抽象类和接口

回顾

今天任务

```
1. 抽象类
2. 抽象方法
3. final
4. 接口
```

教学目标

```
1. 掌握抽象类
2. 掌握抽象方法
3. 掌握final关键字
4. 掌握接口
5. 掌握抽象类和接口的区别
```

第一节： 抽象类

1.1 抽象方法的由来

> 设计一个方法
>
> ```
> public void fun();
> ```
>
> 以上方法程序报错，有两种解决方案：
>
> ```
> add body：以前的做法
> change fun to abstract：将方法设置成抽象的
> ```
>
> 将方法设置成抽象之后，类也报错了 将类设置成抽象的

1.2 abstract 关键字

> 关键字：abstract：抽象
>
> 可以修饰方法：叫做抽象方法，没有方法体，需要使用分号表示声明结束 可以修饰类：叫做抽象类，抽象方法所在的类必须是抽象类

1.3 抽象方法

> 使用abstract关键字修饰，只表示声明了一个方法，但是没有任何的实现

> 特点： 没有方法体，需要使用分号表示方法声明结束 抽象方法所在的类必须是抽象类 抽象方法的存在就是为了被子类重写

1.4 抽象类：

> 使用abstract关键字修饰 特点： 1）抽象类的出现由于类中有抽象方法，但是抽象类中也可以没有抽象方法 2）抽象类中有构造方法，但是不能创建对象，构造方法目的在子类中会被调用 3）抽象类的存在就是为了被继承，抽象类的非抽象子类中必须把抽象类中用的所有抽象方法全部实现 在子类给继承来的抽象方法添加方法体相当于是重写父类的抽象方法

1.5 抽象类和普通类的区别：

> 1）抽象类需要abstract，而普通类不需要 2）构造方法：都有，但是抽象类不能实例化对象，普通类可以 3）成员方法：抽象类中可以存在抽象的成员方法也可以存在非抽象成员方法，而普通类中只能存在非抽象成员方法

> 思考：final和abstract是否可以连用？
>
> 1）两个关键字修饰方法时，final修饰的方法特点：可以被继承不能被重写；abstract修饰的方法特点：必须被重写；所以这两个关键字不能同时修饰同一个方法
>
> 2）两个关键字修饰类时：final修饰的类特点：不能被继承；abstract修饰的类特点：必须被继承；所以这两个关键字不能同时修饰同一个类
>
> 综上所述：final和abstract不可以连用
>
> final的类中能否有abstract方法？不能 abstract类中能否有final方法？可以

第二节：final关键字

2.1 final关键字

```
final 最终的
```

2.2 能够修饰哪些

```
1. 可以修饰全局变量
2. 可以修饰局部变量
3. 可以修饰方法
4. 可以修饰类
```

2.3 final修饰 全局变量和局部变量：常量

```
// final 是可以修饰 属性的     修饰的属性必须  在声明的时候就赋值
    final static int age =4;
    public static void main(String[] args) {
        //final 可以修饰局部变量, 在声明的时候可以不赋值
        final String name = "abc";
        //  name = "abc";
        // final 修饰的局部变量只能被赋值一次
        // final 修饰 的 全局变量  只能被赋值一次
        //age = 9;
        System.out.println(age);
        System.out.println(name);
    }
```

因为被final修饰的变量只能赋值一次，所以把final修饰的变量叫做常量

2.4 final修饰方法：最终方法

```
// final 修饰的额 方法 是 最终的方法,不允许 重写
public final void eat(){
    System.out.println("吃");
}
```

2.5 final修饰的类： 最终类

```
public final class Person{
  // final 可以修饰类 但是 修饰的类不能被继承
}
```

第三节： 接口

> 生活中的接口：USB，插座...
>
> USB：物理：必须满足USB接口的宽度和厚度 内部：需要遵守磁片的个数
>
> 插座：首先需要满足插头的个数 满足电压和电流一些规则

3.1 接口的概念

> 规定了一类事物的一个模板，想使用某个接口的一类对象都需要满足的规定
>
> 接口：是类的模板
>
> 接口就是特殊的抽象类
>
> 接口特殊在：一般情况下接口中的方法都是抽象方法

3.2 接口的特点：

> 1）接口不能创建对象，而且接口中没有构造方法； 2）接口中的方法一般都是共有抽象方法：public abstract 3）接口中所有的属性都是共有静态常量属性：pulbic static final
>
> 在一个接口中声明方法时，若没有声明访问权限，默认也是public，若没其他修饰默认也是abstract；声明属性时，若没有声明访问权限和静态常量，默认也是public static final

```
public abstract void fun();
接口中可以使用一下格式声明方法：
void fun();
public static final int a = 9;
int a = 9;
```

3.3 接口的声明语法

> 关键字interface：表示一个接口，接口interface和类class是平级关系

```
语法：
public interface 接口名{
    //接口中的成员：抽象方法和静态常量
}
```

3.4接口的实现类

> 接口与类的关系：implements
>
> 语法：

```
pulbic interface I{
    public abstract void fun();
}

public class Impl implements I {
    public void fun(){}
}
```

> 思考：一个类实现类某个接口之后，是否可以存在父类？ 可以，实现接口和继承类不冲突
>
> 注意：若一个类有父类同时也实现接口，声明类时，必须先继承再实现接口
>
> 语法：

```
public class Dog extends Animal implements I{
    public void fun(){}
}
```

3.5 接口的分类：

> 1）普通接口：在接口中可以声明抽象方法，和静态常量属性 2）常量群接口：在接口中只声明一组静态常量属性 3）标志性接口：在接口中没有抽象方法，也没有静态常量，作用为了标记某个类具有某个功能

3.6 接口和接口的关系：

> 继承关系：使用关键字extends实现接口与接口的继承关系

> 接口继承的特点： 1）接口中没有私有成员，所以父类接口中的所有成员都会被子接口继承 2）父子关系中都是接口，所以在子接口中不需要实现任何的抽象方法 3）接口可以多继承
>
> 思考： 现有类A和类B，两个类中都有fun()方法，类C继承类A和类B，当使用类C的对象调用fun方法时，如何执行？此时不知道执行那个fun方法，所以类不可以多继承
>
> 现有接口A,B,两个接口中都有fun方法，接口C继承A,B，由于接口中的没有方法体，所在只要在接口C中存在了fun方法即可
>
> 所以接口支持多继承

3.7 接口中特殊的方法：

> 1）在接口中使用static关键字修饰的方法有方法体 静态方法需要有方法体 2）jdk1.8之后新增功能： 在接口中使用default关键字修饰的方法有方法体

3.8 接口的使用：

> 在一个平台或系统中，如果多个类都需要使用到某些方法，可以将这些方法定义到一个接口中，所有需要这些方法的类，可以实现这个接口，可以有效的关系类，有效地实现解耦

> 案例：现有交通工具类Transport类 Transport有三个子类，飞机Plane，车Car，船Ship
>
> Plane有，客机，货机，战斗机 Car有客车，货车，坦克 船有客船，货船，航母
>
> 战斗机，坦克，航母都有开火攻击的功能，通过分析，此功能不能来自于父类，所以可以讲开火攻击的功能设置在一个接口中

```java
//交通工具类
public class Transport {
    //成员方法
    public void yun(String good) {
        System.out.println("运输"+good);
    }
}
//飞机类
public class Plane extends Transport {}
//船类
public class Ship extends Transport {}
//车类
public class Car extends Transport {}
//开火攻击接口
public interface Fire {
    public abstract void fire();//开火攻击方法
}
//航母类：继承Ship，实现Fire
public class Carrer extends Ship implements Fire {
    @Override
    public void fire() {
        // TODO Auto-generated method stub
        System.out.println("xiu~~~");
    }
}
//战斗机：继承Plane，实现Fire
public class FightingPlane extends Plane implements Fire {
    @Override
    public void fire() {
        // TODO Auto-generated method stub
        System.out.println("piupiupiu~~");
    }
}
//坦克类：继承Car，实现Fire
public class Tank extends Car implements Fire {
    @Override
    public void fire() {
        // TODO Auto-generated method stub
        System.out.println("轰轰轰~~~");
    }
}
//测试
public class TestTransport {
    public static void main(String[] args) {
        FightingPlane fp = new FightingPlane();
        fp.yun("炸药");
        fp.fire();//战斗机对象的开火攻击的功能来源于Fire接口
        //可以使用接口的引用指向实现类的实例
        Fire f = new Tank();
    }

}
```

课前默写

> 1.默写单例饿汉式
>
> 2.设计一个圆形类（Circle），半径属性，计算面积、周长功能
>
> 设计一个圆柱类（Column），继承圆形类，私有高属性，求体积、表面积

作业

> 案例：有一个小型企业，有技术员、销售员、销售经理和经理四类员工，这些员工都有编号、 姓名、工资、工资级别信息， 技术员（Technology）的工资 每小时酬金（hourlyPay）与每个月工作时数（hours）积 销售员（Salesman）的工资 每个月销售提成（bonus）与销售额（sales）积 销售经理（SalesManager）工资 部门销售总额（sales）,经理提成（bonus）的乘积并加固定月薪（monthlyPay） 经理（Manager）工资 固定月薪（monthlyPay）
>
> 工资级别： 工资<3000 1 3001~5000 2 5001~8000 3 工资>8000 4
>
> 所有员工自动编号 显示每个员工的信息，格式如下： 编号是xxxx的员工（技术员、销售员、销售经理、经理）的姓名是xxx，工资是xxx，工资级别是xx

面试题

> 接口和抽象类有什么区别?