2、具体内容

子类与父类一旦产生了继承关系之后，实际上子类会继承父类中的全部定义，但是这里面也有可能出现不合适的场景。子类如果发现与父类中设计不足并且需要保留父类中的方法或者属性名称的情况下就会发生覆写。

■方法覆写

当子类定义了与父类方法名称相同，参数类型及个数完全相同（跟父类方法一模一样）的时候，就称为方法的覆写。

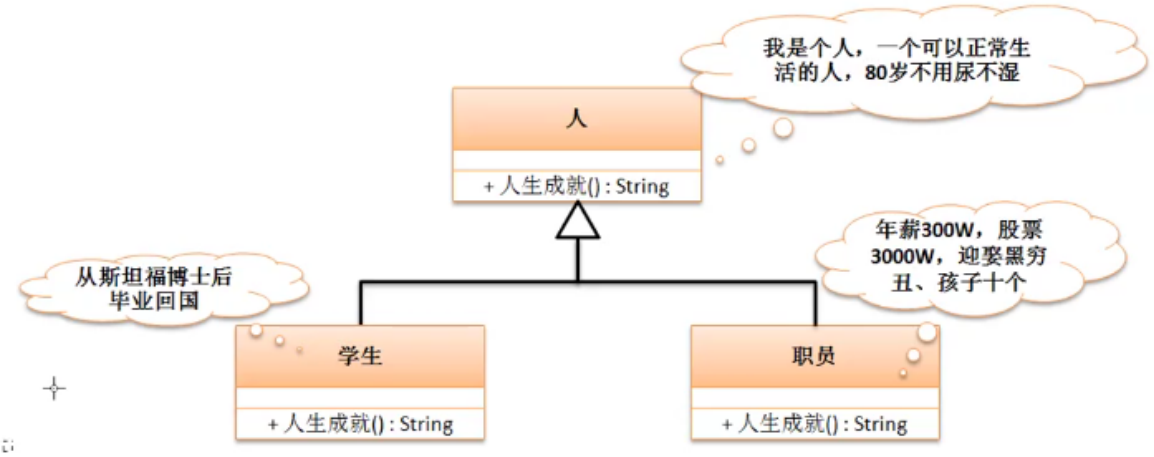
范例：观察方法覆写

```
class Channel {
	public void connect() {
		System.out.println("【Channel父类】进行资源的连接。") ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		Channel channel = new Channel() ;
		channel.connect() ;
	}
}
```

```
class Channel {
	public void connect() {
		System.out.println("【Channel父类】进行资源的连接。") ;
	}
}
class DatabaseChannel extends Channel {	// 要进行数据库连接
	public void connect() {	// 保留已有的方法名称，而后进行覆写
		System.out.println("【子类】进行数据库资源的连接。") ;
	}
}
public class JavaDemo {
```

```
    public static void main(String args[]) {
        DatabaseChannel channel = new DatabaseChannel() ;
        channel.connect() ;
    }
}
```

由于现在实例化的是子类（DatabaseChannel）对象，所以此时调用的方法一定是被子类所覆写过的方法，如果该方法没有被覆写过，那么将调用父类中提供的方法，覆写的意义是在于：优化父类的功能。



在子类进行方法覆写之后如果现在要想继续调用父类中的方法，那么就必须使用“super()方法”

```
class Channel {
    public void connect() {
        System.out.println("【Channel父类】进行资源的连接。") ;
    }
}
class DatabaseChannel extends Channel {     // 要进行数据库连接
    public void connect() {    // 保留已有的方法名称，而后进行覆写
        super.connect() ;    // 直接调用父类中的方法
        System.out.println("【子类】进行数据库资源的连接。") ;
    }
}
public class JavaDemo {
    public static void main(String args[]) {
        DatabaseChannel channel = new DatabaseChannel() ;
        channel.connect() ;
    }
}
```

只要是在子类中调用父类方法的时候一定要在方法前追加有“super.()方法”

■方法覆写限制

虽然利用方法的覆写可以更好的扩充父类的功能，但是对于覆写也是有其自身要求的：被覆写的方法不能够拥有比父类方法更为严格的访问控制权限。

对于访问控制权限现在已经接触过三种：public > default（不写） > private，private权限是最小的，也就是说如果你此时父类中的方法中使用了default权限定义，那么子类定义该方法的时候只能够使用public或default定义，如果父类中的方法使用了public定义，那么子类中方法只能够使用public定义。

范例：观察错误的覆写

```
class Channel {
	public void connect() {
		System.out.println("【Channel父类】进行资源的连接。");
	}
}
class DatabaseChannel extends Channel {	// 要进行数据库连接
	void connect() {//保留已有的方法名称，而后进行覆写
		System.out.println("【子类】进行数据库资源的连接。");
	}
}
```

此时父类的方法使用了public定义，而子类的方法使用了default权限，所以权限更加严格了，那么这种覆写就有错误了。

```
JavaDemo.java:7: 错误: DatabaseChannel中的connect()无法覆盖Channel中的connect()
        void connect() {//保留已有的方法名称，而后进行覆写

  正在尝试分配更低的访问权限; 以前为public
1 个错误
```

当时既然说到了权限问题久必须考虑一下private权限，private除了可以定义在属性上也可以定义在方法上。

```
class Channel {
	private void connect() {
		System.out.println("【Channel父类】进行资源的连接。");
	}
	public void fun() {
		this.connect();		// 调用本类方法
	}
}
class DatabaseChannel extends Channel {	// 要进行数据库连接
	// 此时并不是一个覆写，因为父类的connect()方法不可见，那么这个方法对于子类而言就相当于是一个新定义的方法
	public void connect() {
		System.out.println("【子类】进行数据库资源的连接。");
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		DatabaseChannel channel = new DatabaseChannel();
		channel.fun();
	}
}
```

在以后实际开发之中只要是定义方法，95%情况下都是用public，所以覆写的时候最好也考虑使用public。

面试题：请解释Override与Overloading区别？Overloading时返回参数是否相同？

NO.	区别	Override	Overloading
1	中文含义	重载	覆写
2	概念	方法名称相同，参数的类型及个数不同	方法名称、参数类型及个数、返回值相同
3	权限	没有权限限制	被覆写方法不能拥有更严格的控制权限
4	范围	发生在一个类中	发生在继承关系类中

在进行方法重载的时候并没有对返回值类型做出限制，但是好的习惯应该保持返回类型的一致。

■属性覆盖

当子类定义了与父类相同名称的成员时候就称为属性覆盖。

范例：观察属性覆盖

```
class Channel {
	String info = "www.mldn.cn" ;
}
class DatabaseChannel extends Channel {	// 要进行数据库连接
	String info = "HelloMLDN" ;	// 名称相同
	public void fun() {
		System.out.println(super.info) ;
		System.out.println(this.info) ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		DatabaseChannel channel = new DatabaseChannel() ;
		channel.fun() ;
	}
}
```

如果说这个时候按照标准的开发属性进行了封装，这个时候实际上子类和父类中的私有属性没有什么关系了，即便名称一样，也只是相当于定义了一个新的属性而已。

```
class Channel {
	private String info = "www.mldn.cn" ;
	public String getInfo() {
		return this.info ;
	}
}
class DatabaseChannel extends Channel {	// 要进行数据库连接
	String info = "HelloMLDN" ;	// 名称相同
	public void fun() {
		System.out.println(this.info) ;
		System.out.println(super.getInfo()) ;
	}
}
```

```
public class JavaDemo {
	public static void main(String args[]) {
		DatabaseChannel channel = new DatabaseChannel() ;
		channel.fun() ;
	}
}
```

面试题：请解释super与this的区别？

• 在程序类中使用this表示先从本类查找所需要的属性或方法，如果本类不存在则查找父类定义，如果使用super则表示不查找子类，直接查找父类；

• this与super都可以进行构造方法的调用，但是this()调用的是本类构造，而super()是由子类调用父类构造，两个语句都必须放在构造方法的首行，所以不能够同时出现；

• this可以表示当前对象；

2、具体内容

final在程序之中描述的是种终接器的概念，在java里面使用final关键字可以实现如下的功能呢：定义不能够被继承的类、定义不能够被覆写的方法、常量。

范例：使用final定义不能被继承类

```
final class Channel {}	// 这个类不能够有子类
```

```
final class Channel {	// 这个类不能够有子类
}
class DatabaseChannel extends Channel {	// 错误的
}
public class JavaDemo {
	public static void main(String args[]) {
	}
}
```

当子类继承了父类之后实际上是可以进行父类中方法覆写的，但是如果你现在不希望某一个方法被子类所覆写就可以使用final来进行定义。

范例：定义不能够被覆写的方法

```
class Channel {       // 这个类不能够有子类
    public final void connect() {}
}
class DatabaseChannel extends Channel {     // 错误的
    public void connect() {}
}
public class JavaDemo {
    public static void main(String args[]) {
    }
}
```

在有一些系统开发之中，可能使用1表示开关打开、使用0表示开关关闭，但是如果说现在要是直接操作0或者1会造成混乱，所以就希望可以通过一些名称来表示0或者1。在final关键字里面有一个重要的应用技术：可以利用其定义常量，常量的内容一旦定义则不可修改。

```
class Channel {       // 这个类不能够有子类
    private final int ON = 1 ; // ON就是常量
    private final int OFF = 0 ;       // OFF就是常量
    public final void connect() {
        ON = 2 ;
    }
}
public class JavaDemo {
    public static void main(String args[]) {
    }
}
```

范例：使用final定义常量

```
class Channel {       // 这个类不能够有子类
    private final int ON = 1 ; // ON就是常量
    private final int OFF = 0 ;       // OFF就是常量
}
```

实际上常量往往都是公共的定义，所以为了可以体现出共享的概念，往往会使用一种全局常量的形式来定义，使用public static final来定义全局常量。

```
    public static final int ON = 1 ;     // ON就是常量
    public static final int OFF = 0 ;    // OFF就是常量
```

在定义全局常量的时候每一个字母必须大写表示。

范例：观察一个程序代码

```
public class JavaDemo {
    public static void main(String args[]) {
        final String info = "mldn";
        String strA = "www.mldn.cn";
        String strB = "www."+info+".cn";
        System.out.println(strA==strB);
//加final之后变成true
    }
}
```

在方法的时候也可以使用final来定义参数，此时也表示一个常量的概念。