**Inheritance**

**The introduction of inheritance**

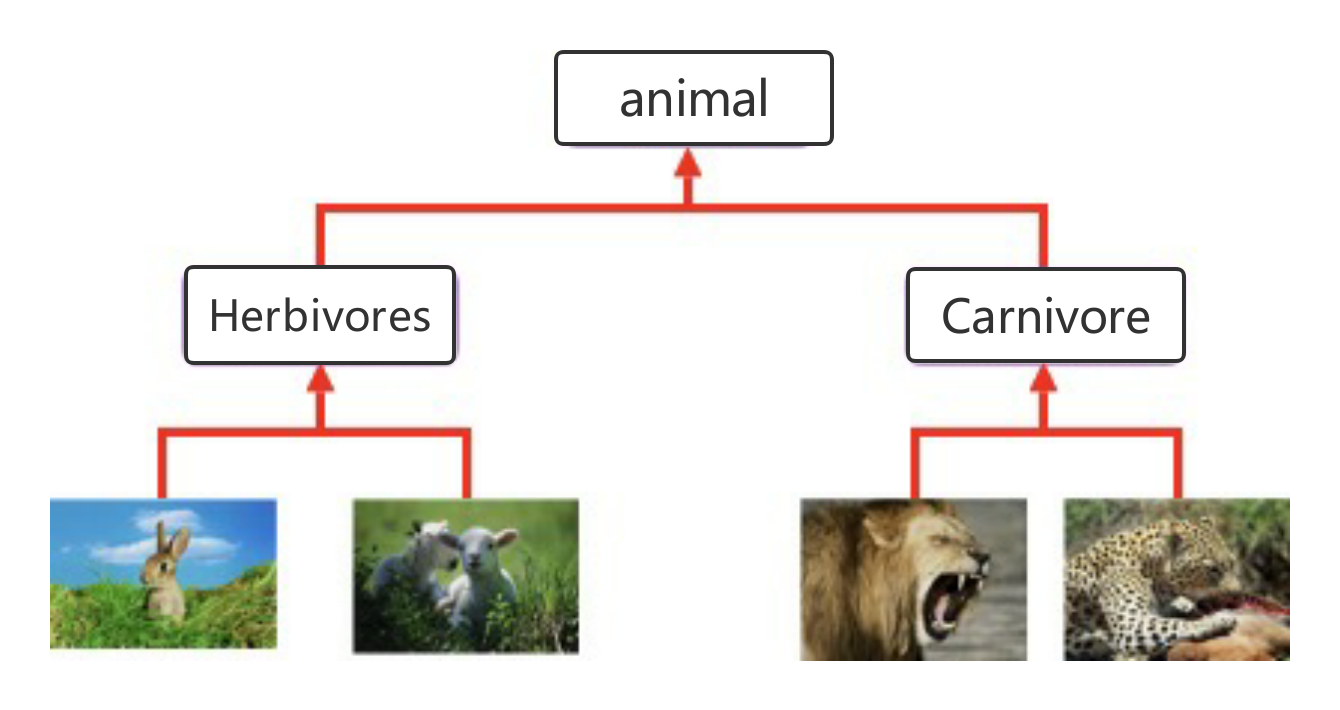
In above chapter we mentioned two classes of Person and Student and found both of them have the attributes of name and age and each attribute has both get and set methods.These same attribute and same methods create repetitions. It is known that repetitive code logic may cause extra efforts in maintaining. Is there any way we can reduce this kind of repetitions? Inheritance is one of the best solutions.

Now we will introduce in detail what is inheritance:

**Definition**

Inheritance is a common phenomenan in nature. As a Chinese saying goes, like father like son, which means a child will inherit the characteristcs and behavior of its parents and can inherit and pass down their good traits.

**Inheritance in nature:**



Software design learns from the heritance phenomenon in nature: a subclass inherits the features and behavior of its parents to obtain its parents' attributes and methods; or a subclass inherits methods from its parents to take on the same behavior.

**Implementation of inheritance:**

In Java, we can use extends keywords to declare a class is inherited from another one. It usually takes below forms:

*Grammar:*

**class** **ParentClass** {

}

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

}

*For example:*

**public** **class** **Person** {

**protected** String name;

**protected** **int** age;

**private** **int** money；

**public** **Person**(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** String **getName**() {

**return** name;

}

**public** **void** **setName**(String name) {

**this**.name = name;

}

**public** **int** **getAge**() {

**return** age;

}

**public** **void** **setAge**(**int** age) {

**this**.age = age;

}

}

**public** **class** **Student** **extends** **Person** {

**private** **int** studentNumber;

**public** **Student**(String name, **int** age) {

**super**(name, age);

}

**public** **int** **getStudentNumber**() {

**return** studentNumber;

}

**public** **void** **setStudentNumber**(**int** studentNumber) {

**this**.studentNumber = studentNumber;

}

}

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

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

Student student = **new** Student("zhangsan", 18);

student.setStudentNumber(101);

System.out.println(student.name);

System.out.println(student.getStudentNumber());

}

}

*Notes:*

Student inherits the Person class so that it obtains the attributes of name and age owned by Person. After initialization, Student will has name and age. In this way, Student not only cut out the repetitive codes in Person, but also inherits its good tradition.

**Advantages of inheritance:**

* Inheritance expands the functions of class: the subclass will get its parent class' methods and attributes once inherited from parent class.
* Inheritance can eliminate repetitive logic.

**Limitations of inheritance**

1. *One subclass can onnly inherit one parent class, which results in the limitations of inheritance.*

Wrong statements:

**class** **A** {}

**class** **B** {}

**class** **C** **extends** **A**, **B** {}

The above codes try to realise multiple inheritance, where one subclass is expected to inherit a number of classes' functions. However above statements are invalid. Actually you can try another way to realise this implementation.

The right statement should be:

**class** **A** {}

**class** **B** **extends** **A** {}

**class** **C** **extends** **B** {}

*Conclusion: Java only allows multi-level inheritance instead of multiple inheritance.*

1. *A subclass can only inherit the non-private attributes and behavior of a parent class:*

For example, as "money" in above Person class is private field (field modified by the key word of private) which is owned exclusively by the parent class, it cannot be used by its subclass. So Student class cannot access the money attributes of its parent class.

1. *A subclass can call (implicitly or explicitly) rather than inherit the constructor of its parent class:*

* If a parent class has parameters, its subclass' constructor should use the keyword of super explicitly to call the parent class' constructor and be attached with an appropriate list of parameters.
* If a parent class doesn't have parameters, its subclass' constructor doesn't need to use the keyword of super to call the parent class' constructor. The system will automatically call the parent class' constructor without parameters.

Take the above example:

**public** **Student**(String name, **int** age) {

**super**(name, age);

}

1. *Every class in Java is derived from the Object class:*

Object class is the base-class of all Java classes, which is present in java.lang package and provides methods to be used by other classes. We will learn more about Object class later in this course.

**To find more on inheritance, please read below:**

* [inheritance](https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html)
* [Detailed explanations on the keywords of this and super](https://www.jianshu.com/p/bc732beac4f9)
* [To understand this and super in Java from the perspective of a dumbhead](https://www.jianshu.com/p/89b5d62728da)