**Java 继承**

继承是java面向对象编程技术的一块基石，因为它允许创建分等级层次的类。继承可以理解为一个对象从另一个对象获取属性的过程。

如果类A是类B的父类，而类B是类C的父类，我们也称C是A的子类，类C是从类A继承而来的。在Java中，类的继承是单一继承，也就是说，一个子类只能拥有一个父类

继承中最常使用的两个关键字是extends和implements。

这两个关键字的使用决定了一个对象和另一个对象是否是IS-A(是一个)关系。

通过使用这两个关键字，我们能实现一个对象获取另一个对象的属性。

所有Java的类均是由java.lang.Object类继承而来的，所以Object是所有类的祖先类，而除了Object外，所有类必须有一个父类。

通过过extends关键字可以申明一个类是继承另外一个类而来的，一般形式如下：

// A.java

public class A {

private int i;

protected int j;

public void func() {

}

}

// B.java

public class B extends A {

}

以上的代码片段说明，B由A继承而来的，B是A的子类。而A是Object的子类，这里可以不显示地声明。

作为子类，B的实例拥有A所有的成员变量，但对于private的成员变量B却没有访问权限，这保障了A的封装性。

**IS-A关系**

IS-A就是说:一个对象是另一个对象的一个分类。

下面是使用关键字extends实现继承。

public class Animal{

}

public class Mammal extends Animal{

}

public class Reptile extends Animal{

}

public class Dog extends Mammal{

}

基于上面的例子，以下说法是正确的：

* Animal类是Mammal类的父类。
* Animal类是Reptile类的父类。
* Mammal类和Reptile类是Animal类的子类。
* Dog类既是Mammal类的子类又是Animal类的子类。

分析以上示例中的IS-A关系，如下：

* Mammal IS-A Animal
* Reptile IS-A Animal
* Dog IS-A Mammal

因此 : Dog IS-A Animal

通过使用关键字**extends**，子类可以继承父类所有的方法和属性，但是无法使用 private(私有) 的方法和属性。

我们通过使用instanceof 操作符，能够确定Mammal IS-A Animal

**实例**

public class Dog extends Mammal{

public static void main(String args[]){

Animal a = new Animal();

Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal);

System.out.println(d instanceof Mammal);

System.out.println(d instanceof Animal);

}

}

以上实例编译运行结果如下：

true

true

true

介绍完**extends**关键字之后，我们再来看下**implements**关键字是怎样使用来表示IS-A关系。

**Implements**关键字使用在类继承接口的情况下， 这种情况不能使用关键字**extends**。

**实例**

public interface Animal {}

public class Mammal implements Animal{

}

public class Dog extends Mammal{

}

**instanceof 关键字**

可以使用 **instanceof** 运算符来检验Mammal和dog对象是否是Animal类的一个实例。

interface Animal{}

class Mammal implements Animal{}

public class Dog extends Mammal{

public static void main(String args[]){

Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal);

System.out.println(d instanceof Mammal);

System.out.println(d instanceof Animal);

}

}

以上实例编译运行结果如下：

true

true

true

**HAS-A 关系**

HAS-A代表类和它的成员之间的从属关系。这有助于代码的重用和减少代码的错误。

**例子**

public class Vehicle{}

public class Speed{}

public class Van extends Vehicle{

private Speed sp;

}

Van类和Speed类是HAS-A关系(Van有一个Speed)，这样就不用将Speed类的全部代码粘贴到Van类中了，并且Speed类也可以重复利用于多个应用程序。

在面向对象特性中，用户不必担心类的内部怎样实现。

Van类将实现的细节对用户隐藏起来，因此，用户只需要知道怎样调用Van类来完成某一功能，而不必知道Van类是自己来做还是调用其他类来做这些工作。

Java只支持单继承，也就是说，一个类不能继承多个类。

下面的做法是不合法的：

public class extends Animal, Mammal{}

Java只支持单继承（继承基本类和抽象类），但是我们可以用接口来实现（多继承接口来实现）,脚本结构如：

public class Apple extends Fruit implements Fruit1, Fruit2{}

一般我们继承基本类和抽象类用extends关键字，实现接口类的继承用implements关键字。