,WhateverInterface {
        //beberapa kode ditulis di sini
 }
```

```
public class ComputerScience extends Student
             Implements PersonInterface, LivingThing {
    //beberapa kode ditulis di sini
}
```

- Catatan bahwa sebuah interface bukan bagian dari hirarki pewarisan class
- Class yang tidak berhubungan dapat mengimplementasikan interface yang sama

- Sama seperti abstract class, interface tidak dapat dibuat objek secara langsung. Bahkan interface lebih abstract lagi.
- Jika abstract class masih dapat menampung variable dan metode real, interface tidak dapat.

	Abstract Class	**Interface**
variable	bebas	Harus public static final
konstruktor	Objek tidak bisa langsung dibuat melalui abstract class. Harus melalui class yang diturunkan dari abstract class	Tidak ada konstruktor
metode	bebas	Harus public abstract

Pewarisan Antar Interface

- Interface bukan bagian dari hirarki class.
- Namun interface dapat mempunyai hubungan pewarisan antar interface sendiri.
- Contohnya, kita punya dua interface `StudentInterface` dan `PersonInterface`.

```
Public interface PersonI {
  ……..
}

Public interface StudentI extends PersonI {
  ……..
}
```

Student meng-`extends PersonInterface`, maka ia mewarisi semua deklarasi method dalam `PersonInterface`.

Terima kasih...