**Polymorphism**

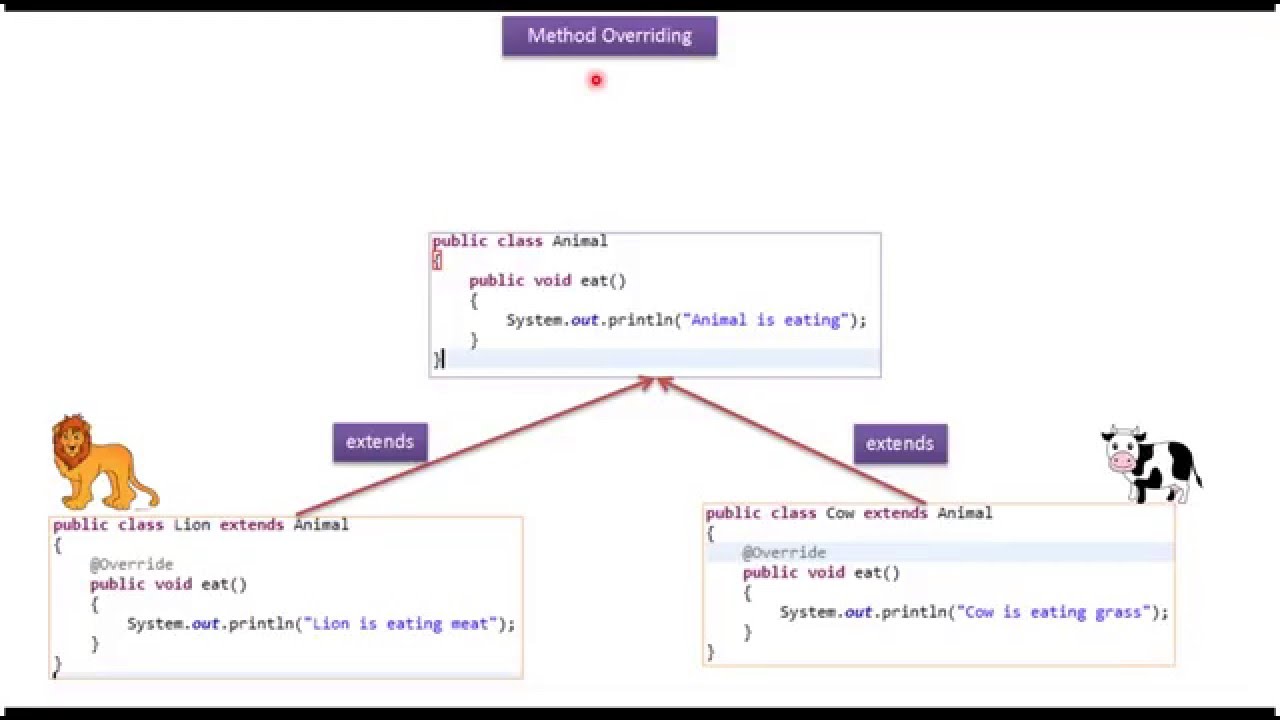
In the previous session we learnt about inheritance and come to know that inheritance enables the subclass to pass down the good traits of its parent class. But it also has its limitations when you want the subclass to develop its unique features. For example, when the subclass inherits its parent class who is able to eat grass, the subclass can eat nothing but grass, which means it must be a herbivore. In other words, carnivores will not be able to inherit this class, which is unresonable in nature.

Is there any way to keep the good traits of the parent class and at the same time let the subclass develops its own abilities? *Polymorphism* will help you reach this goal!

**Definition:**

As one Chinese saying goes "Each of the nine sons of the dragon is different from each other". It is similar in polymorphism: one behavior has many forms or formats.

Take the Animal case for reference, if the parent class Animal has an ability of eating but hasn't been defined the things it can eat, when its subclass (eg. a cow) inherits the Animal class, this subclass will eat grass; but a subclass of lion will eat meat. For different subclasses (cow and lion), the eating behavior can be shown in different forms. Such phenomenon is called polymorphism.



**Implementation of polymorphism**

Before we learn to implement polymorphism, we should first understand "override":

**Override:**

Overriding allows a subclass to provide a specific implementation of a method that is already provided by its parent class by using the keyword of Override. For example, the parent class - Animal class is only capable of eating. But its subclass - lion class can eat meat, which reimplements the process of eating.

*The key of overriding: The appearance (name, parameters, returned value) of method remains the same, while its core is reimplemented!*

There is another related concept *overload:* Please refer to below link:

1. [Override vs overload](http://www.runoob.com/java/java-override-overload.html)
2. [Method Overloading and Overriding in Java](https://www.baeldung.com/java-method-overload-override)

**Grammar of polymorphism**

*Grammar:*

**class** **ParentClass** {

**public** **void** **method**() {

*// code*

}

}

**class** **SubClass** **extends** **ParentClass** {

**@Override**

**public** **void** **method**() {

*// code*

}

}

*Example:*

**public** **class** **Animal** {

**public** **void** **eat**() {

System.out.println("I can eat!");

}

}

**public** **class** **Cow** **extends** **Animal** {

**@Override**

**public** **void** **eat**() {

System.out.println("I am a vegetarian!");

}

}

**public** **class** **Lion** **extends** **Animal** {

**@Override**

**public** **void** **eat**() {

System.out.println("I am a carnivore!");

}

}

**public** **class** **Test** {

**public** **static** **void** **main**(String[] args) {

Animal cow = **new** Cow();

cow.eat();

Animal lion = **new** Lion();

lion.eat();

}

}

*Running result:*

*// I am a vegetarian!*

*// I am a carnivore!*

**Notes on polymorphism:**

* + Subclass is parent class(is-a):\*

Take above example as reference, since cows and lions are all animals, when creating the instances of cow and lion, we can use animals to define them, namely the subclass is its parent class; but the other way around is wrong because we cannot say that "An animal is a cow." Therefore in the inheritance system, a subclass should be more specific than its parent class, so do the instances. This kind of relationship in Java is called "Is-A".

1. *The forming conditions of polymorphism:*

* Inheritance exists: Without inheritance, there will be no polymorphism.
* Override: The subclass should override the method of its parent class.
* The parent class use variable pointing to its subclass' instances: namely the above point - subclass is parent class or the relationship of "IS-A".