

## Inheritance

### Source Code

```
class Pet
{
    protected String name;
    protected int age;
}

class Dog extends Pet
{
    private String breed;

    public void kahol()
    {
        Console.println("Kaw! Kaw!");
    }
}

class Cat extends Pet
{
    public void ngyaw()
    {
        Console.println("Ngyaw! Ngyaw!");
    }
}

class Tiger extends Cat
{
    public void ngyaw()
    {
        Console.println("Rawr! Rawr!");
    }
}

class Snake
{
    private int color;

    public void slither()
    {
        Console.println("Ssss! Ssss!");
    }
}
```

ang **Dog** at **Cat** anak ni **Pet**

ang **Tiger** anak ni **Cat** at apo ni **Pet**

method overloading:  
napapatungan ng **ngyaw()** ni **Tiger** ang **ngyaw()** ni **Cat**

Walang kinalaman ang **Snake** sa ibang classes



## Access Modifiers

<b>public</b>	<ul style="list-style-type: none"> <li>● Nakikita ng lahat; Pwedeng baguhin ng lahat ang laman ng field at pwedeng gamitin ng lahat ang method.</li> <li>● Iwasang gawing public ang mga <b>fields</b>; Gawing private ang mga fields.</li> </ul>
<b>private</b>	<ul style="list-style-type: none"> <li>● Sa loob lang ng class ang pwedeng gumamit.</li> <li>● Iwasang gawing public ang mga <b>fields</b>; Gawing private ang mga fields.</li> </ul>
<b>protected</b>	<ul style="list-style-type: none"> <li>● Mga method lang ng class at mga anak (at apo) ng class ang pwedeng gumamit.</li> <li>● Hwag nyo munang gamitin ang protected. Kapag may <b>inheritance</b>, saka nyo lang isipin kung kinakailangang gawing <b>protected</b> ang mga fields.</li> </ul>
<i>(wala)</i>	<ul style="list-style-type: none"> <li>● Paminsan-minsang tinatawag na <b>Default, Friendly</b> o <b>Package</b>.</li> <li>● Lahat ng nasa parehong <b>package</b> ang pwedeng gumamit. Hindi muna natin paguusapan kung ano ang <b>package</b>.</li> <li>● <u>Iwasan ang access modifier na ito</u>. Gawing private, protected o public. Kung hindi mo pa alam kung ano ang gagamitin, <b>private</b> ang <b>fields</b> at <b>public</b> ang <b>methods</b>.</li> </ul>

### Nakikita / Magagamit ba?

Modifier	Parehong Class	Parehong Package	Child Class ( <i>minamana</i> )	Lahat ng iba
public	oo	oo	oo	oo
protected	oo	oo	oo	hinde
<i>(wala)</i>	oo	oo	hinde	hinde
private	oo	hinde	hinde	hinde

- Hindi muna nating paguusapan kung ano ang **package**. Para sa atin, ang lahat ng nasa isang directory ay nasa isang package.

## Overloading

Tandaan:

- Sa **method overloading** pareho ang pangalan ng methods ng Parent class at Child class pero iba ang arguments. Hindi pwedeng i-overload ang Constructors.
- Sa class inheritance meron ding **field overloading**. Kung pareho ang pangalan ng field ng Child class at field ng Parent class, natatakpan ng field ng Child class ang field ng Parent class. Ang nakikita ng Child ang sarili niyang field; hindi na nakikita ang field ng Parent.
- Gamitin ang **super** keyboard para makita / magamit ang fields ng methods ng Parent class.

Halimbawa

```
public class Sample
{
    public static void main(String[] args)
    {
        Pet[] zoo;
        zoo = new Pet[4];

        zoo[0] = new Cat("Muning");
        zoo[1] = new Dog("Bantay");
        zoo[2] = new Bird("Polly");
        zoo[3] = new Dog("Tagpi");

        for(int i = 0; i < 4; i++)
        {
            zoo[i].speak();
        }
    }
}
```

Tinatanggap ng zoo ang mga **Dog**, **Cat** at **Bird** dahil **Pet** din sila.

```
class Pet
{
    protected String name;

    public Pet(String n)
    {
        name = n;
    }

    public void speak()
    {
        System.out.println("heller");
    }
}
```

```
class Dog extends Pet
{
    public Dog(String n)
    {
        super(n);
    }

    public void speak()
    {
        System.out.println("kahol! kahol!");
    }
}
```

Ang constructor ng **Dog** tinatawag ang constructor ng **Pet**.

**Overloaded method**

```
class Cat extends Pet
{
    public Cat(String n)
    {
        super(n);
    }

    public void speak()
    {
        System.out.println("ngyaw! ngyaw!");
    }
}
```

Ang constructor ng **Cat** tinatawag ang constructor ng **Pet**.

**Overloaded method**

```
class Bird extends Pet
{
    public Bird(String n)
    {
        super(n);
    }

    public void speak()
    {
        System.out.println("lipad! lipad!");
    }
}
```

Ang constructor ng **Bird** tinatawag ang constructor ng **Pet**.

**Overloaded method**