《面向对象》之继承（上）

笔记本:	JAVA		
创建时间:	2018/7/29 16:37	**更新时间:**	2018/8/15 12:11
作者:	Debao Xu		

继承：

好处：
1、提高了代码的复用性
2、让类与类之间产生了关系，为第三个特征“多态”提供了前提
Java支持单继承，不直接支持多继承
单继承：一个类只能有一个父类
多继承：一个类可以有多个父类，Java并不支持

什么时候定义继承？

当事物之间存在着所属关系（xxx是yyy中的一种）时，可以通过继承来体现这个关系，即 xxx extends yyy，也就是说yyy中的所有属性、行为在xxx中都能够有所体现。

场景

学生的属性有：年龄、性别，行为有：学习；
工人的属性有：年龄、性别，行为有：工作；
从上面可以知道学生和工人共有的属性是 年龄和性别，因此考虑将学生和工人的共享代码向上抽取到一个共性的类中。
这种实现手段就是继承（extends）。

```
class Student extends Person  // 继承的关键字 extends
{
	//String name;
	//int age;
	void study() {System.out.println(age+" "+name+" "+"Good Study!");}
}
class Worker extends Person
{
	//String name;
	//int age;
	void work() {System.out.println("Work Hard!");}
}
class Person
{
	String name;
	int age;
}
public class StudentDemo
{
	public static void main(String[] args)
	{
		Student S = new Student();
		S.name = "Rex";
		S.age = 26;
		S.study();
	}
}
```

“子父”类出现后，代码上的一些特点：

一、成员变量　二、成员函数　三、构造函数
1、成员变量，当子父类中出现了同名的成员变量，则用关键字super来区分，见下面的代码：

```
class Fu
```

```
{
	int num = 4;
}
class Zi extends Fu
{
	int num = 5;
	void show()
	{
		int num = 6;
		System.out.println(num);//用这行代码，最后显示num的值为6
	    //System.out.println(this.num);用这行代码，最后显示num的值为5,因为在本类中要是出现了局部变量和成员变量同名，则通过this关键字来区分
	    //System.out.println(super.num);用这行代码，最后显示num的值为4,因为在子父类中要是出现了变量同名，则通过super关键字来区分
	}
}
public class ExtendsDemo
{
	public static void main(String[] args)
	{
		new Zi().show();
	}
}
```

2、成员函数，当子父类中出现了一模一样的方法时，子类对象运行的是子类的方法，这种情况称之为覆盖（返回值类型，函数名，参数列表都一样），举例“覆盖”的应用，

```
class phone
{
	void call() {}  //功能1
	void sendMeg() {}  //功能2
	void show()     //功能3
	{
		System.out.println("number");
	}
}
class newphone extends phone
{
	 //在父类中已经有show()方法了，在子类中再有新功能时，保留父类中的show()方法的名称，只对方法的内容做出改变
	void show()
	{
		super.show();// 用关键字super来调用父类中的同名方法，达到覆盖的效果
		System.out.println("name");
		System.out.println("picture");
	}
}
public class phoneDemo
{
	public static void main(String[] args)
	{
		new newphone().show();
	}
}
运行结果:
number
name
picture
```

上述代码中“父类”phone已经有了3项功能，“子类”newphone继承了这3项功能，并且在功能3 ” show（） “中有了新的内容，【注意】：子类和父类相比，子类中仍然需要 show() 方法，因此我们不要去再定义一个不同的函数名，而是就

用这个名字 show() ，对于父类中show() 的内容用 super关键字继承过来。总之，” 覆盖 “就是一句话，函数名不变，只是改变了函数中的内容。

覆盖使用的注意事项：
子类方法覆盖父类方法时，必须要保证子类方法的访问权限大于等于父类方法的访问权限。
静态方法只能覆盖静态方法。或者被静态方法覆盖

3、构造函数，先看一个例子

```
class phone
{
	phone() //父类中的构造函数
	{
		System.out.println("phone");
	}
}
class newphone extends phone
{
	newphone() //子类中的构造函数
	{
		//super();  这个是"隐形"的（默认存在的），即在访问子类中的构造函数之前，先访问父类中的构造函数
		System.out.println("newphone");
	}
}
public class phoneDemo
{
	public static void main(String[] args)
	{
		new newphone();//创建对象
	}
}
```

运行结果：

```
phone
newphone
```

子父类中构造函数的特点如下：

1、由上面的代码就会发现，创建子类对象时（new newphone();），父类中的空参数构造函数也运行了，这是因为所有的构造函数的第一行默认都有一个隐式的super (); 语句。

2、在调用本类中的构造函数用 this (实参列表) 语句；调用父类中的构造函数用super(实参列表)语句；

3、为什么子类对象初始化都要访问父类中的构造函数呢?
因为子类继承了父类中的内容，所以创建对象时必须要先看父类是如何对内容进行初始化的。

【注意】：当父类中**没有空参数构造函数**时，子类需要通过**显示定义super**语句指定要访问的父类中的构造函数
【注意】：用来调用父类构造函数的super语句在子类构造函数中必须定义在第一行，因为父类的初始化要先完成

```
class phone
{
	phone(int x)
	{
		System.out.println("phone"+x);
	}
}
class newphone extends phone
{
	newphone(int y)
	{
		super(4); //当父类中没有空参数的构造函数时，要显式指定super的方式来访问父类中的构造函数
		System.out.println("newphone"+y);
	}
}
```

```
public class phoneDemo
{
	public static void main(String[] args)
	{
		new newphone(5);
	}
}
```

问题：
1、this和super用于调用构造函数，可以同时存在吗？
答：不可以，因为他们都要定义在第一行
2、为什么要定义在第一行？
答：因为初始化动作要先执行

子类的实例化过程，代码如下：

```
class Person  //父类
{
	private String name;
	private int age;
	Person(String name,int age)
	{
		this.name = name;
		this.age = age;
	}
	public void setName(String name)
	{
		this.name = name;
	}
	public String getName()
	{
		return name;
	}
	public void setAge(int age)
	{
		this.age = age;
	}
	public int getAge()
	{
		return age;
	}
}
class Student extends Person  //子类
{
	Student(String name,int age)
	{
		super(name,age);//通过super来调用父类中的构造函数Person(String name,int age)来完成初始化
	}
}
public class StudentDemo
{
	public static void main(String[] args)
	{
		Student s = new Student("Rex",26);
		System.out.println(s.getName());
		System.out.println(s.getAge());
	}
}
```

final关键字

1、final修饰符，可以修饰类，修饰方法，修饰变量（局部变量或者成员变量）
2、final修饰过的类不可以被继承

3、final修饰过的方法不可被覆盖
4、final修饰过的变量会变成一个常量，并且只能赋值一次。当使用的数据不变时，需要定义阅读性强的名称来表示该数据，并将该数据final化。被final修饰的变量，名称规范是：所有字母都大写，如果由多个单词组成，需要通过下划线"_"进行分隔。
5、final的**静态变量值**必须在声明或静态初始化程序中（非初始化程序中不需要赋值）**赋值，**例如，

```
public class Test
{
    static final int X = 1;  //final修饰的X位于初始化程序中，需要赋初值
    public void test2(final int X) //final修饰的X没有位于初始化程序中，不需要赋初值
    {
        //........
    }
}
```

举例：圆周率的值是不变的为3.14，所以在定义时，将它final化，定义如下：

```
final double PI = 3.14;
```

```
final class Person
{
。。。。
}
class Student extends Person //这句话是错的！因为Person被final修饰了，就不能再被继承了
{
       Student(String name,int age)
       {
             super(name,age);//通过super来调用父类中的构造函数Person(String name,int age)
       }
}
```

```
 public static final int AGE = 27; //AGE为全局常量，public表示对外可以被访问；static表示不需要创建对象，直接用类名就可以访问；final表示将其常量化
```