Java Day 07

创建初始化对象过程

在栈中分配空间，存放引用变量

在堆中分配空间，存储对象；对象属性进行默认初始化

引用变量指向堆地址

使用成员变量值进行属性赋值

调用构造器方法

构造器方法：一种特殊的方法，构建类对象时被调用

和类同名

没有返回值类型

同一类中可以有多个构造器（重载）

可以通过 this(...)调用本类其它构造器;必须是构造器中的第一条语句

如果没有提供构造器，系统会提供一个默认构造器(无参、空体)；

如果提供了构造器，系统将不会提供默认构造器；建议自己在类中都加一个无参的构造器

继承：多个类中存在相同属性和行为时，将这些内容抽取到单独一个类中，那么多个类无需再定义这些属性和行为，只要继承那一个类即可。其中，多个类可以称为子类，单独那一个类称为父类、超类（superclass）或者基类。

继承的好处和弊端

好处：

提高了代码的复用性

提高了代码的维护性

让类与类之间产生了关系，是多态的前提

弊端

类的耦合性增强了

开发的原则：高内聚，低耦合

耦合：类与类的关系

内聚：就是自己完成某件事情的能力

什么时候使用继承

继承体现的是一种关系："is a"。

Person：Student、Teacher

水果：苹果、香蕉、橘子

采用假设法：如果有两个类 A、B，只要他们符合 A 是 B 的一种，或者 B 是 A 的一种，就可以考虑使用继承。

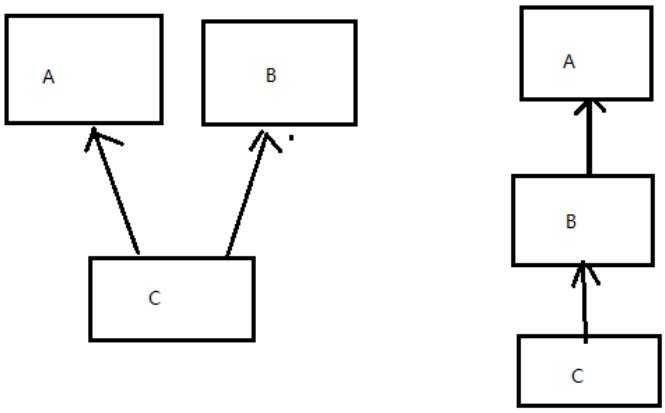
Java 中类的继承特点-->SingleExtends.java

Java 只支持单继承，不支持多继承

Java 支持多层继承(继承体系)

如果想用这个体系的所有功能，用最底层的类创建对象

如果想看这个体系的共性功能，看最顶层的类



继承中成员变量和成员方法的关系-->ExtendsTest2.java

继承中成员变量的关系

不同名的变量

同名的变量

继承中成员方法关系

不同名的方法

同名的方法

先在本类中寻找，没有再到父类中寻找；可以通过 this./super. 访问本类、父类中的成员

方法重写-->OverriddenTest.java

什么是方法重写：子父类出现了一模一样的方法

1）父子类间

2）相同的方法名、参数列表、返回值类型

3）可见性不能被缩小

4）异常不能被扩大

什么时候用：

当子类需要父类的功能，而子类又有自己特有内容时，可以重写父类中的方法。这样，既沿袭了父类的功能，又定义了子类特有的内容。

方法重写的注意事项：

1）父类中私有方法不能被重写

因为父类私有方法子类根本就无法继承

2）子类重写父类方法时，访问权限不能更低，最好一致

3）父类静态方法，子类也必须通过静态方法进行重写

其实这个算不上方法重写，但是现象确实如此，至于为什么算不上方法重写，多态中我会讲解(静态只能覆盖静态)

子类重写父类方法的时候，最好声明一模一样。

继承中构造方法的关系

构造方法不能被继承

子类中所有的构造方法默认都会调用父类中无参数的构造方法

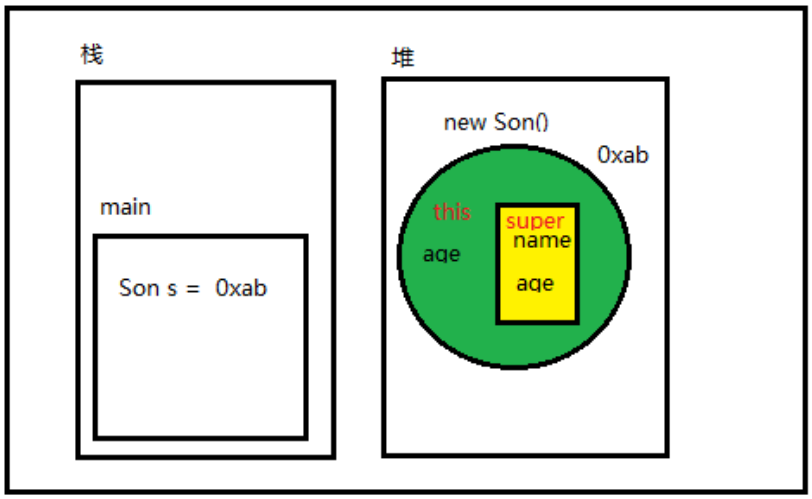
为什么呢?

因为子类会继承父类中的数据，可能还会使用父类的数据

所以，子类初始化之前，一定要先完成父类数据的初始化

子类构造方法的第一条语句默认都是：`super()`

`java.lang.Object` 类最顶层的父类



继承中构造方法的注意事项

父类没有无参构造方法,子类怎么办?

子类构造器中通过 super(...)解决

super(...)或者 this(...)必须是构造方法的第一条语句

---

this 和 super 的区别和应用

---

this 和 super 都代表什么

this:代表当前对象的引用,谁来调用我,我就代表谁

super:代表当前对象父类的引用

this 和 super 的使用区别

调用成员变量

this.成员变量: 调用本类的成员变量,也可以调用父类的成

员变量

super.成员变量：调用父类的成员变量

b:调用成员方法

this.成员方法：调用本类的成员方法,也可以调用父类的方法

super.成员方法：调用父类的成员方法

c:调用构造方法

this(...)：调用本类的构造方法

super(...)：调用父类的构造方法

多态

生活中，比如跑的动作，小猫、小狗和大象，跑起来是不一样的。再比如飞的动作，昆虫、鸟类和飞机，飞起来也是不一样的。可见，同一行为，通过不同的事物，可以体现出来的不同的形态。

概念：

同一类域的不同对象在调用同一方法的时候表现不同-->PolymorphismTest.java

同一类域的含义:是指继承同一父类或实现同一接口的类的集合

多态存在的三个必要条件：

1）要有继承

2）要有方法重写

3）父类引用指向子类对象

代码

父类类型 变量名 = new 子类对象;

变量名.方法名();

当使用多态方式调用方法时，首先检查父类中是否有该方法，如果没有，则编译报错；如果有，执行的是子类重写后的方法。

多态中的成员访问

1）成员变量：编译看左边（父类），运行看左边（父类）

所有的成员变量取决于编译时类型

2）成员方法：编译看左边（父类），运行看右边（子类）

所有的成员方法取决于运行时类型

3）静态方法：编译看左边（父类），运行看左边（父类）

所有的静态方法取决于编译时类型

类型转换

向上转型：多态本身是子类类型向父类类型向上转换的过程，这

个过程是默认的

```
Aninal a1 = new Cat();
```

向下转型：父类类型向子类类型向下转换的过程，这个过程是强制的

```
if(a1 instanceof Cat)
   Cat c = (Cat)a1;
```

为什么要转型?

当使用多态方式调用方法时，首先检查父类中是否有该方法，如果没有，则编译错误。也就是说，不能调用子类拥有，而父类没有的方法。编译都错误，更别说运行了。所以，想要调用子类特有的方法，必须做向下转型。

多态的好处和弊端

好处:

提高了代码的维护性（继承来保证）

提高了代码的扩展性（多态来保证），父类当作形式参数，接收任意子类对象

弊端：不能直接使用子类的特有属性和行为