Java05-面向对象基础

Task1.对象和类

Q1：啥是面向对象（OOP）？

面向对象基本介绍

- 从字面来理解：
  - **面向**：拿、找
  - **对象**：能干活的东西
  - **面向对象编程**：那东西过来做对应的事
- 为什么要用OOP?

  ——符合人类思维习惯，编程更简单，更好理解。
  e.g.我们需要干啥，就拿什么工具（对象）。需要洗衣服就用洗衣机，需要扫地就用扫地机器人。
- OOP要学啥?
  - 学习获取已有对象并使用。例如：Scanner、Random。
  - 如果没有想用的工具——学习如何自己设计对象并使用 -> 面向对象的语法。
- **类和对象**
  - 类（设计图）：是对象共同特征的描述。
  - 对象：是真实存在的具体东西。
  - 在java中，必须先设计类，才能获得对象。
  - **类：**
    - 如何定义类：

```
public class 类名{
    1.成员变量（属性，一般是名词）
    2.成员方法（行为，一般是动词）
    3.构造器
    4.代码块
    5.内部类
}
```

*注意：

1. 一个java文件中可以定义多个class类，且只能一个是public类，public修饰的类名必须是代码文件名
2. 成员变量的完整定义格式：修饰符 数据类型 变量名 = 初始化量；（一般无需初始化，存在默认值）

数据类型	明细	默认值
基本类型	byte、short、int、long	0
基本类型	float、double	0.0
基本类型	boolean	false

数据类型	明细	默认值
引用类型	类、接口、数组、String	null

- **如何得到类的对象：**
  `类名 对象名 = new 类名();`
- 如何使用对象：
  - 访问属性：对象名.成员变量 （*注：如果在两个软件包中执行属性，需要在类定义属性的时候前面加`oublic`表示公开）
  - 访问行为：对象名.方法名(...)

类的主要构成

1. **字段（Field）**：也叫属性或成员变量，用于描述对象的状态或特征。
2. **方法（Method）**：也叫成员**函数**，用于定义对象的行为或功能，操作字段来完成特定任务。
3. **构造方法（Constructor）**：一种特殊的方法，专门用于在创建对象（实例化）时**初始化新对象。**
   - 特点：
     - 方法名必须与类名完全相同。
     - 没有返回类型（连`void`也没有）。
     - 可以重载，即一个类可以有多个参数列表不同的构造方法。

问题回答

> 1. 请你为这个Person类添加构造方法实现复制对象，并在题解附上你的Person类代码。你的构造方法用到this关键字了吗？请说说它的作用。

A1：

1. **如何使用构造方法实现复制对象：**
   - 方法的定义：

```
public Person(Person p2) {

}
```

   - 方法的调用：**在内存中创建一个新的Person对象，p2是一个全新的对象，拥有独立的内存空间**

```
Person p2 = new Person(p1);
```

2. `this`**关键字：**

   由于我们取变量名时需要做到“见名知意”，所以常常会出现变量名重复（冲突）的情况。于是，`this`常用于**解决成员变量和局部变量的命名冲突**，`this.变量名`来指代成员变量，而没有`this`前缀的则指的是局部变量。

比如：

```
public void setName(String name) { // 局部变量name
        this.name = name; // 用this.name表示成员变量，等号右边的name是局部变量
}
```

[拓展] `super`关键字用法类似，其用于指向**直接父类对象**（和后面的继承有关），主要用于在子类中访问父类的成员。

3. 还需要知道的——啥是**JavaBean**：

一言以蔽之：**JavaBean 是一个符合特定标准的 Java 类。**

其要求如下：

- 类名需要见名知意
- 成员变量需要`private`修饰
- 提供至少两个构造方法
  - 无参
  - 带全部参数的构造方法
- 成员方法
  - 提供每一个成员变量对应的setter和getter（如其名，也就是字段的赋值和返回）
  - 如果还有其他行为也需要写上

有了以上基础知识，**可得Person类代码如下：（请见注释）**

```
package Person;

public class Person {
    private String name;
    private int age;
    private int sex;//男为1，女为0

    //无参构造对象
    public Person() {
    }

    //有参构造对象----初始化
    public Person(String name, int age, int sex) {
        this.name = name;
        this.age = age;
        this.sex = sex;
    }

    //复制对象的构造方法
    public Person(Person two){
        //此处this关键字有两种写法都正确，被注释掉的是第一种写法
        //this.name = two.name;
        //this.age = two.age;
        //this.sex = two.sex;
        this(two.name, two.age, two.sex);
    }
```

```
    //以下全是getter和setter，coding时使用快捷键即可（上面的构造对象也是）
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getSex() {
        return sex;
    }

    public void setSex(int sex) {
        this.sex = sex;
    }

    //方法
    public void eat() {
        System.out.println(name+"正在吃东西");
    }

    public void sleep() {
        System.out.println(name+"正在睡觉");
    }

    public void dadoudou() {
        System.out.println(name+"正在打豆豆");
    }
}
```

> 2. 在主类的main方法中创建Person类的一个对象，并给它的字段赋值（可以用构造函数，也可以用引用变量）。说说对象和类的关系。

main类代码如下：（请看注释部分）

```
package Person;

public class PersonTest {
    public static void main(String[] args) {

        //new第一个对象p1,对其初始化赋值
```

```
        Person p1 = new Person("Glimmer", 18, 1);

        //输出字段
        System.out.println("Name: " + p1.getName());
        System.out.println("Age: " + p1.getAge());
        System.out.println("Sex: " + (p1.getSex() == 1 ? "女" :
"男"));

        //调用方法
        p1.eat();
        p1.sleep();
        p1.dadoudou();

        //new第二个对象p2，实现复制对象
        Person p2 = new Person(p1);

        //利用setter对字段重新进行赋值
        p2.setName("Zack");
        p2.setAge(19);
        p2.setSex(0);

        //输出重新赋值后的字段
        System.out.println("Name: " + p2.getName());
        System.out.println("Age: " + p2.getAge());
        System.out.println("Sex: " + (p2.getSex() == 1 ? "女" :
"男"));

        //调用方法
        p2.eat();
        p2.sleep();
        p2.dadoudou();

    }
}
```

输出结果为：

```
Name: Glimmer
Age: 18
Sex: 女
Glimmer正在吃东西
Glimmer正在睡觉
Glimmer正在打豆豆

Name: Zack
Age: 19
Sex: 男
Zack正在吃东西
Zack正在睡觉
Zack正在打豆豆
```

A2：类和对象的关系

就像最上面谈到的一样，

- 类（设计图）：是对象共同特征的描述。

- 对象：是真实存在的具体东西，是“设计图”设计出来的“房子”。

所以我们可以很容易理解，在数量上**一个类可以创建无数个对象，多个对象可以属于同一个类。**

在 Java 中，使用 `new` 关键字来根据类创建对象，这个过程也叫**实例化。**从代码角度来看，**对象 = new 类();**

所以在coding时我们也需要先有“设计图”才能“造房子”，也就是先必须有类，才能有对象。

> 3. 学习访问修饰符，为你的Person类的字段和方法添加你认为合适的访问修饰符。尝试在不同的位置（当前类，相同包的其它类，包的外部等）访问这些字段和方法，并总结出各种访问修饰符的限制范围。

在最上面的A1的`Person`类代码中，和题干不同的是，**我对题干中的方法前的`private`自己修改成了`public`**，否则无法进行方法调用，会报错——这是和访问修饰符的范围有关。

A3：

访问修饰符用于定义 Java 中类、方法、变量和构造函数的**访问权限，即它们可以在哪里被看到和使用。**Java 提供了四种访问权限级别，按从最宽松到最严格的顺序排列如下：

访问修饰符总结表

修饰符	当前类	同一包内	不同包的子类	不同包的非子类	说明
`public`	✅	✅	✅	✅	项目内完全公开
`protected`	✅	✅	✅	❌	主要提供给子类使用
`default`	✅	✅	❌	❌	包内可见，默认选项
`private`	✅	❌	❌	❌	仅当前类内部可见

所以，由于`private`只能在当前类调用，题干中的方法都在`Person`类，而main方法在`PersonTest`类，所以无法调用，因此我修改成了`public`。

Task2.类中的变量和方法

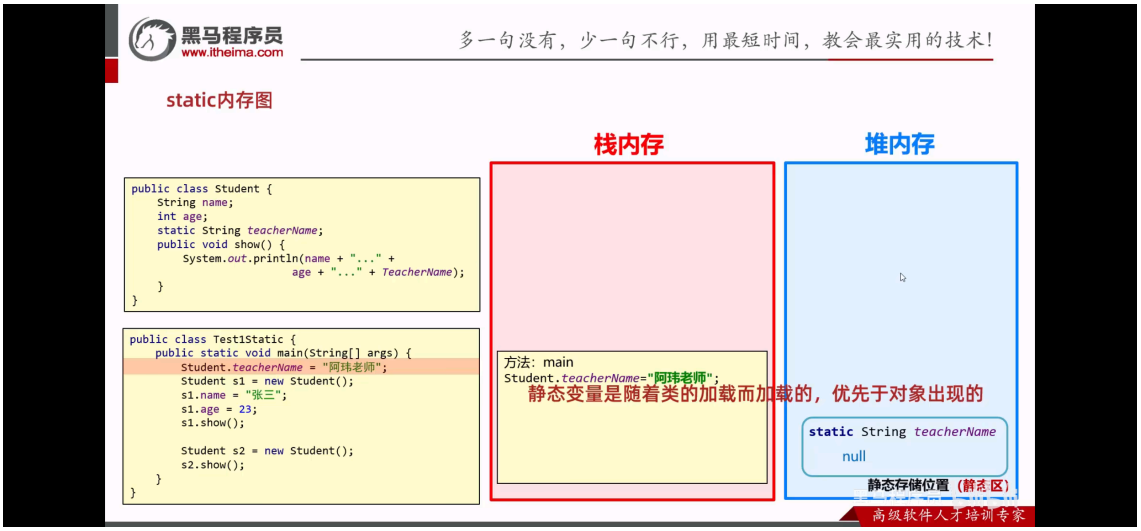
> 在上面我们知道，一个类可以有成员变量，不同对象的成员变量之间是独立且互不干扰的，比如这个Person对象的变量name值为“ISEKAI”，另一个Person对象的变量name值则可能为“Zack”。但类也可以有公共的类变量，也叫静态变量。
>
> 同理，类中也有静态方法。回想之前调用Person类的eat方法时，是不是必须先创建对象，再使用对象来调用方法？其实这里的eat方法就叫做实例方法。

我们需要理解什么是`static`关键字以及其作用。

static关键字

`static` 的意思是“**静态的**”，它控制的是这个成员的**归属和生命周期**。

- **没有 `static` 的普通成员**：属于**对象**（也称为“实例成员”）。
  - 只有当你在代码中用 `new` 关键字创建了一个对象（实例）后，这些成员才会在内存中被创建。
  - 每个对象都拥有这些成员的一份**独立拷贝**。修改一个对象的成员，不会影响另一个对象。
- **有 `static` 的静态成员**：属于**类本身**。
  - 它在程序加载类的时候就被创建并初始化，**不需要创建任何对象**。
  - 无论这个类被实例化了多少个对象，静态成员在内存中**只有唯一的一份拷贝**，所有对象共享这一份数据。
  - **static：被该类中所有对象共！享！**
- 为了方便理解，**static的内存**如图：



静态方法和实例方法

特性	静态方法	实例方法
关键字	使用 `static`	不使用 `static`
归属	属于类本身	属于对象实例
调用方式	类名.方法名()	对象名.方法名()
内存分配	类加载时分配	对象创建时分配
访问实例变量	❌ 不能直接访问	✅ 可以直接访问
使用 `this`	❌ 不能使用	✅ 可以使用
生命周期	与类相同	与对象相同

注：参考学习资料（截图来源）

Task.2 `static` 的内存图 [面向对象进阶-01-static-静态变量_哔哩哔哩_bilibili]: