多态

多态：对象在不同时刻表现出来的不同状态。

前提条件：

A:要有继承关系。

B:要有方法重写。

C:要有父类引用指向子类对象。

(没有方法的重写也行,只不过那样就没有意义了)

多态就是想让父类的引用类调用调用子类重写的方法

方法有重写，而变量没有

```
class Fu
{
    public void show()  {        System.out.println("fu show");  }
}

class Zi extends Fu    //继承
{
    public void show()    {    System.out.println("zi show"); }    //重写
}

class DuoTaiDemo
{
    public static void main(String[] args)
    {
        Fu f = new Fu();//f 是 Fu 的引用

        Zi z = new Zi();//new Zi()是 Zi 的对象

        Fu fu = new Zi();//多态    父类的引用指向子类对象
    }
}
```

多态就只是调用子类的重写方法，其他不改变

多态间的成员特点：方法有重写。而变量没有。

A:成员变量

编译看左边,运行看左边。

B:成员方法

编译看左边,运行看右边。

弊端：强耦合（父类若重新定义了新方法，则子类强制性继承）

父类的引用不能使用子类的特有功能（即子类的其他没有重写的功能）--- 解决：下转型

基本类型：隐式转换(小到大)，强制转换(大到小)。

引用类型：向上转型(小到大)，向下转型(大到小)。

常见异常：类型转换异常（ClassCastException），

-----因为下转型之后再赋给另一个类（动物转到猫，在另转成狗就类型转换错误，因为已经转成猫了）

多态的好处：提高代码的扩展性和可维护性。

抽象

（强制要求子类实现抽象方法，不能直接实例化对象）

抽象方法：只有方法声明，没有方法体。

抽象类：只要有抽象方法的类

抽象类的特点：1、用 abstract 修饰

2、有抽象方法的必须为抽象类或为接口。（抽象类不一定非要有抽象方法）

3、抽象类不能实例化（即不能创建对象）

4、子类必须重写抽象方法，要不该子类也得规定为抽象类

抽象类的：强制规定子类必须重写抽象方法!

抽象类的成员特点：

A:成员变量： 可以有成员变量，也可以有常量。

B:构造方法： 有构造方法的。----用于对父类数据进行初始化。

C:成员方法： 可以有抽象方法，也可以有非抽象方法。

抽象方法是为了要求子类做某些事情。

非抽象方法是为了提高代码复用性，被子类继承。

接口（interface）

（）

接口 --- 所有方法都是抽象方法

接口与类之间是实现关系----implements

接口多态 --- 与一般的多态一样

接口的特点：

A:接口不能被实例化。

B:接口中的方法： 要么被子类重写 要么子类也是抽象类

接口的成员特点：

成员变量：接口中只有常量 --- 默认修饰符： public static final

构造方法：没有构造方法。

成员方法：接口中的方法都是抽象的 ----- 默认修饰符：public abstract

所有类都直接或间接继承至 Object 类

- 类与类关系：继承关系。而且只能单继承，可以多层继承。
- 类与接口的关系：实现关系。可以单实现，也可以多实现。

  并且还可以在继承一个类的同时，实现多个接口。
- 接口与接口的关系：继承关系。可以单继承，也可以多继承。

接口：对外暴露的原则、是程序的功能拓展、降低了耦合性