**学习总结**

姓名\_\_董金伟\_ 专业\_计算机科学与技术\_ 班级\_2班\_\_ 日期\_\_2022/1/19\_\_\_

1. 当天所学习主要内容（做一总结）

**子类构造方法**

super：

* 在子类构造器方法中调用 super 方法，指定调用父类的构造方法。
* super 方法必须是第一条可执行语句。

this：

* 在类构造方法中调用 this 方法，指定调用该类的某个构造方法。
* this 方法必须是第一条可执行语句。

不要再子类中对父类成员初始化。

**final 关键字**

final 数据成员：不可以被更改值

final 方法成员：不可以被重写

final 类：可以以被继承

**使用继承注意事项**

1. 尽量隐藏父类的内部结构，吧仅为辅助的方法设为 private。
2. 如果外部类步行往被继承，再设为 final。
3. 尽量不要在父类的构造方法中调用可能被重写的方法。

**抽象类**

如果一个类没有包含足够的信息描述一个具体的对象，则这个类就是抽象类。

abstract class 类名{ 数据成员； 方法成员； }

抽象类：不能实例化抽象对象；

作用：用来被继承，从而实现多态。

注意事项：

* 可以定义抽象类变量。
* 不能实例化抽象对象。
* 抽象类中又构造方法。

**抽象方法**

abstract 返回类型 方法名(形参表); //没有方法体，分号结束

抽象方法：没有方法体。不能被调用。只能在抽象类中定义。

作用：用来被重写，从而实现多态。

注意事项：

* 构造方法和静态方法不能是抽象方法。
* 抽象类可以又抽象方法，也可以又普通方法。
* 父类中如果又抽象方法，如果子类没有重写父类的抽象方法，则子类也为抽象方法。

# 多态

## 一、多态的概念和好处

多态：相同的引用变量，调用同一个方法，变量引用的对象不同， 则调用的方法不同。

好处：代码简练有利于后期维护修改

## 二、实现多态的方法

1. 继承
2. 对方法重写
3. 使用父类调用方法

## 三、方法重写

在子类对父类方法重写。

访问权限 修饰符 返回类型 方法名 形参表 抛出异常

重写

* 访问权限 >= 父类
* 返回类型相同，或子类
* 方法名相同
* 形参相同
* 抛出异常 <= 父类

|  | **访问权限** | **修饰符** | **返回类型** | **方法名** | **形参表** | **抛出异常** |
| --- | --- | --- | --- | --- | --- | --- |
| 重写 | 大于等于 | 相同或子类 | 相同 | 相同 | 小于等于 | 父类子类 |
| 重载 | 无关 | 无关 | 相同 | 不同 | 无关 | 同类 |

final 方法、static 方法、构造方法：不能重写；可以重载；

## 四、复制兼容性原则

向上转型：将子类型转换为父类型

* 系统自动进行向上转型。
* 父类型变量可以引用继承了该父类的子类对象。
* 本质：父累变量引用的子类型对象中的父类型对象。
* 总结：通过父类型变量只能访问父类型中的成员
* 作用：实现多态

向下转型：将父类型转换为子类型

* 强制类型转换，方法：（子类型）父类型变量引用的对象
* 父类型变量引用的是该子类对象时，才可以进行向下转型，否则，抛出异常。
* instanceof 运算符： 对象 instanceof 类型名
* 作用：使用子类中特有的方法

## 五、多态注意事项

数据成员不会实现多态。

静态方法成员不会实现多态。

**抽象的作用**

抽象类、抽象方法，在软件开发中都是设计层面的概念（what to do）

非抽象类、非抽象方法实现具体的功能。（how to do）

1. 学习中所存在的问题及解决办法

问题：1. 子类重写了父类的方法，然后用父类对象调用这个方法。

解决：1. 子类重写父类的方法，父类在调用这个方法的时候，调用的是重写以后的方法，因为重写之后，内存种的方法表就发生了变化，所以，调用父类的方法，依然是重写之后的方法。

3．学习收获及建议

子类的构造方法中，可以使用super关键字访问父类的数据成员与构造方法。this可以访问自己类的数据成员和构造方法。但是super和this都必须在第一条语句执行。所以，使用了super就不可以使用this了。

final 数据成员：不可以被更改；final 方法成员：不可以被重写；final 类：可以以被继承

使用abstract关键字修饰的类是抽象类，修饰方法，就是抽象方法，继承了抽象类，就要实现抽象类中的抽象方法。抽象类不的抽象方法不需要具体实现。

多态：使用继承，方法重写，调用父类的方法来实现多态。

方法重写：

|  | **访问权限** | **修饰符** | **返回类型** | **方法名** | **形参表** | **抛出异常** |
| --- | --- | --- | --- | --- | --- | --- |
| 重写 | 大于等于 | 相同或子类 | 相同 | 相同 | 小于等于 | 父类子类 |
| 重载 | 无关 | 无关 | 相同 | 不同 | 无关 | 同类 |

复用性原则中，有向上转型（将子类型转换为父类型），实现多态，类似自动类型转换程序自己执行，不需要手动设置；和向下转型（将父类型转换为子类型），使用类中的特有方法，类似强制转换，需要用（类名）这样的方式来实现。