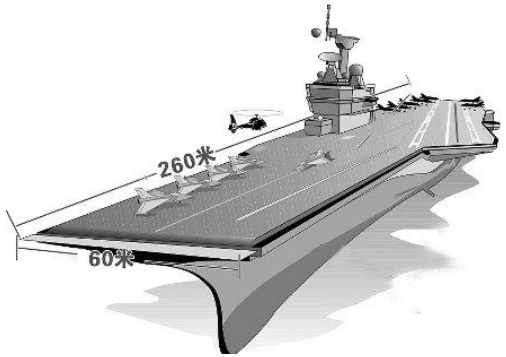
面向对象进阶

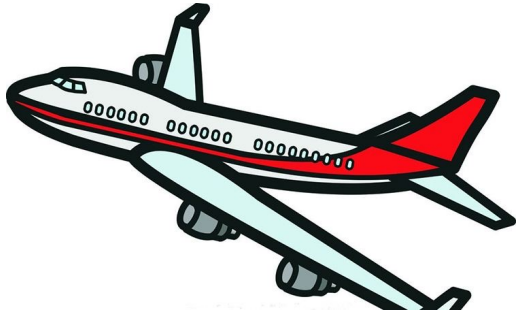
写程序的套路

讲师：徐磊

黑马程序员
www.itheima.com
传智教育旗下
高端IT教育品牌

面向对象知识的学习建议





多关注语法点的基本作用 → 多进行思考和记忆、练习 → 要自信，不要在短期想能做什么？

面向对象进阶课程：第一天同学们需要学习哪些语法？

static关键字

之前定义的成员变量：name，age属于每个对象的。如何表示共享的信息？如在线人数等

设计模式：单例

有些类只需要一个对象就可以了，如任务管理器对象，如何实现一个类只能对外产生一个对象？

面向对象三大特征：继承

系统中很多实体类的属性和行为存在重复代码，如何把这些类信息进行优化，降低代码冗余，提升代码复用呢？

目录

Contents

- **static静态关键字**
  - static是什么，static修饰成员变量的用法
  - static修饰成员变量的应用：在线人数统计
  - static修饰成员方法的用法
  - static修饰成员方法的应用：工具类
  - static的注意事项
- **static应用知识：代码块**
- **static应用知识：单例**
- **面向对象三大特征之二：继承**

static是什么

- static是静态的意思，可以用来修饰成员变量、成员方法。
- static修饰成员变量之后称为静态成员变量（类变量），修饰方法之后称为静态方法（类方法）。
- static修饰后的成员变量，可以被类的所有对象共享（访问、修改)。

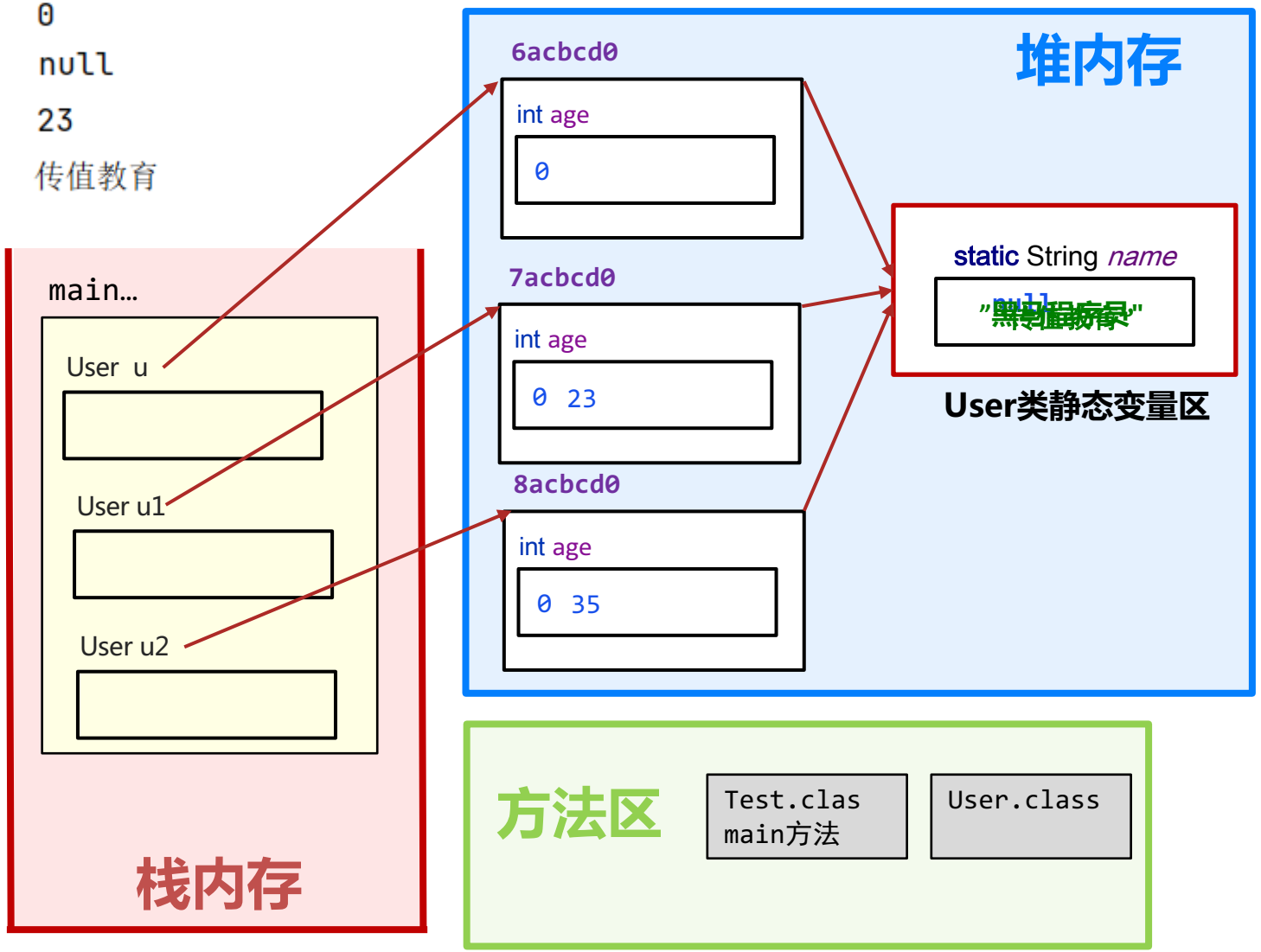
```
public class User {
    static String name;
    int age;
}
```

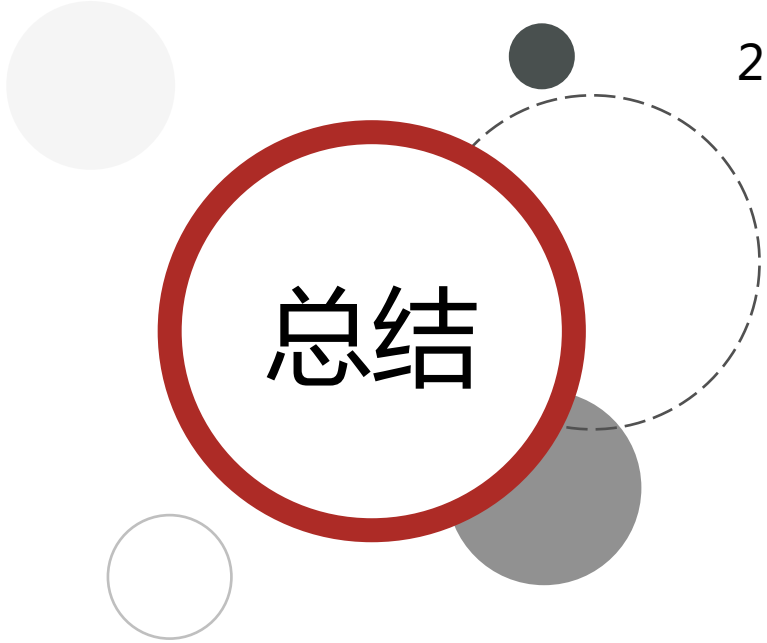
成员变量内存原理

```
public class User {
    /** 静态成员变量 */
    static String name;
    /** 实例成员变量 */
    int age;
}
```

```
public class Test {
    public static void main(String[] args) {
        // 目标：认识static，掌握static修饰成员变量的特点和使用场景。
        // 1、访问的区别
        System.out.println(User.name);
        // System.out.println(User.age); // 报错

        User u = new User();
        System.out.println(u.age);
        System.out.println(u.name);
        // 静态成员变量可以用类，也可以用对象访问
        // 实例成员变量只能用对象访问。

        // 2、static修饰静态成员变量的特点。
        User u1 = new User();
        u1.age = 23;
        u1.name = "黑马程序员";

        User u2 = new User();
        u2.age = 35;
        u2.name = "传值教育";

        System.out.println(u1.age); // 23
        System.out.println(u1.name); // 传值教育

    }
}
```

```
null
0
null
23
传值教育
```





1. static是什么 ？
   - **静态的意思，可以修饰成员变量、成员方法**
2. static修饰的成员变量是什么？ 有什么特点？
   - **静态成员变量（有static修饰，属于类、加载一次，内存中只有一份），访问格式**

     类名.静态成员变量(推荐)

     对象.静态成员变量(不推荐)。
   - **实例成员变量（无static修饰，属于对象），访问格式：**

     对象.实例成员变量。
3. 两种成员变量各自在什么情况下定义？
   - **静态成员变量：表示在线人数等需要被类的所有对象共享的信息时。**
   - **实例成员变量：属于每个对象，且每个对象的该信息不同时（如：name,age,money...）**

目录

Contents

- **static静态关键字**
  - static是什么，static修饰成员变量的用法
  - static修饰成员变量的应用：在线人数统计
  - static修饰成员方法的用法
  - static修饰成员方法的应用：工具类
  - static的注意事项
- **static应用知识：代码块**
- **static应用知识：单例**
- **面向对象三大特征之二：继承**

目录

Contents

- **static静态关键字**
  - static的作用、修饰成员变量的用法
  - static修饰成员变量的内存原理
  - static修饰成员方法的基本用法
  - static修饰成员方法的内存原理
  - static的注意事项
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例**
- **面向对象三大特征之二：继承**

之前我们定义的方法有的有static修饰，有的是没有的，有什么不同？

```
public void run(){
    System.out.println(name +"正在好好学习，天天向上~~");
}
```

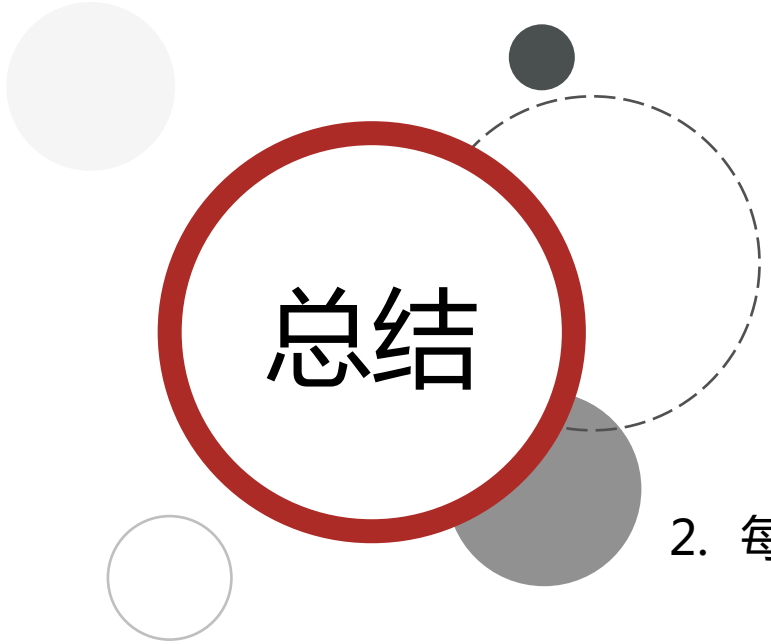
```
public static int getMax(int a , int b){
      return a > b ? a : b;
}
```

成员方法的分类：

- 静态成员方法（有static修饰，归属于类），建议用类名访问，也可以用对象访问。
- 实例成员方法（无static修饰，归属于对象），只能用对象触发访问。

使用场景

- 表示对象自己的行为的，且方法中需要访问实例成员的，则该方法必须申明成实例方法。
- 如果该方法是以执行一个共用功能为目的，则可以申明成静态方法。



1. 成员方法的分类和访问分别是什么样的？

- **静态成员方法（有static修饰，属于类和对象共享）访问格式：**
  - **类名.静态成员方法。**
  - **对象.静态成员方法。（不推荐）**
- **实例成员方法（无static修饰，属于对象）的访问格式：**
  - **对象.实例成员方法。**

2. 每种成员方法的使用场景是怎么样的？

- **表示对象自己的行为的，且方法中需要直接访问实例成员，则该方法必须申明成实例方法。**
- **如果该方法是以执行一个通用功能为目的，或者需要方便访问，则可以申明成静态方法**

练习 定义员工类的实例

8 分钟

需求：请完成一个标准实体类的设计，并提供如下要求实现。

①：某公司的员工信息系统中，需要定义一个公司的员工类Employee，包含如下信息（name, age，所在部门名称dept ），定义一个静态的成员变量company记录公司的名称。

②：需要在Employee类中定义一个方法showInfos()，用于输出当前员工对象的信息。如name, age ，dept 以及公司名称company的信息。

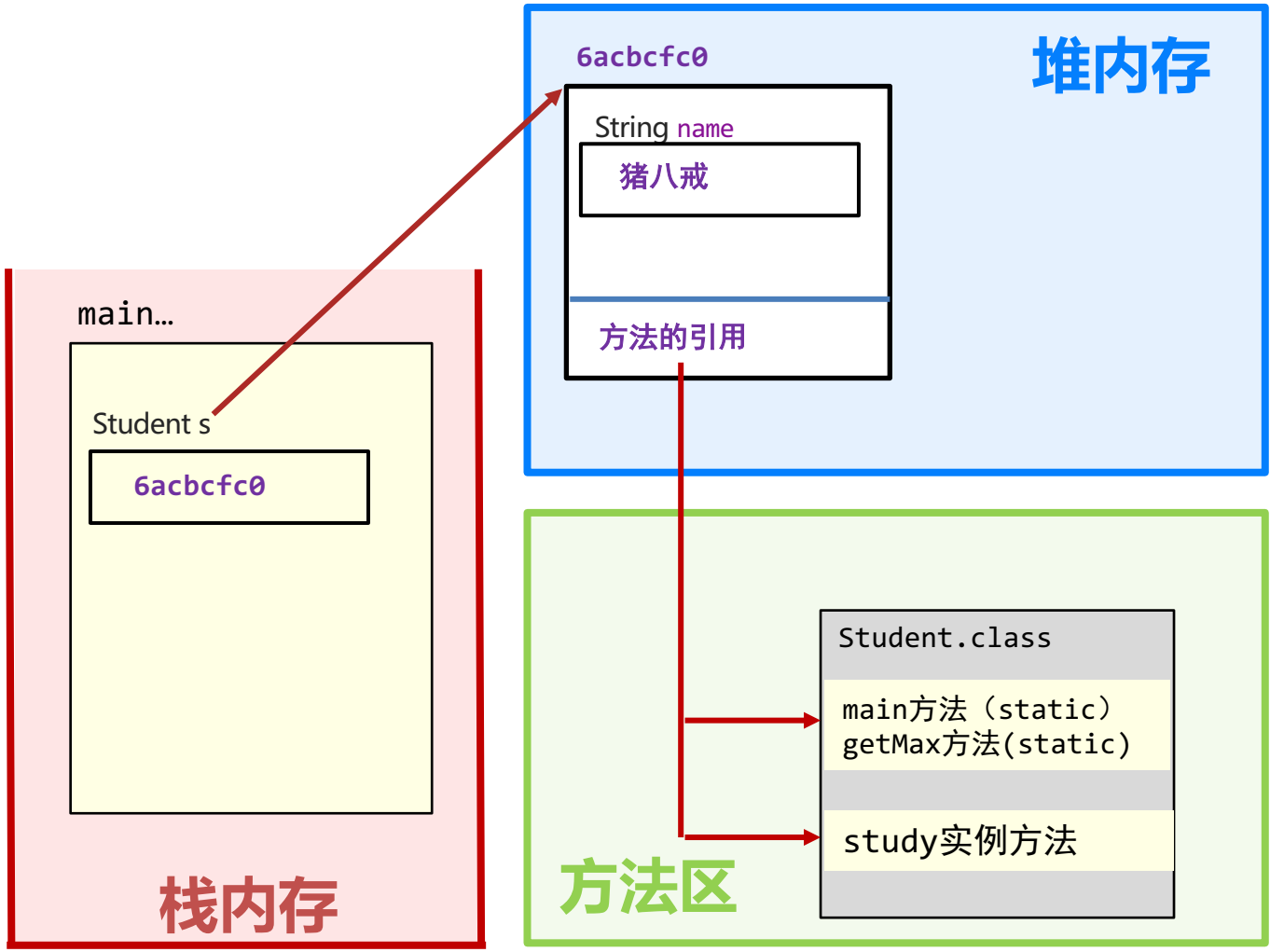
③：需要在Employee类中定义定义一个通用的静态方法compareByAge，用于传输两个员工对象的年龄进入，并返回比较较大的年龄，例如：2个人中的最大年龄是:45岁。

目录

Contents

- **static静态关键字**
  - static的作用、修饰成员变量的用法
  - static修饰成员变量的内存原理
  - static修饰成员方法的基本用法
  - static修饰成员方法的内存原理
  - static的注意事项
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例**
- **面向对象三大特征之二：继承**

```java
public class Student {
    private String name;
    // 1. 实例方法: 无static修饰的，属于对象的
    public void study(){
        System.out.println(name + "在好好学习~~~");
    }
    // 2. 静态方法：有static修饰，属于类和对象共享的
    public static int getMax(int a , int b){
        return a > b ? a : b;
    }
    public static void main(String[] args) {
        // 1. 类名.静态成员方法
        System.out.println(Student.getMax(10 , 2));
        // 注意：同一个类中访问静态成员类名可以不写
        System.out.println(getMax(2 , 10));

        // 2. 对象.实例成员方法
        // study(); // 会报错
        Student s = new  Student();
        s.name = "猪八戒";
        s.study();

        // 3. 对象.静态成员方法。（不推荐）
        System.out.println(s.getMax(20 , 10));
    }
}
```



目录

Contents

- **static静态关键字**
  - static的作用、修饰成员变量的用法
  - static修饰成员变量的内存原理
  - static修饰成员方法的基本用法
  - static修饰成员方法的内存原理
  - static的注意事项[拓展]
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例**
- **面向对象三大特征之二：继承**

static访问注意事项：

- 静态方法只能访问静态的成员，不可以直接访问实例成员。
- 实例方法可以访问静态的成员，也可以访问实例成员。
- 静态方法中是不可以出现this关键字的。

目录

Contents

- static静态关键字
- static应用知识：工具类
- static应用知识：代码块
- static应用知识：单例
- 面向对象三大特征之二：继承

工具类是什么？

- 类中都是一些静态方法，每个方法都是以完成一个共用的功能为目的，这个类用来给系统开发人员共同使用的。

案例导学：

- 在企业的管理系统中，通常需要在一个系统的很多业务处使用验证码进行防刷新等安全控制。

问题：

- 同一个功能多处开发，会出现代码重复度过高。

使用工具类的好处

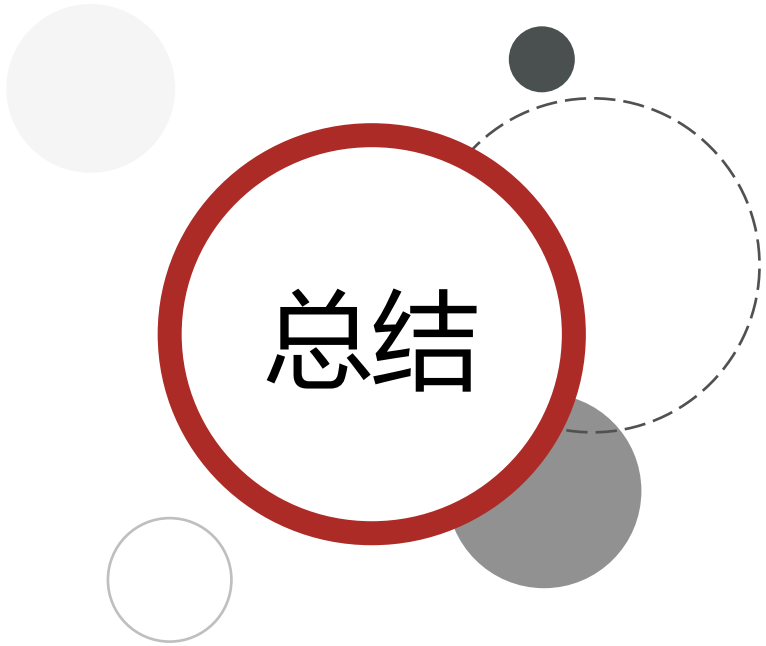
- 一是调用方便，二是提高了代码复用（一次编写，处处可用）

为什么工具类中的方法不用实例方法做？

- **实例方法需要创建对象调用。**
- **此时用对象只是为了调用方法，这样只会浪费内存。**

工具类定义时的其他要求：

- 由于工具里面都是静态方法，直接用类名即可访问，因此，工具类无需创建对象，建议将工具类的构造器进行私有。



1. 工具类是什么，有什么好处？
   - **内部都是一些静态方法，每个方法完成一个功能**
   - **一次编写，处处可用，提高代码的重用性。**
2. 工具类有什么要求？
   - **建议工具类的构造器私有化处理。**
   - **工具类不需要创建对象。**

练习 定义数组工具类

10 分钟

需求：在实际开发中，经常会遇到一些数组使用的工具类。请按照如下要求编写一个数组的工具类：ArraysUtils

① ：我们知道数组对象直接输出的时候是输出对象的地址的，而项目中很多地方都需要返回数组的内容，请在ArraysUtils中提供一个工具类方法toString，用于返回整数数组的内容，返回的字符串格式如：[10, 20, 50, 34, 100]（**只考虑整数数组，且只考虑一维数组**）

② ：经常需要统计平均值，平均值为去掉最低分和最高分后的分值，请提供这样一个工具方法getAerage，用于返回平均分。（**只考虑浮点型数组，且只考虑一维数组**）

③ ：定义一个测试类TestDemo，调用该工具类的工具方法，并返回结果。

目录

Contents

- static静态关键字
- static应用知识：工具类
- static应用知识：代码块
  - 代码块的分类、作用
  - 静态代码块的应用案例
- static应用知识：单例
- 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），定义在类中方法外。
- 在Java类下，使用 { } 括起来的代码被称为代码块 。

代码块分为

- **静态代码块**:
  - **格式**：static{}
  - **特点**：需要通过static关键字修饰，随着类的加载而加载，并且自动触发、只执行一次
  - **使用场景**：在类加载的时候做一些静态数据初始化的操作，以便后续使用。

- **构造代码块（了解，见的少）**：
  - **格式**：{}
  - **特点**：每次创建对象，调用构造器执行时，都会执行该代码块中的代码，并且在构造器执行前执行
  - **使用场景**：初始化实例资源。

目录

Contents

- static静态关键字
- static应用知识：工具类
- static应用知识：代码块
  - 代码块的分类、作用
  - 静态代码块的应用案例
- static应用知识：单例
- 面向对象三大特征之二：继承

案例 斗地主游戏

需求：

在启动游戏房间的时候，应该提前准备好54张牌，后续才可以直接使用这些牌数据。

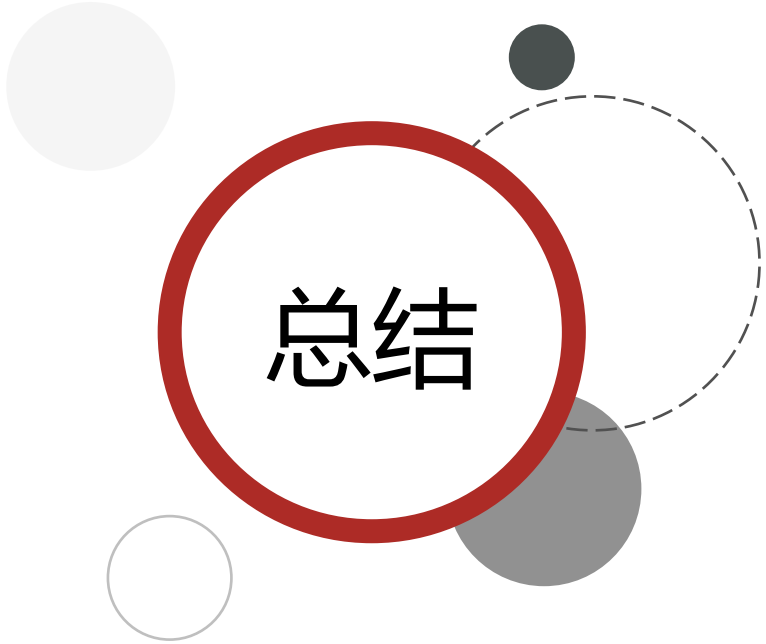
分析：

① 该房间只需要一副牌。

② 定义一个静态的ArrayList集合存储54张牌对象，静态的集合只会加载一份。

③ 在启动游戏房间前，应该将54张牌初始化好

④ 当系统启动的同时需要准备好54张牌数据，此时可以用静态代码块完成。



1. 静态代码块的作用是什么?

- **如果要在启动系统时对静态资源进行初始化，则建议使用静态代码块完成数据的初始化操作。**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
  - 设计模式、单例模式介绍、饿汉单例模式
  - 懒汉单例模式
- **面向对象三大特征之二：继承**

什么是设计模式（Design pattern）

- 开发中经常遇到一些问题，一个问题通常有n种解法的，但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。
- 学设计模式主要是学2点：
  - 第一：这种模式用来解决什么问题。
  - 第二：遇到这种问题了，该模式是怎么写的，他是如何解决这个问题的。

单例模式

- 可以保证系统中，应用该模式的这个类永远只有一个实例，即一个类永远只能创建一个对象。
- 例如任务管理器对象我们只需要一个就可以解决问题了，这样可以节省内存空间。

单例的实现方式很多

- 饿汉单例模式。
- 懒汉单例模式。
- ...
- ...

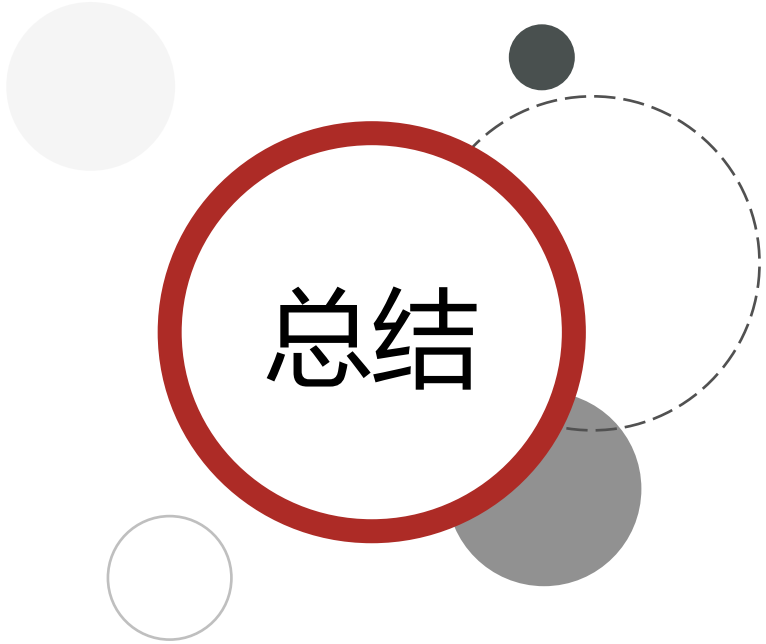
饿汉单例设计模式

- 在用类获取对象的时候，对象已经提前为你创建好了。

设计步骤：

- 定义一个类，把构造器私有。
- 定义一个静态变量存储一个对象。

```
/** a、定义一个单例类 */
public class SingleInstance {
    /** c.定义一个静态变量存储一个对象即可 :属于类，与类一起加载一次 */
    public static SingleInstance instance = new SingleInstance ();

    /** b.单例必须私有构造器*/
    private SingleInstance (){
        System.out.println("创建了一个对象");
    }
}
```



1. 饿汉单例的实现步骤?

- **定义一个类，把构造器私有。**
- **定义一个静态变量存储一个对象**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
  - 设计模式、单例模式介绍、饿汉单例模式
  - 懒汉单例模式
- **面向对象三大特征之二：继承**

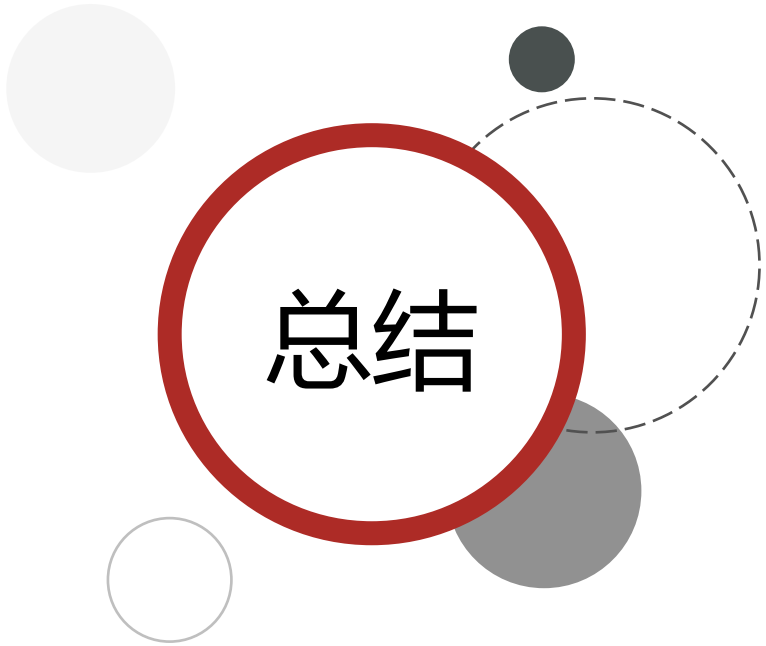
懒汉单例设计模式

- 在真正需要该对象的时候，才去创建一个对象(延迟加载对象)。

设计步骤：

- 定义一个类，把构造器私有。
- 定义一个静态变量存储一个对象。
- 提供一个返回单例对象的方法

```
/** 定义一个单例类 */
class SingleInstance{
    /** 定义一个静态变量存储一个对象即可 :属于类，与类一起加载一次 */
    public static SingleInstance instance ; // null

    /** 单例必须私有构造器*/
    private SingleInstance(){}

    /** 必须提供一个方法返回一个单例对象  */
    public static SingleInstance getInstance(){
        ...
        return ...;
    }
}
```



1. 懒汉单例的实现步骤?

- **定义一个类，把构造器私有。**
- **定义一个静态变量存储一个对象。**
- **提供一个返回单例对象的方法**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

什么是继承？

- Java中提供一个关键字extends，用这个关键字，我们可以让一个类和另一个类建立起父子关系。

```
public class Student extends People {}
```

- Student称为子类（派生类），People称为父类(基类 或超类)。
- 作用：当子类继承父类后，就可以直接使用父类公共的属性和方法了

使用继承的好处

- 可以提高代码的复用性。

为什么用继承？

案例练习：请阅读以下代码存在的问题，并使用继承这个技术进行优化

```
public class Student{
    private String name;
    private int age;

    public void study(){
        System.out.println("努力学习");
    }

    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;
    }
}
```

相同的属性

相同特征

相同的方法

```
public class Teacher {
    private String name;
    private int age;

    public void teach(){
        System.out.println("教书育人");
    }

    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;
    }
}
```

解决方案：把相同的属性和行为抽离出来，可以降低重复代码的书写，抽取出来放到何处呢？

```
public class Student {

    public void study(){
        System.out.println("努力学习");
    }

}
```

相同的属性

相同特征

相同的方法

```
public class Teacher {

    public void teach(){
        System.out.println("教书育人");
    }

                                    {

}
```

解决：继承（extends）关系， 好处：提高代码复用

你俩什么身份啊

我的东西你俩说用就用

```
public class People {
    private String name;
    private int age;
    public String getName() {
        return name;
    }

    public void setName(String
        this.name = name;

    public int getAge() {
        return age;

    public void setAge(int age)
        this.age = age;
```

```
public class Student {

    public void study(){
        System.out.println("努力学习");
    }
```

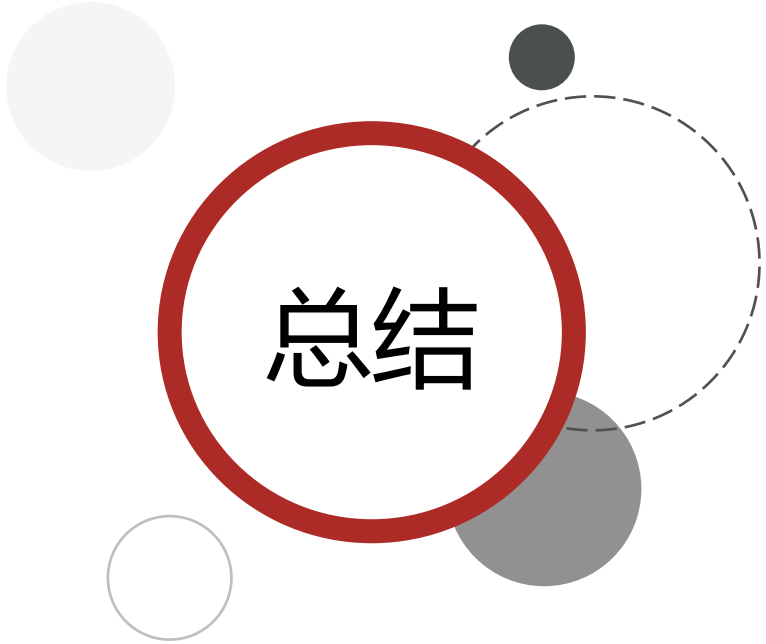
爸爸！！！

```
public class Student extends People{
    public void study(){
        System.out.println("努力学习");
    }
}
```

```
public class Teacher {

    public void teach(){
        System.out.println("教书育人");
    }
```

爸爸！！！

```
public class Teacher extends People{
    public void teach(){
        System.out.println("教书育人");
    }
}
```



1. 什么是继承? 继承的好处是啥？
   - **继承就是java允许我们用extends关键字，让一个类和另一个类建立起一种父子关系。**
   - **提高代码复用性，减少代码冗余，增强类的功能扩展性。**
2. 继承的格式
   - **子类 extends父类**
3. 继承后子类的特点？
   - **子类 继承父类，子类可以得到父类的属性和行为，子类可以使用。**
   - **Java中子类更强大**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

继承设计规范：

- 子类们相同特征（共性属性，共性方法）放在父类中定义，子类独有的的属性和行为应该定义在子类自己里面。

为什么？

- 如果子类的独有属性、行为定义在父类中，会导致其它子类也会得到这些属性和行为，这不符合面向对象逻辑。

案例 继承的设计规范

需求：

在传智教育的tlias教学资源管理系统中，存在学生、老师角色会进入系统。

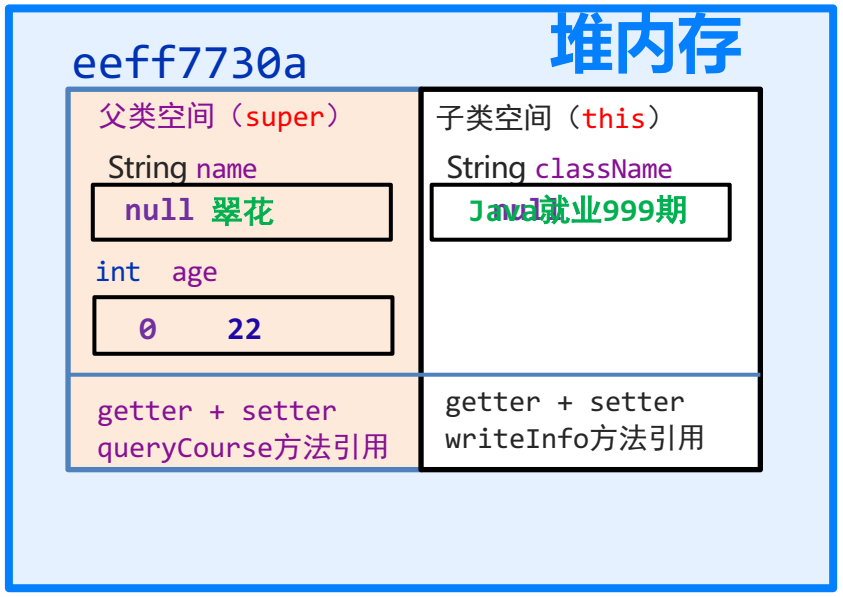
分析：

- 学生信息和行为（名称，年龄，所在班级，查看课表，填写听课反馈）
- 老师信息和行为（名称，年龄，部门名称，查看课表，发布问题）
- 定义角色类作为父类包含属性（名称，年龄），行为（查看课表）
- 定义子类：学生类包含属性（所在班级），行为（填写听课反馈）
- 定义子类：老师类包含属性（部门名称），行为（发布问题）

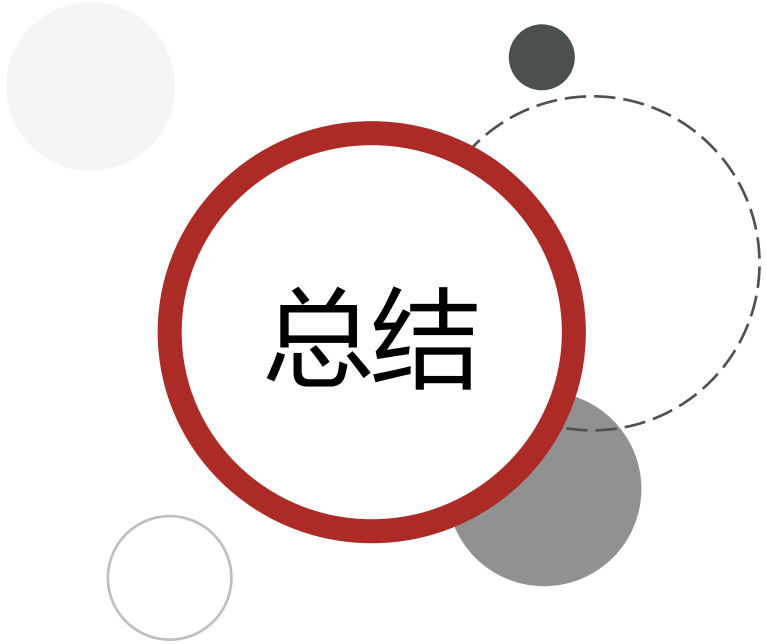
```java
public class People {
    private String name;
    private int age;

    /**
      共同行为
     */
    public void queryCourse(){
        System.out.println(name  + "，您可以开始查看您的课表信息了~~~");
    }

    // getter + setter
}
```

```java
public class Student extends People{
    private String className;
    /**
        独有行为
     */
    public void writeInfo(){
        System.out.println(getName() + "今天自己棒棒的，老师也是666~~~");
    }

    // getter + setter

}
```

```java
public class Test {
    public static void main(String[] args) {
        Student s = new Student();
        s.setName("翠花"); // 父类的
        s.setAge(22); // 父类的
        s.setClassName("Java就业999期");// 子类的
        System.out.println(s.getName());
        System.out.println(s.getAge());
        System.out.println(s.getClassName());
        s.queryCourse(); // 父类的
        s.writeInfo(); // 子类的
    }
}
```




```
翠花
22
Java就业999期
翠花，您可以开始查看您的课表信息了~~~
翠花今天自己棒棒的，老师也是666~~~
```



1. 继承需要满足什么样的设计规范？

- **子类们相同特征（共性属性，共性方法）放在父类中定义。**
- **子类独有的的属性和行为应该定义在子类自己里面。**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

继承的特点

① 子类可以继承父类的属性和行为，但是子类不能继承父类的构造器。

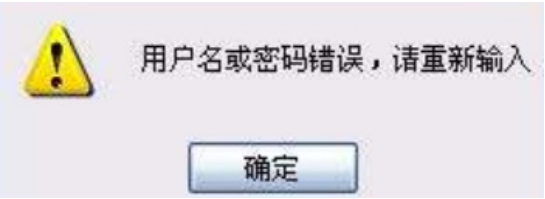
② Java是单继承模式：一个类只能继承一个直接父类。

③ Java不支持多继承、但是支持多层继承。

④ Java中所有的类都是Object类的子类。

1、子类是否可以继承父类的构造器？

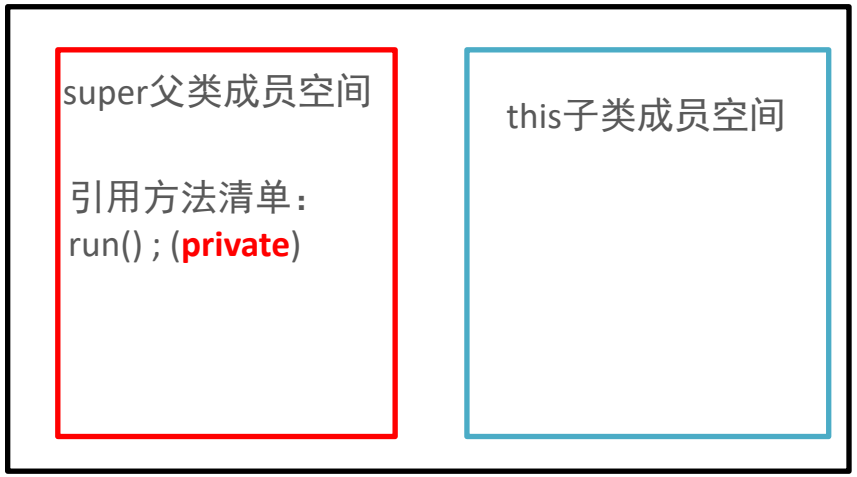
- 不可以的，子类有自己的构造器，父类构造器用于初始化父类对象。

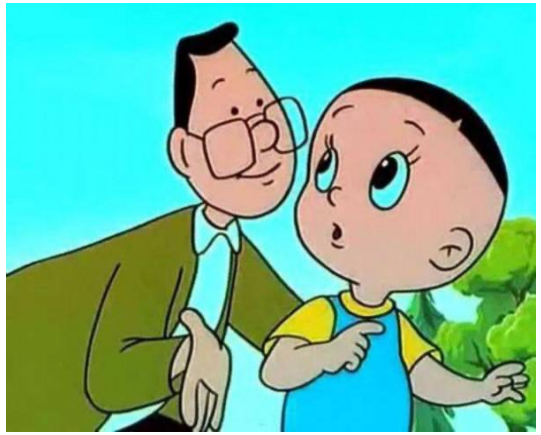


2、子类是否可以继承父类的私有成员？

- 可以的，只是不能直接访问。

子类对象002





1、子类是否可以继承父类的静态成员？

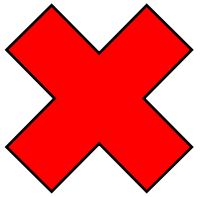
- 有争议的知识点。
- 子类可以直接使用父类的静态成员（共享）
- 但个人认为：子类不能继承父类的静态成员。（共享并非继承）

Java只支持单继承，不支持多继承。

单继承：子类只能继承一个直接父类

不支持多继承：子类不能同时继承多个父类

```
class 子类 extends 父类A , 父类B {

}
```



为何不支持多继承，请看如下反证法：

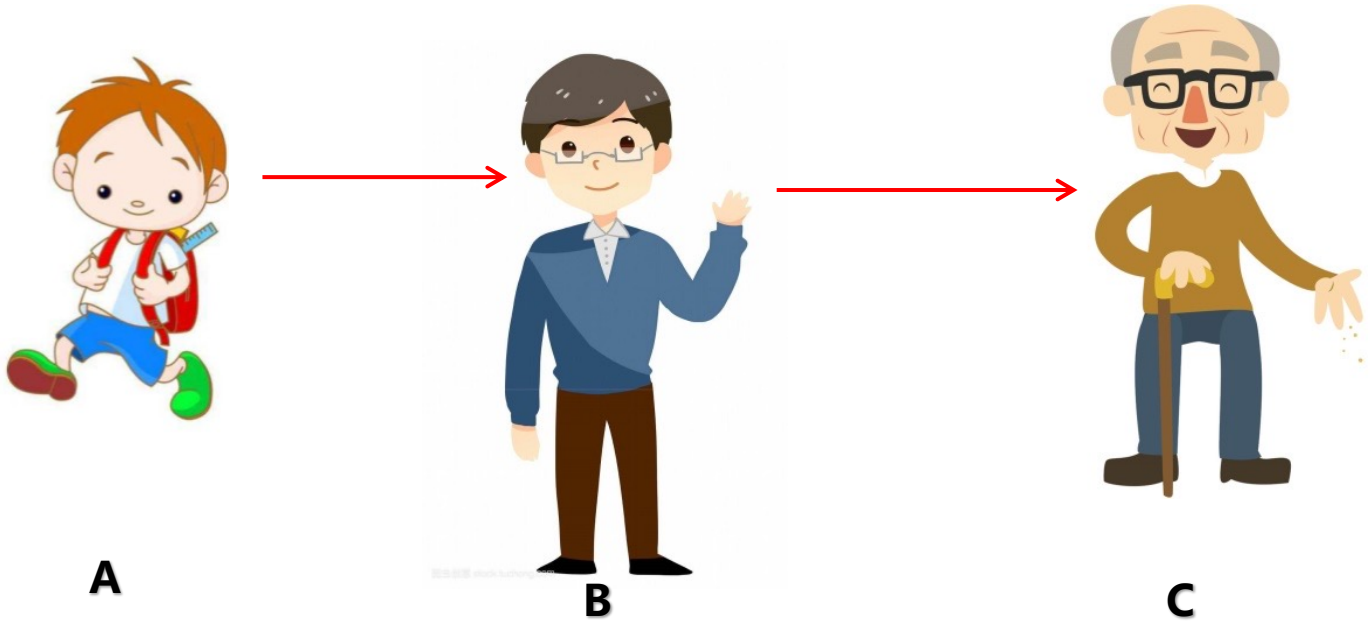
```
public class 父类A {
    public void method(){
        System.out.println("复习数学");
    }
}
```

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

```
public class 子类C extends 父类A , 父类B{
    public static void main(String[] args) {
        子类 z = new 子类();
        z.method();    // 复习啥？出现二义性，听哪个爸爸的呢？java懵了！因此不支持多继承
    }
}
```

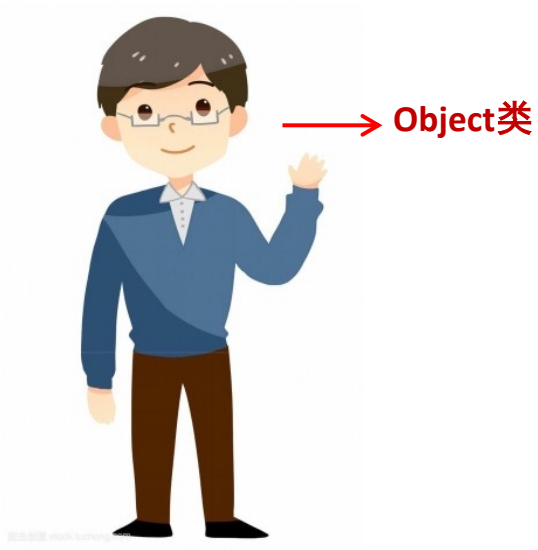
Java支持多层继承

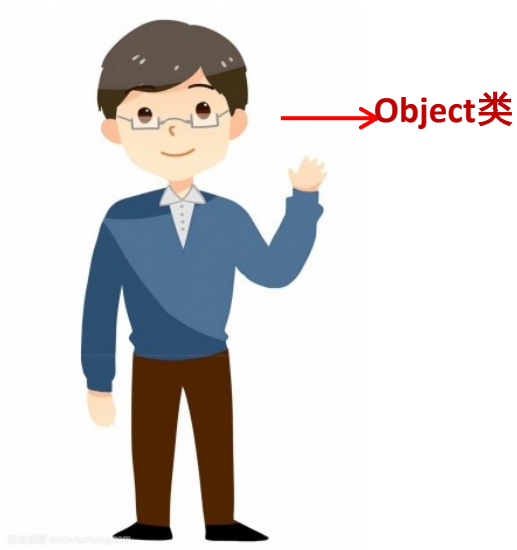
- 子类 A 继承父类 B ，父类B 可以 继承父类 C

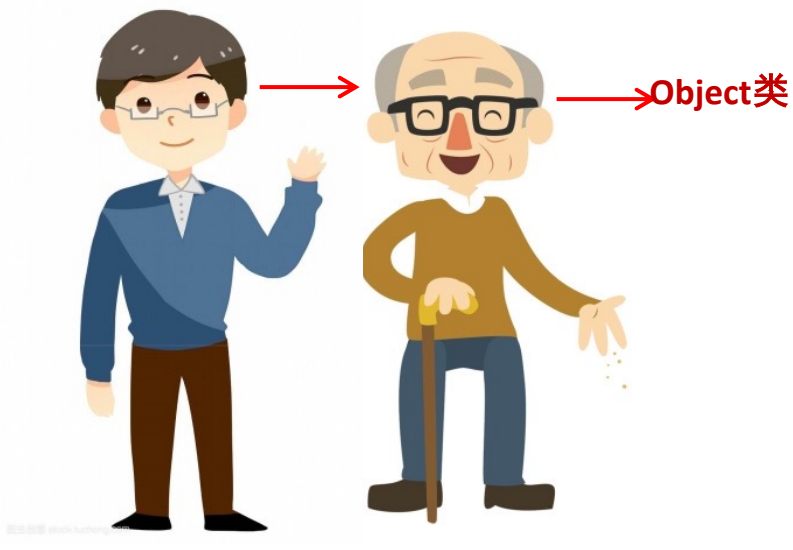


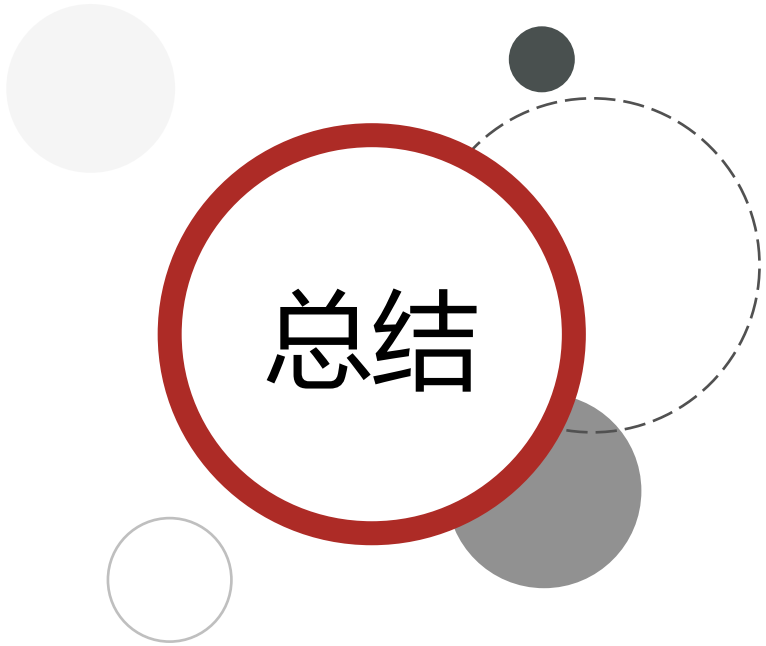
Object特点：

- Java中所有类，要么直接继承了Object，要么默认继承了Object，要么间接继承了Object, Object是祖宗类。









1. 继承有哪些特点？

① **子类可以继承父类的属性和行为，但是子类不能继承父类的构造器。**

② **Java是单继承模式：一个类只能继承一个直接父类。**

③ **Java不支持多继承、但是支持多层继承。**

④ **Java中所有的类都是Object类的子类。**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

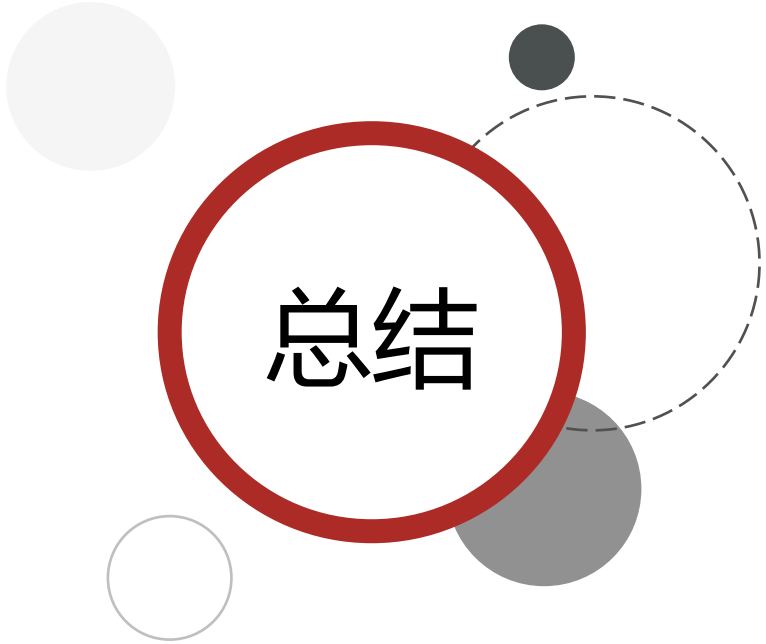
在子类方法中访问成员（成员变量、成员方法）满足：就近原则

- 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。

如果子父类中，出现了重名的成员，会优先使用子类的，此时如果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。

格式：super.父类成员变量/父类成员方法



1. 在子类方法中访问成员（成员变量、成员方法）满足：
   - **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，此时如果一定要在子类中使用父类的怎么办？

格式：super.父类成员变量/父类成员方法

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

什么是方法重写？

- 在继承体系中，子类出现了和父类中一模一样的方法声明，我们就称子类这个方法是重写的方法。

方法重写的应用场景

- 当子类需要父类的功能，但父类的该功能不完全满足自己的需求时。
- 子类可以重写父类中的方法。

案例演示：

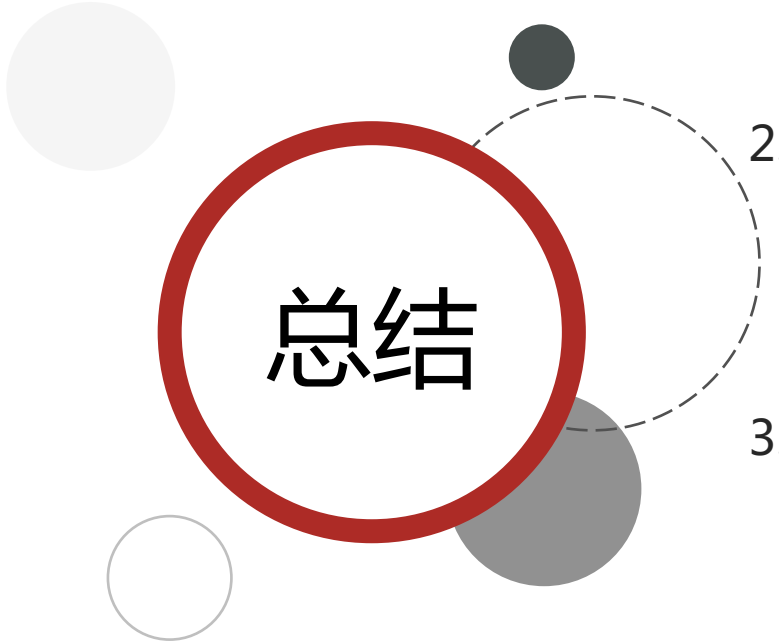
- 旧手机的功能只能是基本的打电话，发信息
- 新手机的功能需要能够：基本的打电话下支持视频通话。基本的发信息下支持发送语音和图片。

@Override重写注解

- @Override是放在重写后的方法上，作为重写是否正确的校验注解。
- 加上该注解后如果重写错误，编译阶段会出现错误提示。
- 建议重写方法都加@Override注解，代码安全，优雅！

方法重写注意事项和要求

- 重写方法的名称、形参列表必须与被重写方法的名称和参数列表一致。
- 私有方法不能被重写。
- 子类重写父类方法时，访问权限必须大于或者等于父类（暂时了解 ：缺省 < protected < public ）
- 子类不能重写父类的静态方法，如果重写会报错的。



1. 方法重写是什么样的？

   - **子类写一个与父类申明一样的方法覆盖父类的方法。**

2. 方法重写建议加上哪个注解，有什么好处？

   - **@Override注解可以校验重写是否正确，同时可读性好。**

3. 重写方法有哪些基本要求？

   - **重写方法的名称和形参列表应该与被重写方法一致。**
   - **私有方法不能被重写。**
   - **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。**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

子类继承父类后构造器的特点：

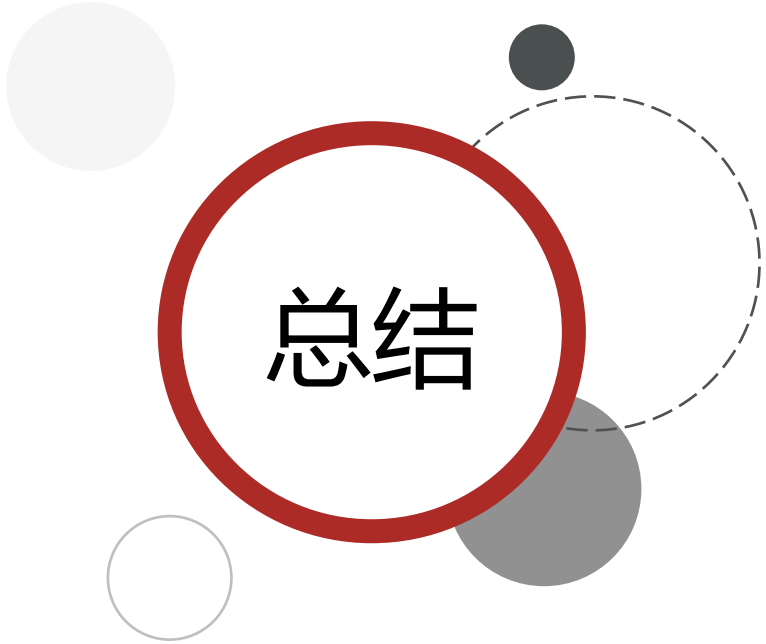
- 子类中所有的构造器默认都会先访问父类中无参的构造器，再执行自己。

为什么？

- 子类在初始化的时候，有可能会使用到父类中的数据，如果父类没有完成初始化，子类将无法使用父类的数据。
- 子类初始化之前，一定要调用父类构造器先完成父类数据空间的初始化。

怎么调用父类构造器的？

- 子类构造器的第一行语句默认都是：**super()，不写也存在。**



1. 子类继承父类后构造器的特点是什么样的？

- **子类中所有的构造器默认都会先访问父类中无参的构造器，再执行自己。**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

super调用父类有参数构造器的作用：

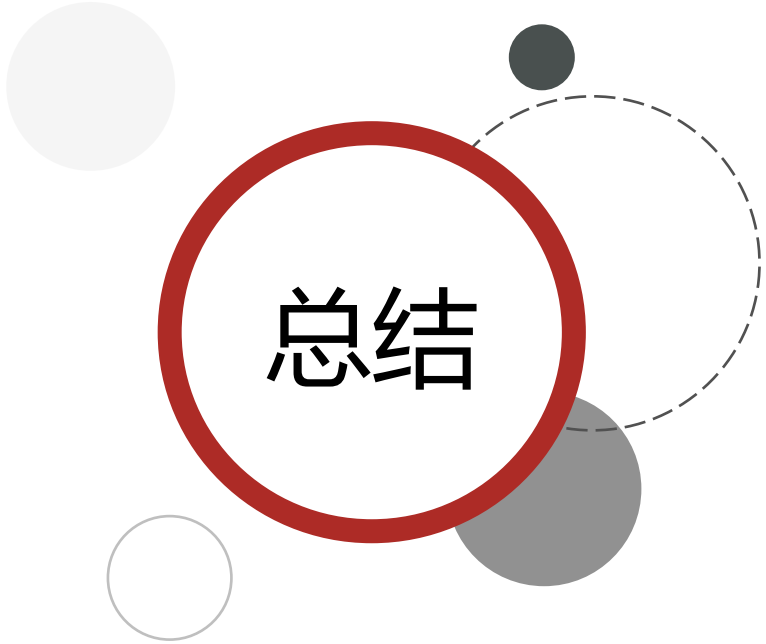
- 初始化继承自父类的数据。

如果父类中没有无参数构造器，只有有参构造器，会出现什么现象呢？

- 会报错。因为子类默认是调用父类无参构造器的。

如何解决？

- 子类构造器中可以通过书写 super(...)，手动调用父类的有参数构造器



1、super调用父类构造器的作用是什么？

- **通过调用父类有参数构造器来初始化继承自父类的数据**

目录

Contents

- **static静态关键字**
- **static应用知识：工具类**
- **static应用知识：代码块**
- **static应用知识：单例设计模式**
- **面向对象三大特征之二：继承**
  - 继承概述、使用继承的好处
  - 继承的设计规范、内存运行原理
  - 继承的特点
  - 继承后：成员变量、成员方法的访问特点
  - 继承后：方法重写
  - 继承后：子类构造器的特点
  - 继承后：子类构造器访问父类有参构造器
  - this、super使用总结

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。

案例需求：

- 在学员信息登记系统中，后台创建对象封装数据的时候如果用户没有输入学校，则默认使用“黑马培训中心”。
- 如果用户输入了学校则使用用户输入的学校信息。


```
public class Student {
    private String schoolName;
    private String name;

    public Student(String name){
          this(name , “黑马培训中心”);
    }

    public Student(String name , String schoolName ){
          this.name = name;
          this.schoolName = schoolName;
    }
}
```

this(...)和super(...)使用注意点：

- 子类通过 this (...) 去调用本类的其他构造器，本类其他构造器会通过 super 去手动调用父类的构造器，最终还是会调用父类构造器的。
- 注意：this(...) super(...) 都只能放在构造器的第一行，所以二者不能共存在同一个构造器中。

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

传智教育旗下高端IT教育品牌