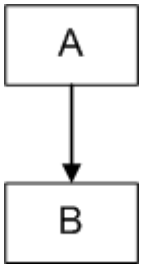
Pemrograman Berorientasi Obyek

Pertemuan 5: Pewarisan (*inheritance*)

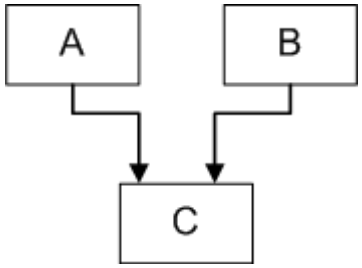
Pewarisan *(Inheritance)*

- Salah satu keunggulan Pemrograman Berorientasi Objek adalah *Reusability* (penggunaan kembali) dari sebuah class yang telah dibuat dan diuji coba sebelumnya
- *Reusability* dapat :
  - Menghemat waktu, uang, dan tenaga
  - Mengurangi stress
  - Meningkatkan reliabilitas (daya tahan) software
- Konsep *reusability* diterjemahkan ke dalam pemrograman dengan mekanisme pewarisan *(inheritance)* atau penurunan *(derivation)*
- Class induk disebut *base class* atau *superclass* atau *parent class*, dan class anak disebut *derived class (class turunan)* atau *subclass* atau *child class*
- Class turunan dapat mewarisi sebagian atau semua sifat dari class induk

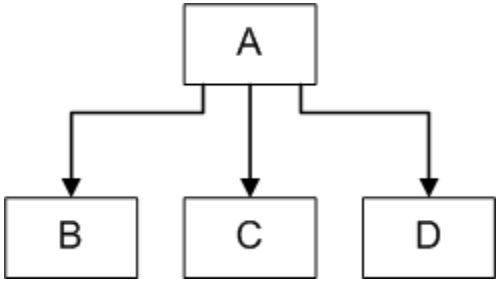
Beberapa Tipe Inheritance



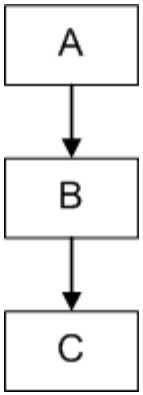
Single inheritance



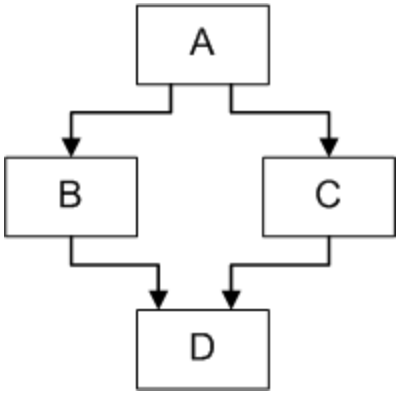
Multiple inheritance



Hierarchical inheritance

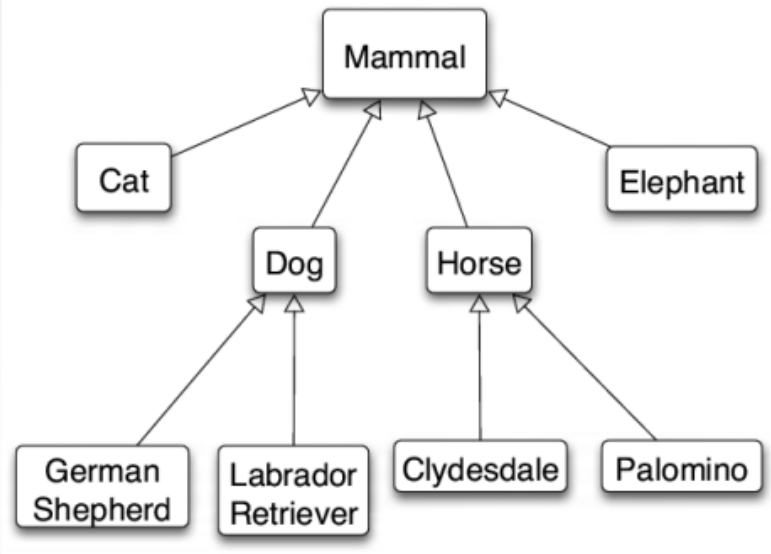


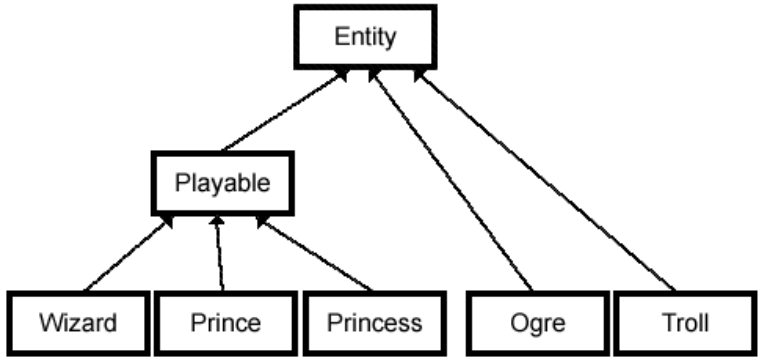
Multi-level inheritance

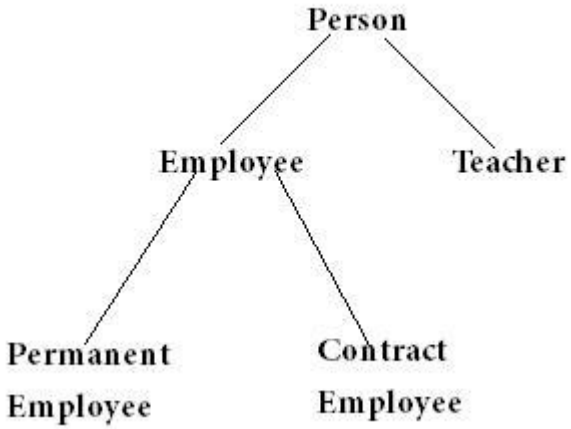


Hybrid inheritance

Contoh Inheritance

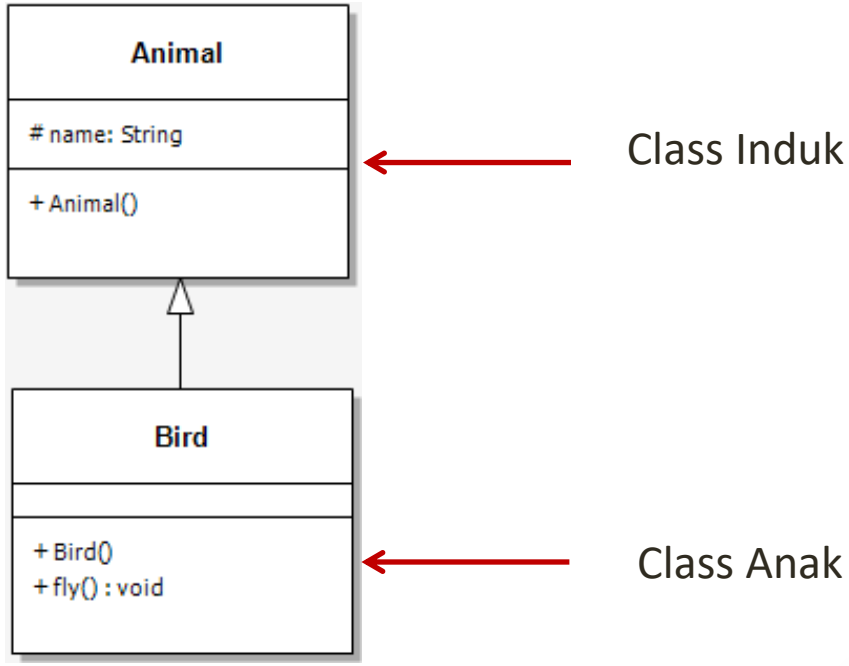






Membuat Inheritance di Raptor

- Hubungkan class yang memiliki inheritance dengan tanda



Inheritance pada Ruby dan C++

Ruby

```ruby
class Animal
    def initialize()
        puts "I am an animal"
    end
    def speak()
        puts "Hello"
    end
end

class Bird < Animal
    def fly()
        puts "I am flying"
    end
end
```

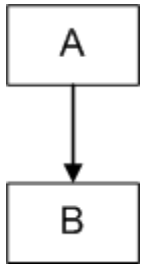
Menggunakan keyword **<**

C++

```cpp
class A {
public:
    A ()
    {
        cout<<"New A";
    }
};

class B: public A
{
public:
    B ()
    {
        cout<<"New B";
    }
};
```

Menggunakan keyword ***:***

1. Single Inheritance



- Contoh dalam Java:

```
1  package samples;
2
3  public class Animal {
4      public Animal() {
5          System.out.println("I am an animal");
6      }
7      public void speak(){
8          System.out.println("Hello");
9      }
10     public static void main(String[] args){
11         Animal a = new Animal();
12         a.speak();
13         Bird b = new Bird();
14         b.speak();
15         b.fly();
16     }
17
18 }
19
20 class Bird extends Animal {
21     public void fly(){
22         System.out.println("I'm flying");
23     }
24 }
```

Notasi inheritance

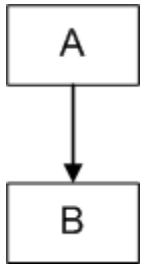
Output:

```
I am an animal
Hello
I am an animal
Hello
I am flying
```

- Class `Bird` dapat dituliskan di dalam class `Animal`, atau terpisah di luar class
- Jika ditulis di luar superclass, penulisannya ditambah modifier `public`

- Subclass `Bird` mewarisi method constructor dan `speak` milik superclass `Animal`, sehingga tidak perlu didefinisikan lagi, KECUALI jika diperlukan
- Class Bird menambahkan method `fly()` yang tidak dimiliki oleh superclassnya

1. Single Inheritance



- Contoh pendefinisian ulang method milik superclass

```
1  package samples;
2
3  public class Animal {
4      public Animal() {
5          System.out.println("I am an animal");
6      }
7      public void speak(){
8          System.out.println("Hello");
9      }
10     public static void main(String[] args){
11         Animal a = new Animal();
12         a.speak();
13         Bird b = new Bird();
14         b.speak();
15         b.fly();
16     }
17 }
18
19 class Bird extends Animal {
20     public Bird(){
21         super();
22         System.out.println("I am a bird");
23     }
24     public void speak(){
25         System.out.println("Chirp...chirp....");
26     }
27     public void fly(){
28         System.out.println("I'm flying");
29     }
30 }
```

Output:

```
I am an animal
Hello
I am an animal
I'm a bird
Chirp .... chirp.....
I am flying
```

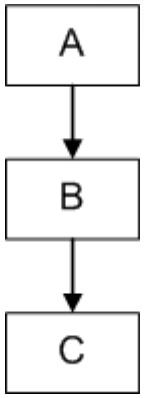
`super()`: Memanggil method constructor milik superclass (bersifat opsional)

Mendefinisikan ulang method `speak` milik superclass

Subclass `Bird` dapat mendefinisikan ulang method constructor dan `speak` milik superclass `Animal`

2. Multi-level Inheritance

- Membuat subclass dari subclass

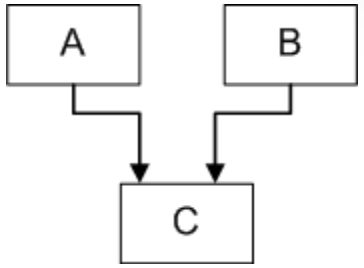



```
1  package samples;
2
3  public class Animal {
4      public Animal() {
5          System.out.println("I am an animal");
6      }
7      public void speak(){
8          System.out.println("Hello");
9      }
10     public static void main(String[] args){
11         Bird b = new Bird();
12         b.speak();
13         b.fly();
14         Penguin p = new Penguin();
15         p.speak();
16         p.fly();
17     }
18 }
19
20 class Bird extends Animal {
21     public Bird(){
22         super();
23         System.out.println("I am a bird");
24     }
25     public void speak(){
26         System.out.println("Chirp...chirp....");
27     }
28     public void fly(){
29         System.out.println("I'm flying");
30     }
31 }
32
33 class Penguin extends Bird {
34     public void fly(){
35         System.out.println("No thanks. I'd rather swimming");
36     }
37 }
```

Output:

```
I am an animal
I'm a bird
Chirp .... chirp.....
I am flying
I am an animal
I'm a bird
Chirp .... chirp.....
No, thanks. I'd rather swimming
```

3. Multiple Inheritance

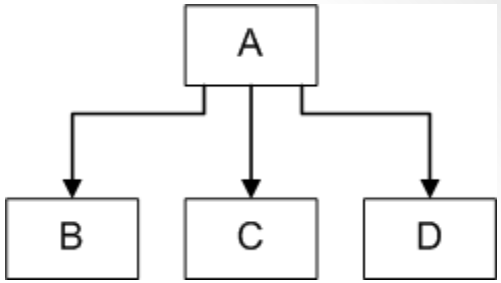


- Ruby, Java, C# dan sebagian besar bahasa pemrograman yang baru dan berorientasi objek tidak memiliki konsep *multiple inheritance*
- Bahasa pemrograman C++ masih mengimplementasikan *multiple inheritance*

```
#include <iostream>
using namespace std;
class Area {
  public:
    float area_calc(float l,float b){
        return l*b;
    }
};
class Perimeter {
  public:
    float peri_calc(float l,float b)
        return 2*(l+b);
    }
};
```

```
/* Rectangle class is derived from classes Area and Perimeter. */
class Rectangle : private Area, private Perimeter {
    private:
        float length, breadth;
    public:
       Rectangle() : length(0.0), breadth(0.0) { }
       void get_data( ) {
           cout<<"Enter length: ";
           cin>>length;
           cout<<"Enter breadth: ";
           cin>>breadth;
       }
       float area_calc() {
       /* Calls area_calc() of class Area and returns it. */
           return Area::area_calc(length,breadth);
       }

       float peri_calc() {
       /* Calls peri_calc() function of class Perimeter and returns it. */
           return Perimeter::peri_calc(length,breadth);
       }
};

int main() {
    Rectangle r;
    r.get_data();
    cout<<"Area = "<<r.area_calc();
    cout<<"\nPerimeter = "<<r.peri_calc();
    return 0;
}
```

4. Hierarchical Inheritance



- Konsepnya sama dengan Single Inheritance

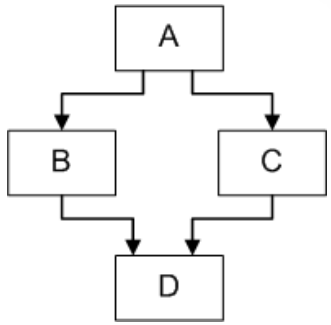
```java
package samples;

public class Animal {
    String name;
    public Animal() {
        System.out.println("I am an animal");
    }
    public void speak(){
        System.out.println("Hello");
    }
    public static void main(String[] args){
        Animal a = new Animal();
        a.speak();
        Bird b = new Bird();
        b.speak();
        Cat c = new Cat();
        c.speak();
    }
}
class Bird extends Animal {
    public Bird(){
        super();
        name="Bird";
        System.out.println("I am a "+name);
    }
    public void speak(){
        System.out.println("Chirp...chirp....");
    }
    public void fly(){
        System.out.println("I'm flying");
    }
}
```

```java
class Cat extends Animal {
    public Cat(){
        super();
        name="cat";
        System.out.println("I am a "+name);
    }
    public void speak(){
        System.out.println("Meow");
    }
}
```

Output:

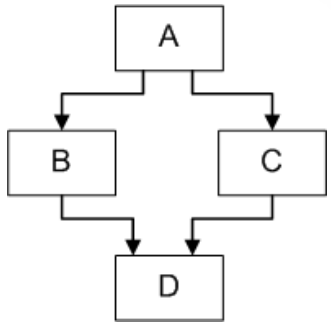
```
I am an animal
Hello
I am an animal
I'm a bird
Chirp .... chirp.....
I am an animal
I'm a cat
Meow
```

5. Hybrid Inheritance



- Jika bahasa pemrograman tidak mendukung *multiple inheritance*, maka dia juga tidak mendukung *hybrid inheritance*
- Hanya C++ saja yang mendukung *hybrid inheritance*. Ruby dan C# tidak
- Java dapat mendukung hybrid inheritance tidak dengan menggunakan *class*, tetapi *interface*

5. Hybrid Inheritance

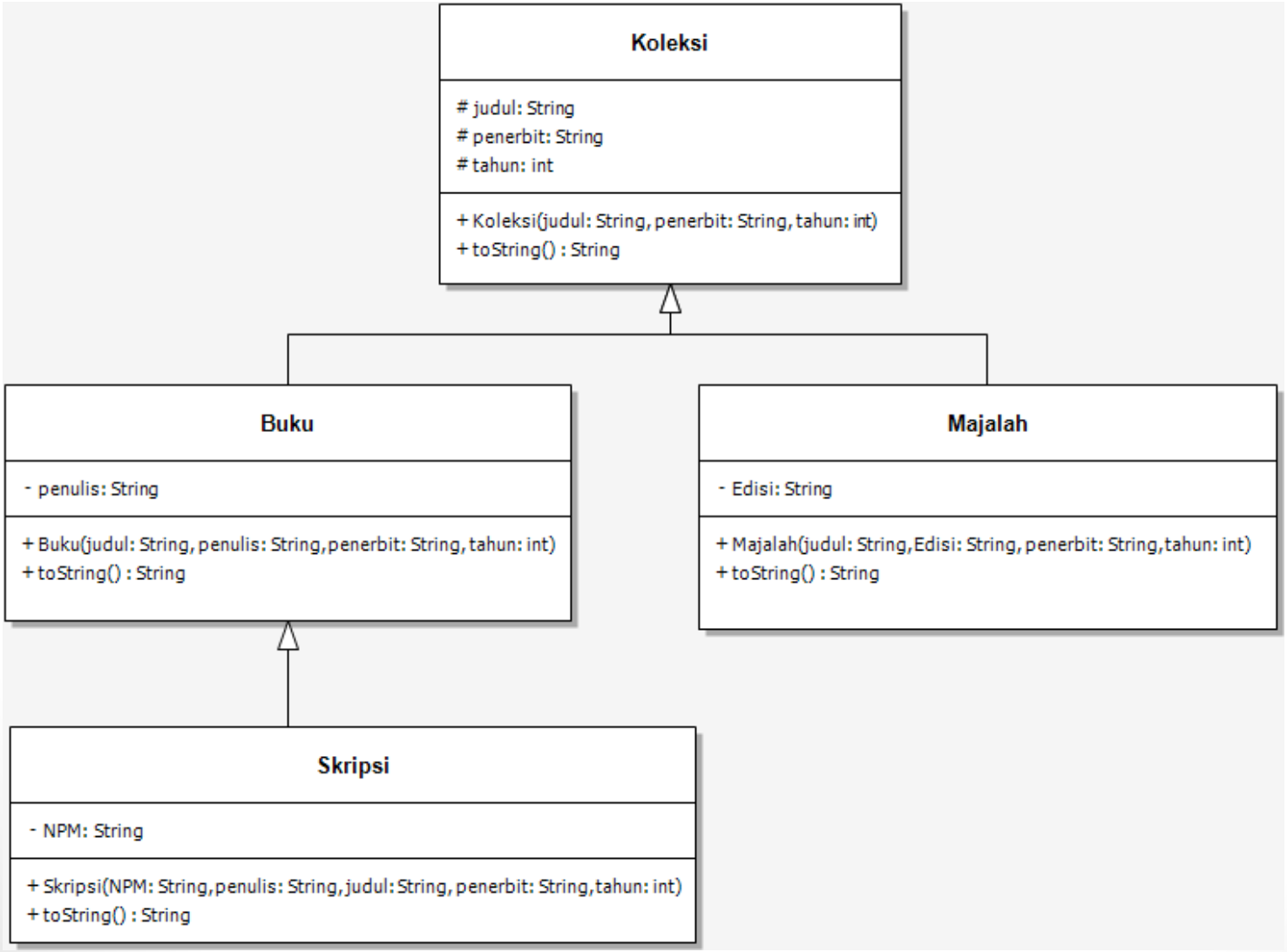


- Contoh *hybrid inheritance* pada Java dengan menggunakan *interface*

```
interface A
{
     public void methodA();
}
interface B extends A
{
     public void methodB();
}
interface C extends A
{
     public void methodC();
}
class D implements B, C
{
    public void methodA()
    {
         System.out.println("MethodA");
    }
    public void methodB()
    {
         System.out.println("MethodB");
    }
    public void methodC()
    {
         System.out.println("MethodC");
    }
    public static void main(String args[])
    {
         D obj1= new D();
         obj1.methodA();
         obj1.methodB();
         obj1.methodC();
     }
}
```

Latihan

- Buat program Java untuk class seperti pada diagram class berikut:



Next Week

- Pengenalan Java Greenfoot
- Release tugas Final Project
- Kisi-kisi UTS