**今天完成的事情：**

**第一部分：JAVA基础知识**

**学习预览：**

* 继承
* 重写与重载（Override/Overload）

1. **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

* 继承就是子类继承父类的特征和行为
* 使子类对象（实例）具有父类的实例域和方法
* 或子类从父类继承方法，使得子类具有父类相同的行为

1. **类的继承格式**

在JAVA中通过extends关键字来申明一个类是另一个类的继承

形式如下：

**class 父类 { }**

**class 子类 extends 父类 { }**

1. **继承类型**

* **单继承**

public class A｛｝

class class B extends A｛｝

* **多重继承**

public class A｛｝

public class B extends A｛｝

public class C extends B｛｝

* **不同类继承一个类**

public class A｛｝

public class B extends A｛｝

public class C extends A｛｝

* **多继承**

extends是不允许多继承的

而implements允许类多继承，但是父类必须为接口interface型

例如：

public interface A｛｝

public interface B｛｝

public class C implements A，B｛｝

1. **继承关键字**

* **extends**

extends为继承关键字

* **implements**

implements为扩展关键字，允许多继承特性

1. **this与super关键字**

* **this关键字**

指向自己的引用

引用当前对象的属性，而不是方法内的参数

* **super关键字**

通过super对父类成员（对象或方法）的访问，用来引用当前对象的父类

1. 构造器

* 子类不继承父类的构造器（构造方法或构造函数），只是调用（隐式或显式）
* 如果父类的构造器带有参数，则必须在子类的构造器中，显式地通过super关键字来调用父类的构造器，并配以适当的参数列表
* 如果父类的构造器不带参数，则在子类的构造器中不需要使用super关键字来调用父类的构造器，系统会自动调用父类的无参构造器

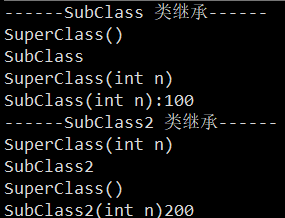
示例如下：





根据自己的理解添加注释

结果为



1. **重写与重载（Override/Overload）**
2. **重写**

重写是子类通过extends关键字继承父类中的方法，再对继承的方法进行修改的方式

重写只允许对继承方法的实现过程进行修改

**在用方法构建对象时，类名仍是父类名，但是重写对象名为继承类名**

例如



**执行过程：**

* **new Dog()语句，先把**

**特性：**

* **返回值和形参都不能改变**
* **在编译阶段，只检查参数的引用类型**
* **然而在运行时，Java虚拟机(JVM)指定对象的类型并且运行该对象的方法**

**应用范围：**

* 修改算法，修改功能逻辑

**好处：**

* 可以根据自己的需要，定义特殊的行为

既子类可以根据需要，修改父类方法的实现过程

**重写规则：**

* 参数列表必须完全与被重写方法的相同

返回类型必须完全与被重写方法的返回类型相同

* 访问权限不能比父类中被重写的方法的访问权限更低。如果父类的一个方法被声明为public，那么在子类中重写该方法就不能声明为protected
* 父类的成员方法只能被它的子类重写
* 声明为final的方法不能被重写

声明为static的方法不能被重写，但是能够被再次声明

* 子类和父类在同一个包中，那么子类可以重写父类所有方法，除了声明为private和final的方法

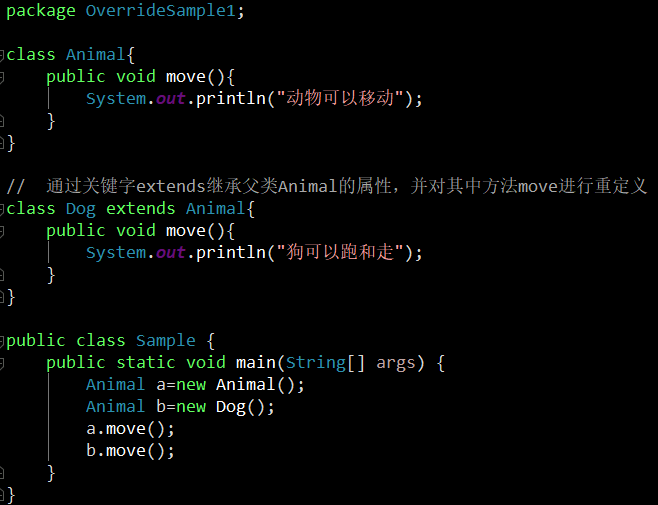
子类和父类不在同一个包中，那么子类只能够重写父类的声明为public和protected的非final方法

* 重写的方法能够抛出任何非强制异常，无论被重写的方法是否抛出异常

但是，重写的方法不能抛出新的强制性异常，或者比被重写方法声明的更广泛的强制性异常，反之则可以

* 构造方法不能被重写
* 如果不能继承一个方法，则不能重写这个方法

**实例如下：**



由于Animal与Dog中都有move方法，所以可以编译成功

但是考虑到重写的运行要求，还有一种情况要考虑

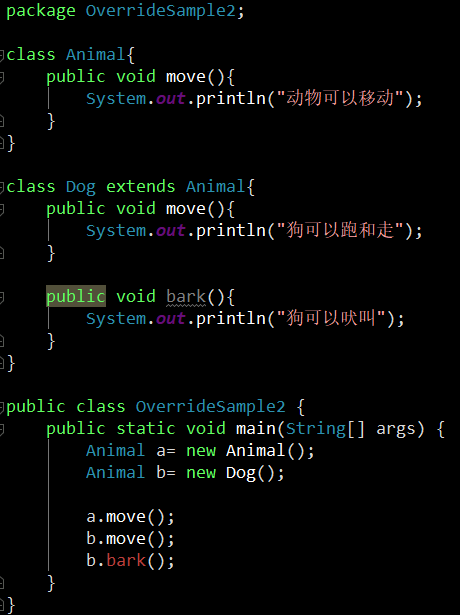
**这里有一个创建对象的知识：** **向上转型**

既由子类Dog（）为模版创建的对象b，转换为了父类Animal的类型

既子类引用的对象转换为父类类型

这种方式叫做向上转型

**实例如下：**



以下是错误提示：



该程序抛出一个编译错误，因为b的引用类型Animal没有bark方法

1. **重载**

重载(overloading) 是把类中的方法，按照不同格式进行多次声明，使其具有多态性

* **在一个类里面，方法名字相同，而参数不同，返回类型可以相同也可以不同**
* **每个重载的方法（或者构造函数）都必须有一个独一无二的参数类型列表**

最常用的地方就是构造器的重载

**重载规则：**

* 被重载的方法必须改变参数列表(参数个数或类型不一样)
* 被重载的方法可以改变返回类型
* 被重载的方法可以改变访问修饰符
* 被重载的方法可以声明新的或更广的检查异常
* 方法能够在同一个类中或者在一个子类中被重载
* 无法以返回值类型作为重载函数的区分标准

实例：



1. **重写与重载的区别**



1. **总结**

方法的重写（Override）和（Overloading）是JAVA多态性的不同表现

* **重写是父类与子类之间多态性的一种表现**
* **重载可以理解成多态的具体表现形式**

明天计划完成的事情：

由于明天后天请假，所以下个工作日要学习的内容有

1. JAVA基础语法知识

* 多态
* 抽象类
* 封装
* 接口
* 包

1. Spring框架知识的理解

* Bean的使用
* AOP机制

遇到的问题：

看完JAVA反射机制的教学后，有看见评论说，最好不要使用JAVA反射机制来编程

好奇是因为什么原因，不能在工作环境下使用JAVA反射机制的编程思想呢？

我看反射机制可以很好的减少类与对象及方法间的耦合性，安全性高具有很好优势

为什么不能使用呢

收获：

今天学习了继承与重写和重载的概念