



面向对象高级

写程序的套路

设计对象来处理数据解决问题

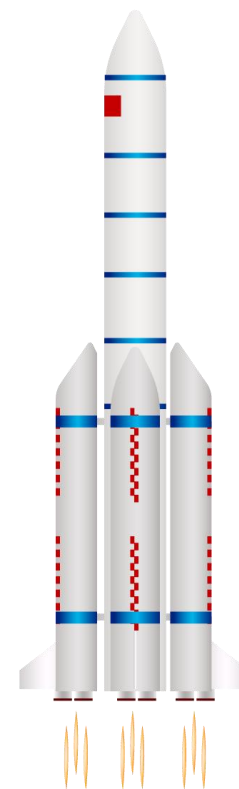


黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



面向对象高级部分的学习建议

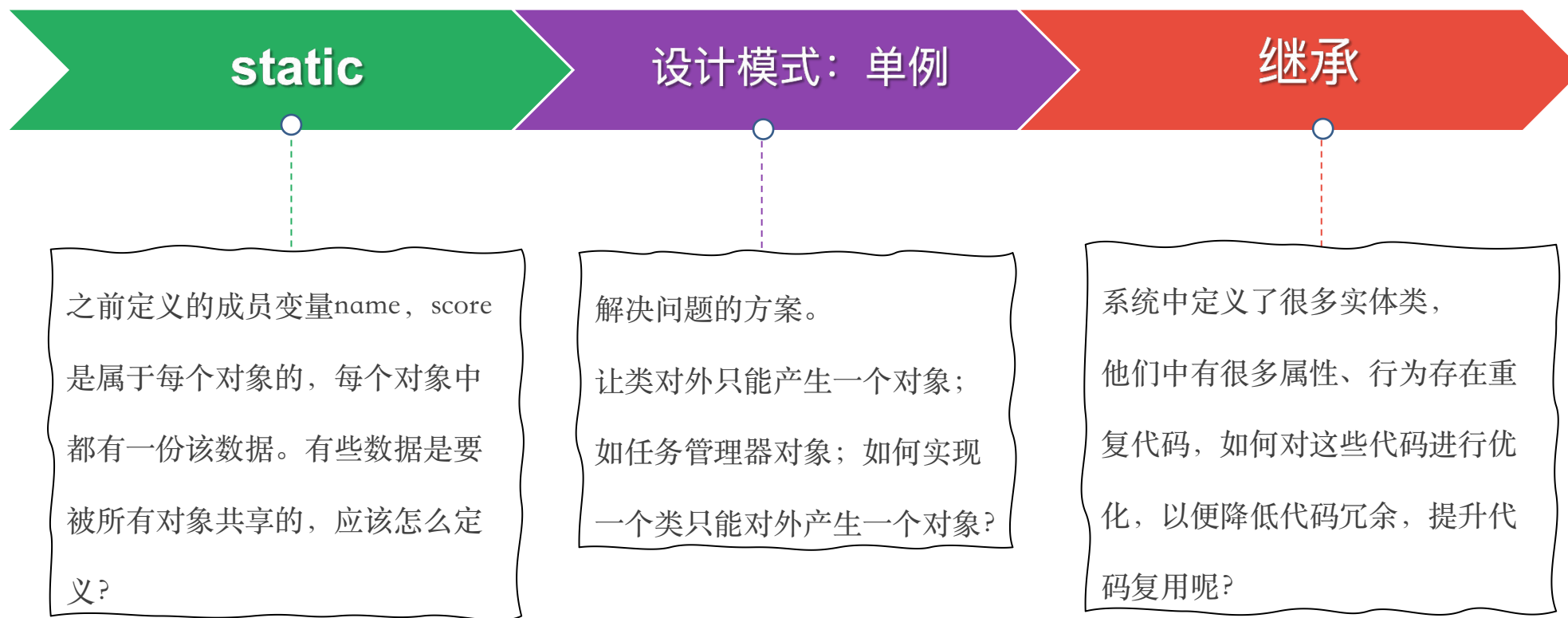


1、多关注语法的基本作用

2、多思考、练习、记忆

3、要自信，后面用起来就会了

面向对象高级第一天，同学们需要学习哪些知识？





目录

Contents

➤ static

◆ static修饰成员变量

◆ static修饰成员变量的应用场景

◆ static修饰成员方法

◆ static修饰成员方法的应用场景

◆ static的注意事项

◆ static的应用知识：代码块

◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

static

- 叫静态，可以修饰成员变量、成员方法。

成员变量按照有无static修饰，分为两种：

- 类变量
- 实例变量（对象的变

```
public class ...  
    // 类  
    static String name;  
  
    // 实例变量（对象的变量）  
    int age;  
}
```

特点、用法、应用场景

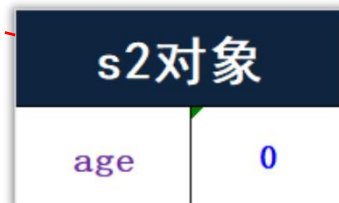
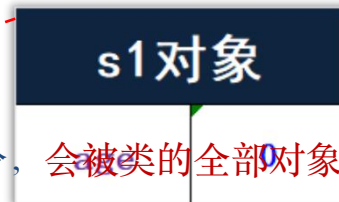
· 有static修饰，属于类，在计算机里只有一份，会被类的全部对象共享。

：无static修饰，属于每个对象的。

类名.类变量(推荐)

对象.类变量(不推荐)

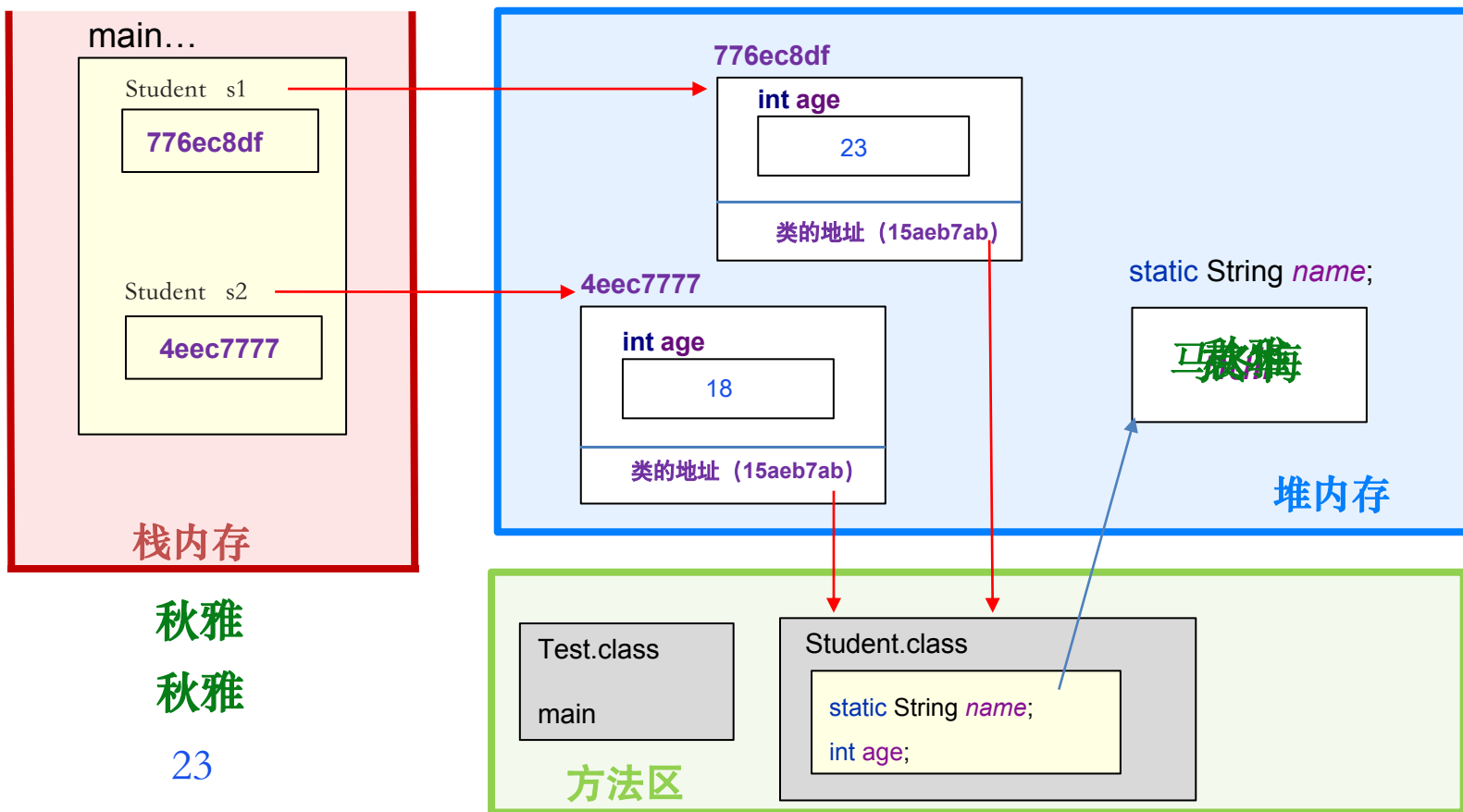
对象.实例变量



成员变量的执行原理

```
public class Test {  
    public static void main(String[] args)  
    // 1、类变量的用法  
    // 类名.类变量(推荐)  
    Student.name = "袁华";  
  
    // 对象.类变量 (不推荐)  
    Student s1 = new Student();  
    s1.name = "马冬梅";  
  
    Student s2 = new Student();  
    s2.name = "秋雅";  
  
    System.out.println(s1.name); // 秋雅  
    System.out.println(Student.name); // 秋雅  
  
    // 2、实例变量的用法  
    // 对象.实例变量  
    s1.age = 23;  
    s2.age = 18;  
    System.out.println(s1.age); // 23  
    // System.out.println(Student.age); // 报错  
}
```

```
public class Student {  
    // 类变量  
    static String name;  
    // 实例变量(对象变量)  
    int age;  
}
```



- **类变量:** 属于类，与类一起加载一次，在内存中只有一份，可以被类和类的所有对象共享。
- **实例变量:** 属于对象，每个对象中都有一份，只能用对象访问。



总结

1、static是什么？

- 叫静态，可以修饰成员变量、成员方法。

2、static修饰的成员变量叫什么？ 如何使用？ 有啥特点？

- **类变量**（静态成员变量）

类名.类变量(推荐)

对象名.类变量(不推荐)

- 属于类，与类一起加载一次，在内存中只有一份，会被类的所有对象共享。

3、无static修饰的成员变量叫什么？ 如何使用？ 有啥特点？

- **实例变量**(对象变量)

对象.实例变量

- 属于对象，每个对象中都有一份。



目录

Contents

➤ static

- ◆ static修饰成员变量

- ◆ static修饰成员变量的应用场景

- ◆ static修饰成员方法

- ◆ static修饰成员方法的应用场景

- ◆ static的注意事项

- ◆ static的应用知识：代码块

- ◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

类变量的应用场景

- 在开发中，如果某个数据只需要一份，且希望能够被共享(访问、修改)，则该数据可以定义成类变量来记住。

案例导学：

- 系统启动后，要求用户类可以记住自己创建了多少个用户对象了。

```
new User();
```

```
new User();
```

```
new User();
```

```
public class User {  
  
    // 类变量  
    public static int number;  
  
    // 构造器  
    public User(){  
        User.number++;  
    }  
}
```

总结

1、成员变量有几种？各自在什么情况下定义？

- 类变量：数据只需要一份，且需要被共享时（访问，修改）
- 实例变量：每个对象都要有一份，数据各不同（如：name、score、age）

2、访问自己类中的类变量，是否可以省略类名不写？

- 可以的
- 注意：在某个类中访问其他类里的类变量，必须带类名访问



目录

Contents

➤ static

- ◆ static修饰成员变量
- ◆ static修饰成员变量的应用场景
- ◆ static修饰成员方法
- ◆ static修饰成员方法的应用场景
- ◆ static的注意事项
- ◆ static的应用知识：代码块
- ◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

成员方法的分类

- **类方法**：有static修饰的成员方法，属于类。

```
public static void printHelloWorld(){  
    System.out.println("Hello World!");  
    System.out.println("Hello World!");  
}
```

类名.类方法 (推荐)

对象名.类方法(不推荐)

- **实例方法**：无static修饰的成员方法，属于对象。

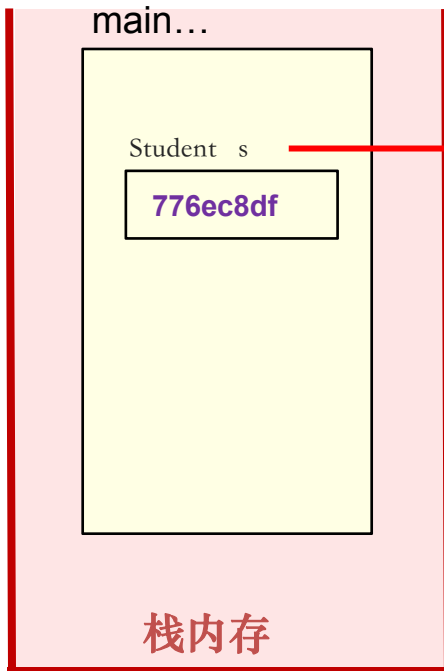
```
public void printPass(){  
    ...  
}
```

对象.实例方法

成员方法的执行原理

```
public class Test {  
    public static void main(String[] args) {  
        // 1、类方法的用法  
        // 类名.类方法 (推荐)  
        Student.printHelloWorld();  
  
        // 对象.类方法 (不推荐)  
        Student s = new Student();  
        s.printHelloWorld();  
  
        // 2、实例方法的用法  
        // 对象.实例方法  
        s.printPass();  
        // Student.printPass(); // 报错  
    }  
}
```

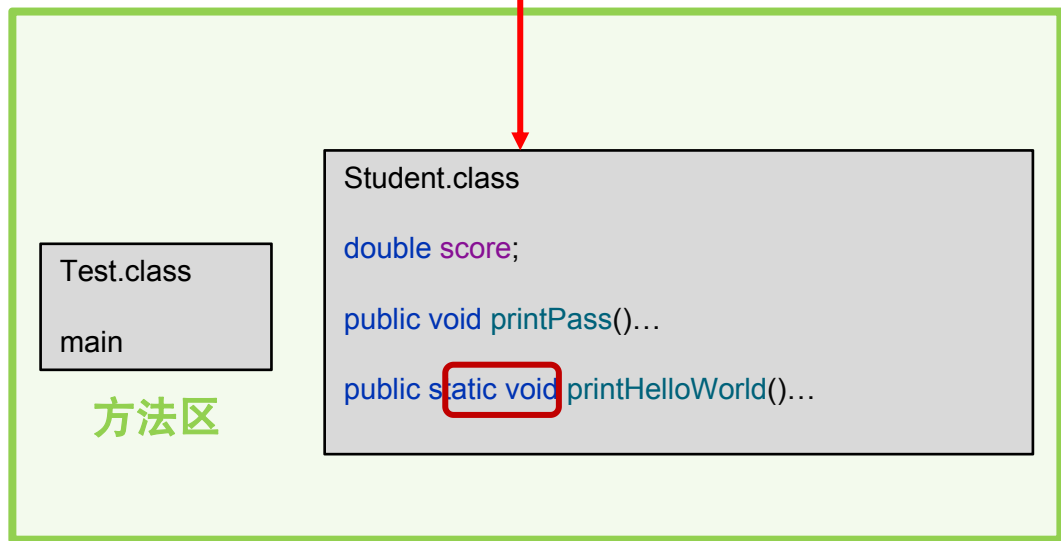
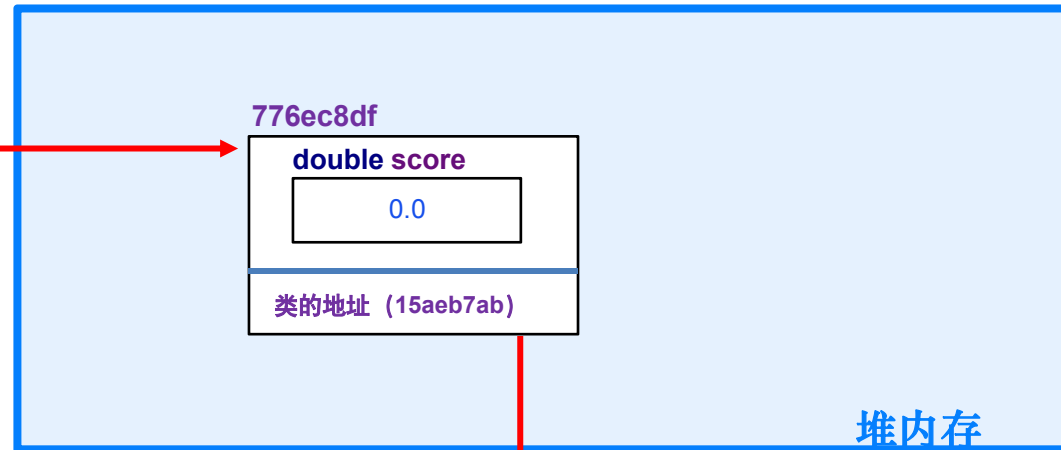
```
public class Student {  
    double score;  
  
    // 类方法: 有static修饰, 属于类  
    public static void printHelloWorld(){  
        System.out.println("Hello World!");  
        System.out.println("Hello World!");  
    }  
  
    // 实例方法 (对象的方法)  
    public void printPass(){  
        System.out.println("成绩: " +  
            (score >= 60 ? "及格" : "不及格"));  
    }  
}
```



Hello World!
Hello World!

Hello World!
Hello World!

成绩: 不及格





总结

1、static修饰的成员方法叫什么？ 如何使用？

- **类方法**（静态方法）
- 属于类，可以直接用类名访问，也可以用对象访问

类名.类方法(推荐)

对象名.类方法(不推荐)

2、无static修饰的成员方法叫什么？ 如何使用？

- **实例方法**(对象的方法)
- 属于对象，只能用对象访问

对象.实例方法

补充知识：搞懂main方法

```
public class Test {  
    public static void main(String[] args) {  
        ...  
    }  
}
```

类方法

1、main方法是啥方法？

2、main方法咋就能直接跑起来？

java Test ----> Test.main(...)



目录

Contents

➤ static

- ◆ static修饰成员变量
- ◆ static修饰成员变量的应用场景
- ◆ static修饰成员方法
- ◆ static修饰成员方法的应用场景
- ◆ static的注意事项
- ◆ static的应用知识：代码块
- ◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

类方法

实例方法

类方法的常见应用场景

- 类方法最常见的应用场景是做工具类。

工具类是什么?

- 工具类中的方法都是一些类方法，每个方法都是用来完成一个功能的，工具类是给开发人员共同使用的。

```
public class XxxUtil {  
    public static void xxx(){  
        ...  
    }  
    public static boolean xxxx(String email){  
        ...  
    }  
    public static String xxxxx(int n){  
        ...  
    }  
    ...  
}
```

区分大小写

密码?



为什么工具类中的方法要用类方法，而不用实例方法？

- 实例方法需要创建对象来调用，此时对象只是为了调用方法，对象占内存，这样会浪费内存。
- 类方法，直接用类名调用即可，调用方便，也能节省内存。

多学一招：

- 工具类没有创建对象的需求，建议将工具类的构造器进行私有。



总结

1. 类方法有啥应用场景?
 - 可以用来设计工具类。
2. 工具类是什么，有什么好处?
 - 工具类中的方法都是类方法，每个类方法都是用来完成一个功能的。
 - 提高了代码的复用性；调用方便，提高了开发效率。
3. 为什么工具类要用类方法，而不是用实例方法?
 - 实例方法需要创建对象来调用，会浪费内存。
4. 工具类定义时有什么要求?
 - 工具类不需要创建对象，建议将工具类的构造器私有化。



目录

Contents

➤ static

- ◆ static修饰成员变量
- ◆ static修饰成员变量的应用场景
- ◆ static修饰成员方法
- ◆ static修饰成员方法的应用场景
- ◆ static的注意事项
- ◆ static的应用知识：代码块
- ◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

使用类方法、实例方法时的几点注意事项

- 类方法中可以直接访问类的成员，不可以直接访问实例成员。
- 实例方法中既可以直接访问类成员，也可以直接访问实例成员。
- 实例方法中可以出现this关键字，类方法中不可以出现this关键字的。



目录

Contents

➤ static

- ◆ static修饰成员变量
- ◆ static修饰成员变量的应用场景
- ◆ static修饰成员方法
- ◆ static修饰成员方法的应用场景
- ◆ static的注意事项
- ◆ static的应用知识：代码块
- ◆ static的应用知识：单例设计模式

➤ 面向对象三大特征之二：继承

代码块概述

- 代码块是类的5大成分之一（成员变量、构造器、方法、**代码块**、内部类）。

代码块分为两种：

- **静态代码块：**

- **格式：** static { }
- **特点：** 类加载时自动执行，由于类只会加载一次，所以静态代码块也只会执行一次。
- **作用：** 完成类的初始化，例如：对类变量的初始化赋值。

- **实例代码块：**

- **格式：** { }
- **特点：** 每次创建对象时，执行实例代码块，并在构造器前执行。
- **作用：** 和构造器一样，都是用来完成对象的初始化的，例如：对实例变量进行初始化赋值。



目录

Contents

➤ static

◆ static修饰成员变量

◆ static修饰成员变量的应用场景

◆ static修饰成员方法

◆ static修饰成员方法的应用场景

◆ static的注意事项

◆ static的应用知识：代码块

架构师 框架

◆ static的应用知识：单例设计模式

面试笔试

➤ 面向对象三大特征之二：继承

看源码

什么是设计模式 (Design pattern) ?

- 一个问题通常有n种解法，其中肯定有一种解法是最优的，这个最优的解法被人总结出来了，称之为**设计模式**。
- 设计模式有20多种，对应20多种软件开发中会遇到的问题。

关于设计模式的学习，主要学什么?

01

解决什么问题?

02

怎么写?

单例设计模式

单例设计模式

- 确保一个类只有一个对象。

写法

- 把类的构造器私有。
- 定义一个类变量记住类的一个对象。
- 定义一个类方法，返回对象。

```
// 单例类
public class A {
    // 2、定义一个类变量记住类的一个对象
    private static A a = new A();

    // 1、私有构造器
    private A(){
    }

    // 3、定义一个类方法返回对象
    public static A getObject(){
        return a;
    }
}
```

单例模式的应用场景和好处?

Runtime

任务管理器					
文件(F) 选项(O) 查看(V)					
进程 性能 应用历史记录 启动 用户 详细信息 服务					
名称	状态	3% CPU	59% 内存	0% 磁盘	0% 网络
> Camtasia 9		0%	606.2 MB	0 MB/秒	0 Mbps
> IntelliJ IDEA (5)		0%	238.5 MB	0 MB/秒	0 Mbps
> WXWork.exe (32 位) (8)		0.1%	162.6 MB	0 MB/秒	0 Mbps
> Microsoft PowerPoint (2)		0.1%	118.2 MB	0 MB/秒	0 Mbps
^ 简略信息(D) 结束任务(E)					

单例设计模式的实现方式很多

- **饿汉式单例**：拿对象时，对象早就创建好了。

- 懒汉式单例：拿对象时，才开始创建对象。

- ...

- ...

- ...

```
public class A {  
    // 2、定义一个类变量记住类的一个对象  
    private static A a = new A();  
    // 1、私有构造器  
    private A(){  
    }  
    // 3、定义一个类方法返回对象  
    public static A getObject(){  
        return a;  
    }  
}
```



总结

1. 什么是设计模式，设计模式主要学什么？单例模式解决了什么问题？
 - 设计模式就是具体问题的最优解决方案。
 - 解决了什么问题？ 怎么写？
 - 确保一个类只有一个对象。
2. 单例怎么写？ 饿汉式单例的特点是什么？
 - 把类的构造器私有；定义一个类变量存储类的一个对象；提供一个类方法返回对象。
 - 在获取类的对象时，对象已经创建好了。
3. 单例有啥应用场景，有啥好处？
 - 任务管理器对象、获取运行时对象。
 - 在这些业务场景下，使用单例模式，可以避免浪费内存。

懒汉式单例设计模式

- 拿对象时，才开始创建对象。

写法

- 把类的构造器私有。
- 定义一个类变量用于存储对象。
- 提供一个类方法，保证返回的是同一个对象。

```
public class B {  
    // 2、定义一个类变量用于存储对象  
    public static B b ; // null  
  
    // 1、单例必须私有构造器  
    private B(){  
    }  
  
    // 3、提供一个类方法返回类的一个对象  
    public static B getObject(){  
        if(b == null){  
            b = new B();  
        }  
        return b;  
    }  
}
```



总结

1. 懒汉单例模式的特点是什么?
 - 要用类的对象时才创建对象（延迟加载对象）
2. 懒汉单例模式怎么写?
 - 把构造器私有。
 - 定义一个类变量用于存储对象。
 - 提供一个类方法，保证返回的是同一个对象。



目录

Contents

➤ static

➤ 面向对象的三大特征之二：继承

◆ 继承的快速入门

封装 继承 多态

◆ 继承相关的注意事项

① 单继承、Object类

② 方法重写

③ 子类中访问其他成员的特点

④ 子类构造器的特点

⑤ 权限修饰符

⑥ 注意事项的小结

继承的学习目标



- Java中提供了一个关键字extends，用这个关键字，可以让一个类和另一个类建立起父子关系。

```
public class B extends A{  
  
}
```

什么是继承呢？

B类称为子类（派生类）。

继承的特点

- 子类能继承父类的非私有成员（成员变量、成员方法）。

继承后对象的创建

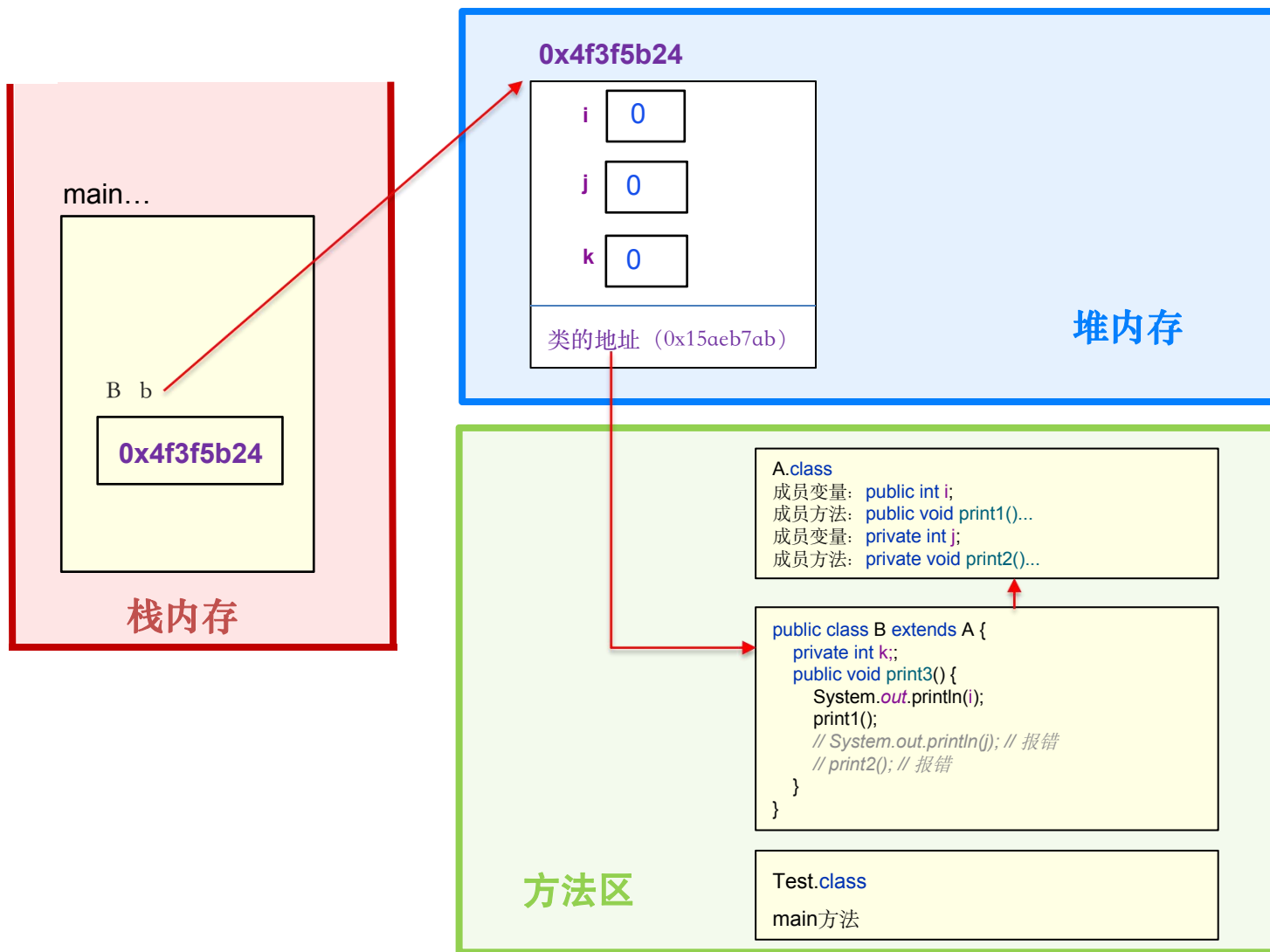
- 子类的对象是由子类、父类共同完成的。

继承的执行原理

```
public class A {  
    public int i;  
    public void print1(){  
        System.out.println("===print1===");  
    }  
  
    private int j;  
    private void print2(){  
        System.out.println("===print2===");  
    }  
}
```

```
public class B extends A {  
    private int k;  
    public void print3() {  
        // 子类可以继续父类的非私有成员  
        System.out.println(i);  
        print1();  
        // System.out.println(j); // 报错  
        // print2(); // 报错  
    }  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        // 目标: 认识继承、掌握继承的特点。  
        B b = new B();  
        System.out.println(b.i);  
        // System.out.println(b.j); // 报错  
        // System.out.println(b.k); // 报错  
        b.print1();  
        // b.print2(); // 报错  
        b.print3();  
    }  
}
```



- 子类对象实际上是由子父类这两张设计图共同创建出来的。



总结

1、什么是继承？继承后有啥特点？

- 继承就是用extends关键字，让一个类和另一个类建立起一种父子关系。
- 子类可以继承父类非私有的成员。

2、带继承关系的类，Java会怎么创建它的对象？对象创建出来后，可以直接访问哪些成员？

- 带继承关系的类，java会用类和其父类，这多张设计图来一起创建类的对象。
- 对象能直接访问什么成员，是由子父类这多张设计图共同决定的，这多张设计图对外暴露了什么成员，对象就可以访问什么成员。

继承的学习目标



使用继承有啥好处?

- 减少重复代码的编写。

需求

黑马的员工管理系统中

需要处理讲师、咨询师的数据

讲师的数据有：姓名、具备的技能；

咨询的数据有：姓名、解答问题的总人数。

```
public class Teacher {  
    private String name;  
    private String skill; // 技能  
  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getSkill() {  
        return skill;  
    }  
    public void setSkill(String skill) {  
        this.skill = skill;  
    }  
}
```

重复的
代码

```
public class Consultant {  
    private String name;  
    private int number; // 解答问题的人数  
  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public int getNumber() {  
        return number;  
    }  
    public void setNumber(int number) {  
        this.number = number;  
    }  
}
```

使用继承有啥好处?

- 减少重复代码的编写。

需求

黑马的员工管理系统中

需要处理讲师、咨询师的数据

讲师的数据有：姓名、具备的技能；

咨询的数据有：姓名、解答问题的总人数。

爸爸!

```
public class People {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

爸爸!

```
public class Teacher extends People{  
    private String skill; // 技能  
  
    public String getSkill() {  
        return skill;  
    }  
    public void setSkill(String skill) {  
        this.skill = skill;  
    }  
}
```

```
public class Consultant extends People{  
    private int number; // 解答问题的总人数  
  
    public int getNumber() {  
        return number;  
    }  
    public void setNumber(int number) {  
        this.number = number;  
    }  
}
```



总结

1、使用继承有啥好处?

- 减少了重复代码的编写，提高了代码的复用性。



目录

Contents

➤ **static**

➤ **面向对象三大特征之二：继承**

◆ 继承的快速入门

◆ 继承相关的注意事项

① 权限修饰符

public

private

protected

缺省

② 单继承、Object类

③ 方法重写

④ 子类中访问其他成员的特点

⑤ 子类构造器的特点

⑥ 注意事项的小结

什么是权限修饰符?

- 就是是用来限制类中的成员（成员变量、成员方法、构造器、代码块…）能够被访问的范围。

权限修饰符有几种？各自的作用是什么？

private

只能本类

缺省

本类、同一个包中的类

protected

本类，同一个包中的类、子孙类中

public

任意位置

private < 缺省 < protected < public

什么是权限修饰符?

- 就是是用来限制类中的成员（成员变量、成员方法、构造器、代码块…）能够被访问的范围。

权限修饰符有几种？各自的作用是什么？

修饰符	本类里	同一个包中的类	子孙类	任意类
private	✓			
缺省	✓	✓		
protected	✓	✓	✓	
public	✓	✓	✓	✓

private < 缺省 < protected < public



目录

Contents

- **static**
- **面向对象三大特征之二：继承**
 - ◆ 继承的快速入门
 - ◆ 继承相关的注意事项
 - ① 权限修饰符
 - ② 单继承、Object
 - ③ 方法重写
 - ④ 子类中访问其他成员的特点
 - ⑤ 子类构造器的特点
 - ⑥ 注意事项的小结

Java是**单继承的**，Java中的类**不支持多继承**，但是**支持多层继承**。

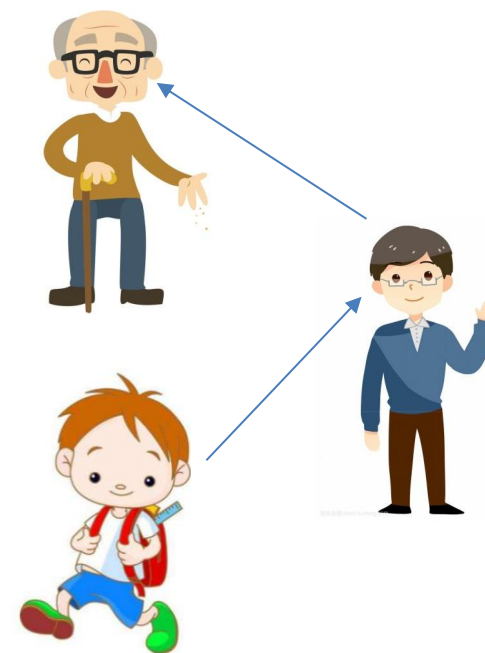
单继承



Java中的类**不支持多继承**



支持多层继承



为何Java中的类不支持**多继承**

```
class 子类 extends 父类A, 父类B {  
    }  
    
```



请看如下反证法:




```
public class A {  
    public void method(){  
        System.out.println("复习数学~");  
    }  
}
```



```
public class B {  
    public void method(){  
        System.out.println("复习语文~");  
    }  
}
```



```
public class C extends A, B {  
    }  
    
```



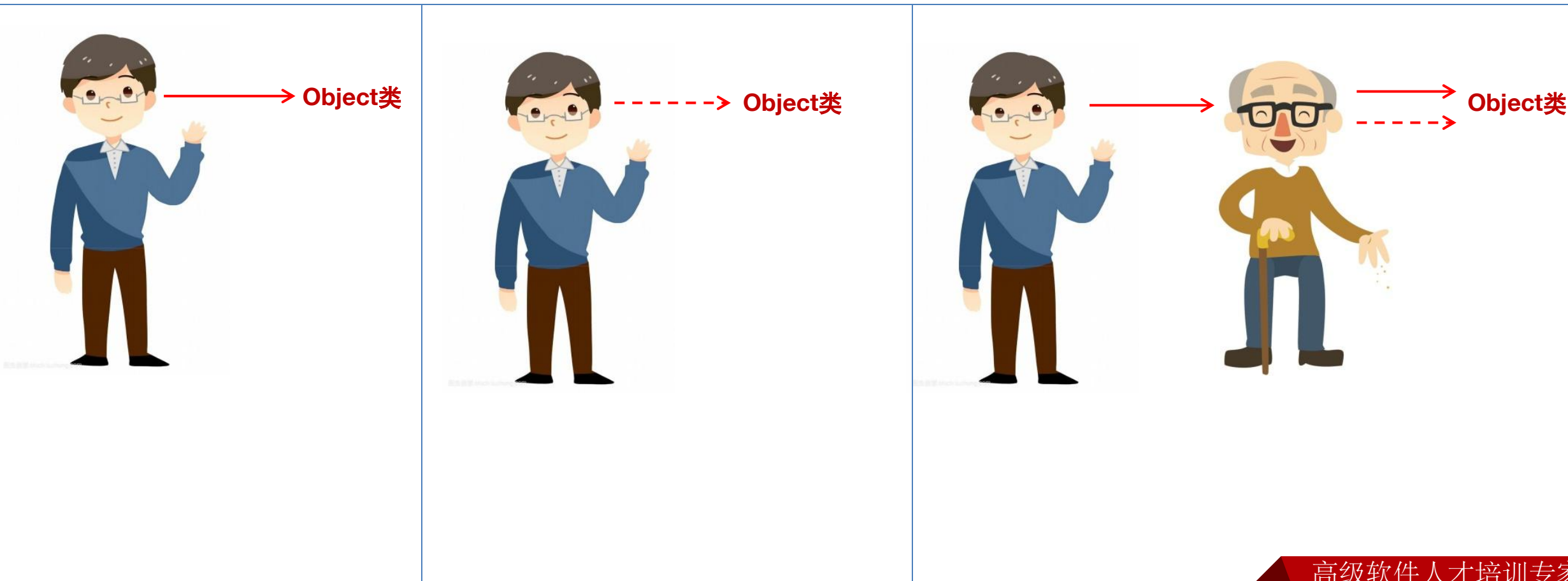
```
public class Test{  
    public static void main(String[] args) {  
        C c = new C ();  
        c.method();  
    }  
}
```

// 懵了，不知道听哪个爸爸的

结论：Java中的类不支持**多继承**

Object类

- object类是java所有类的祖宗类。我们写的任何一个类，其实都是object的子类或子孙类。





总结

1、继承相关的两个注意事项?

- ① Java是单继承的：一个类只能继承一个直接父类；Java中的类不支持多继承，但是支持多层继承。
- ② Object类是Java中所有类的祖宗。



目录

Contents

- **static**
- **面向对象三大特征之二：继承**
 - ◆ 继承的快速入门
 - ◆ 继承相关的注意事项
 - ① 权限修饰符
 - ② 单继承、Object类
 - ③ **方法重写**
 - ④ 子类中访问其他成员的特点
 - ⑤ 子类构造器的特点
 - ⑥ 注意事项的小结

方法重写的学习路径



什么是方法重写?

- 当子类觉得父类中的某个方法不好用，或者无法满足自己的需求时，子类可以重写一个方法名称、参数列表一样的方法，去覆盖父类的这个方法，这就是方法重写。
- 注意：重写后，方法的访问，Java会遵循就近原则。

方法重写的其它注意事项

- 重写小技巧：使用**Override注解**，他可以指定java编译器，检查我们方法重写的格式是否正确，代码可读性也会更好。
- 子类重写父类方法时，访问权限必须大于或者等于父类该方法的权限（**public > protected > 缺省**）。
- 重写的方法返回值类型，必须与被重写方法的返回值类型一样，或者范围更小。
- 私有方法、静态方法不能被重写，如果重写会报错的。

- 子类重写Object类的toString()方法，以便返回对象的内容。

方法重写在开发中的常见应用场景



总结

1. 方法重写是什么?

- 子类写了一个**方法名称，形参列表与父类某个方法一样**的方法去覆盖父类的该方法。

2. 重写方法有哪些注意事项?

- 建议加上：`@Override`注解，可以校验重写是否正确，同时可读性好。
- 子类重写父类方法时，访问权限必须大于或者等于父类被重写的方法的权限。
- 重写的方法返回值类型，必须与被重写方法的返回值类型一样，或者范围更小。
- 私有方法、静态方法不能被重写。

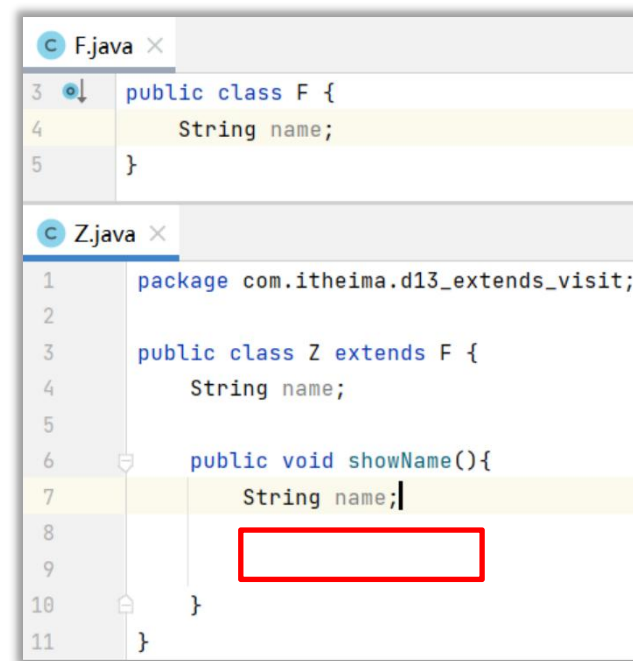
3. 方法重写有啥应用场景?

- 当子类觉得父类的方法不好用，或者不满足自己的需求时，就可以用方法重写。

目录

Contents

- static
- 面向对象三大特征之二：继承
 - ◆ 继承的快速入门
 - ◆ 继承相关的注意事项
 - ① 权限修饰符
 - ② 单继承、Object类
 - ③ 方法重写
 - ④ 子类中访问其他成员的特点
 - ⑤ 子类构造器的特点
 - ⑥ 注意事项的小结



```
F.java
3 public class F {
4     String name;
5 }

Z.java
1 package com.itheima.d13_extends_visit;
2
3 public class Z extends F {
4     String name;
5
6     public void showName(){
7         String name;
8     }
9
10 }
11 }
```

1、在子类方法中访问其他成员（成员变量、成员方法），是依照**就近原则**的。

- 先子类局部范围找。
- 然后子类成员范围找。
- 然后父类成员范围找，如果父类范围还没有找到则报错。

2、如果子父类中，出现了重名的成员，会优先使用子类的，如果此时一定要在子类中使用父类的怎么办？

- 可以通过**super关键字**，指定访问父类的成员：**super.父类成员变量/父类成员方法**



总结

- 1、在子类方法中访问成员（成员变量、成员方法）是什么特点?
 - 就近原则，子类没有找子类、子类没有找父类、父类没有就报错!
- 2、如果子父类中出现了重名的成员，如果此时一定要在子类中使用父类的怎么办?

super.父类成员变量/父类成员方法



目录

Contents

➤ **static**

➤ **面向对象三大特征之二：继承**

◆ 继承的快速入门

◆ 继承相关的注意事项

① 权限修饰符

② 单继承、Object类

③ 方法重写

④ 子类中访问其他成员的特点

⑤ 子类构造器的特点

⑥ 注意事项的小结

子类构造器的特点



子类构造器的特点:

- 子类的全部构造器，都会先调用父类的构造器，再执行自己。

子类构造器是如何实现调用父类构造器的:

- 默认情况下，子类全部构造器的第一行代码都是 `super()`（写不写都有），它会调用父类的无参数构造器。
- 如果父类没有无参数构造器，则我们必须在子类构造器的第一行手写`super(...)`，指定去调用父类的有参数构造器。

子类构造器的特点

子类的全部构造器都会先调用父类的构造器

01

认识子类构造器的特点



02

为什么这么干？

常见的应用场景

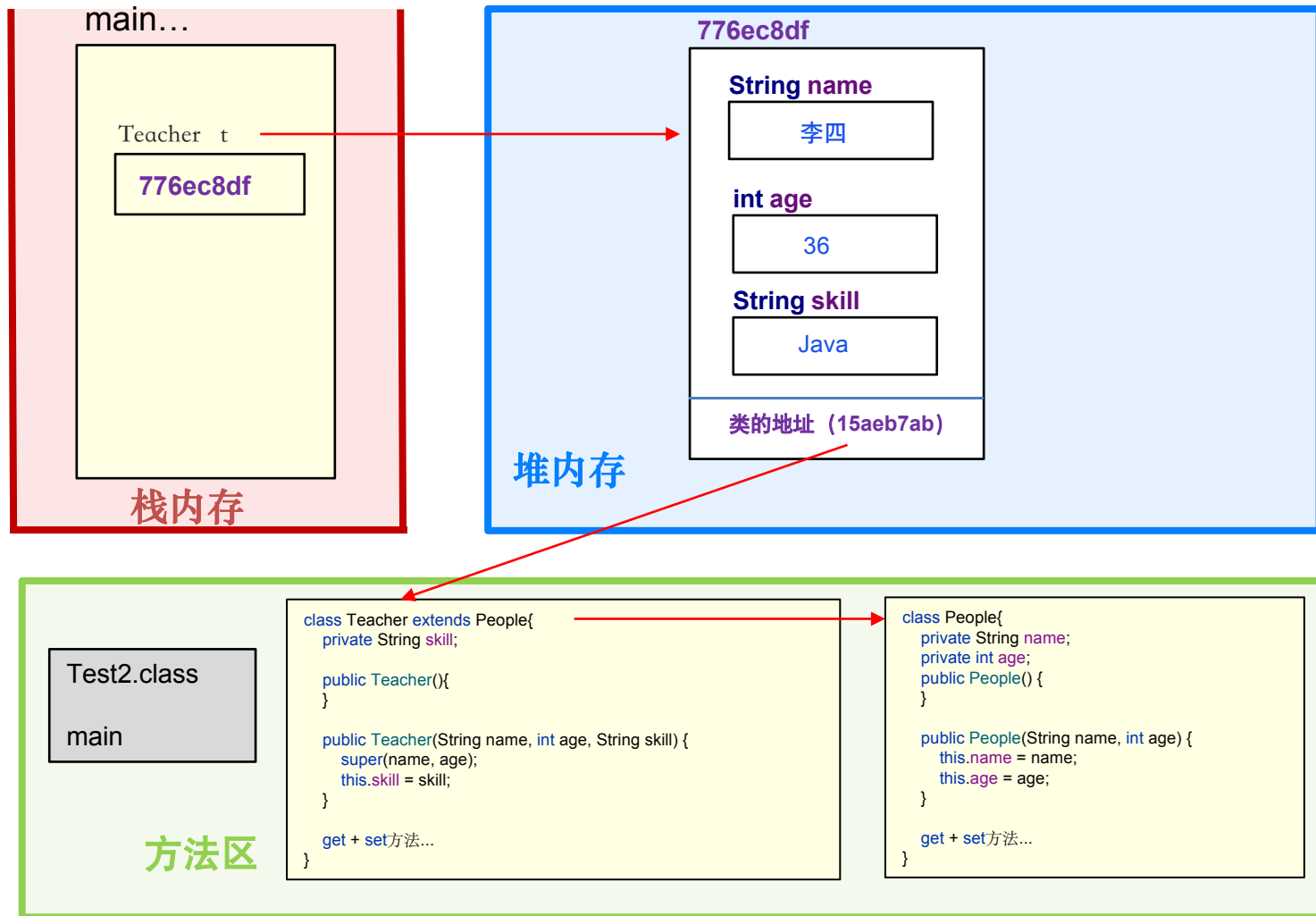
小结

子类构造器调用父类构造器的常见应用场景

```
class People{  
    private String name;  
    private int age;  
    public People() {  
    }  
  
    public People(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    get + set方法...  
}
```

```
class Teacher extends People{  
    private String skill;  
    public Teacher(){  
    }  
  
    public Teacher(String name, int age, String skill) {  
        super(name, age);  
        this.skill = skill;  
    }  
  
    get + set方法...  
}
```

```
public class Test2 {  
    public static void main(String[] args) {  
        // 目标：子类调用父类构造器的常见应用场景。  
        Teacher t = new Teacher("李四", 36, "Java");  
        System.out.println(t.getName());  
        System.out.println(t.getAge());  
        System.out.println(t.getSkill());  
    }  
}
```




- 子类构造器可以通过调用父类构造器，把对象中包含父类这部分的数据先初始化赋值，再回来把对象里包含子类这部分的数据也进行初始化赋值。

补充知识：this(···)调用兄弟构造器

- 任意类的构造器中，是可以通过this(···)去调用该类的其他构造器的。

```
public class Student {  
    private String schoolName;  
    private String name;  
  
    public Student(String name){  
        this(name, "黑马程序员");  
    }  
  
    public Student(String name, String schoolName){  
        this.name = name;  
        this.schoolName = schoolName;  
    }  
}
```



this(...)和super(...)使用时的注意事项:

- this(···)、super(···) 都只能放在构造器的第一行，因此，有了this(···)就不能写super(···)了，反之亦然。



总结

1. 子类构造器有啥特点?
 - 子类中的全部构造器，都必须先调用父类的构造器，再执行自己。
2. `super(...)`调用父类有参数构造器的常见应用场景是什么?
 - 为对象中包含父类这部分成员变量进行赋值。
3. `this(...)`的作用是什么?
 - 在构造器中调用本类的其他构造器。
3. `this(...)`和`super(...)`的使用需要注意什么?
 - 都必须放在构造器的第一行。

千里之行、始于足下

好记性不如烂笔头

写代码、写代码、写代码



目录

Contents

- **static**
- **面向对象三大特征之二：继承**
 - ◆ 继承的快速入门
 - ◆ 继承相关的注意事项
 - ① 权限修饰符
 - ② 单继承、Object类
 - ③ 方法重写
 - ④ 子类中访问其他成员的特点
 - ⑤ 子类构造器的特点
 - ⑥ 注意事项的小结

this和super详情

- **this**：代表本类对象的引用；**super**：代表父类存储空间的标识。

关键字	访问成员变量	访问成员方法	访问构造方法
this	this.成员变量 访问本类成员变量	this.成员方法(...) 访问本类成员方法	this(...) 访问本类构造器
super	super.成员变量 访问父类成员变量	super.成员方法(...) 访问父类成员方法	super(...) 访问父类构造器



实际上，在以上的总结中，**唯独只有this调用本类其他构造器我们是没有接触过的。**

为什么子类构造器一定要先调用父类的构造器:

- 引入继承后，子类对象是由子类和父类这两张设计图共同创建出来的，当创建子类对象，调用子类构造器为子类的成员变量初始化时，子类构造器肯定需要先调用父类构造器，把父类中的成员变量先进行初始化。
- 注意: `super(...)`必须放在构造器的第一行，否则报错

子类构造器调用父类构造器有啥应用场景?

- 为父类这部分成员变量进行初始化赋值。





传智教育旗下高端IT教育品牌