1. 类和对象

1.1 类和对象的理解【理解】

客观存在的事物皆为对象 ，所以我们也常常说万物皆对象。

- 类
  - 类的理解
    - 类是对现实生活中一类具有共同属性和行为的事物的抽象
    - 类是对象的数据类型，类是具有相同属性和行为的一组对象的集合
    - 简单理解：类就是对现实事物的一种描述
  - 类的组成
    - 属性：指事物的特征，例如：手机事物（品牌，价格，尺寸）
    - 行为：指事物能执行的操作，例如：手机事物（打电话，发短信）
- 类和对象的关系
  - 类：类是对现实生活中一类具有共同属性和行为的事物的抽象
  - 对象：是能够看得到摸的着的真实存在的实体
  - 简单理解：**类是对事物的一种描述，对象则为具体存在的事物**

1.2 类的定义【应用】

类的组成是由属性和行为两部分组成

- 属性：在类中通过成员变量来体现（类中方法外的变量）
- 行为：在类中通过成员方法来体现（和前面的方法相比去掉static关键字即可）

类的定义步骤：

①定义类

②编写类的成员变量

③编写类的成员方法

```
public class 类名 {
    // 成员变量
    变量1的数据类型 变量1；
    变量2的数据类型 变量2；
    …
    // 成员方法
    方法1；
    方法2；
}
```

示例代码：

```
/*
    手机类：
        类名：
```

```
        手机(Phone)

        成员变量：
        品牌(brand)
        价格(price)

        成员方法：
        打电话(call)
        发短信(sendMessage)
 */
public class Phone {
    //成员变量
    String brand;
    int price;

    //成员方法
    public void call() {
        System.out.println("打电话");
    }

    public void sendMessage() {
        System.out.println("发短信");
    }
}
```

1.3 对象的使用【应用】

- 创建对象的格式：
  - 类名 对象名 = new 类名();
- 调用成员的格式：
  - 对象名.成员变量
  - 对象名.成员方法();
- 示例代码

```
/*
    创建对象
        格式：类名 对象名 = new 类名();
        范例：Phone p = new Phone();

    使用对象
        1：使用成员变量
            格式：对象名.变量名
            范例：p.brand
        2：使用成员方法
            格式：对象名.方法名()
            范例：p.call()
 */
public class PhoneDemo {
    public static void main(String[] args) {
        //创建对象
```

```
        Phone p = new Phone();

        //使用成员变量
        System.out.println(p.brand);
        System.out.println(p.price);

        p.brand = "小米";
        p.price = 2999;

        System.out.println(p.brand);
        System.out.println(p.price);

        //使用成员方法
        p.call();
        p.sendMessage();
    }
}
```

1.4 学生对象-练习【应用】

- 需求：首先定义一个学生类，然后定义一个学生测试类，在学生测试类中通过对象完成成员变量和成员方法的使用
- 分析：
  - 成员变量：姓名，年龄...
  - 成员方法：学习，做作业...
- 示例代码：

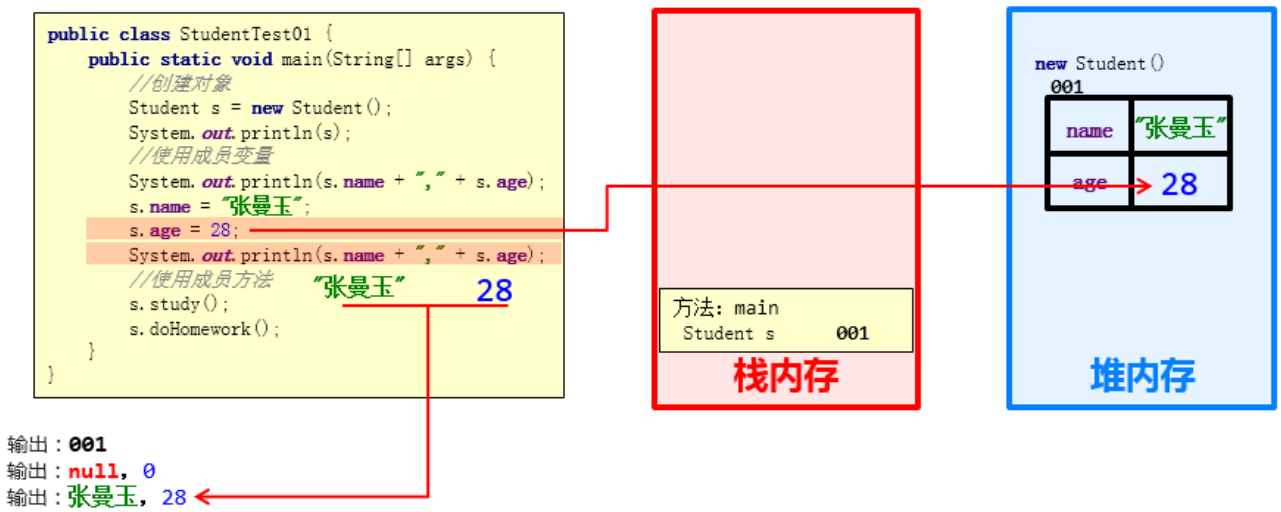
```
class Student {
    //成员变量
    String name;
    int age;

    //成员方法
    public void study() {
        System.out.println("好好学习，天天向上");
    }

    public void doHomework() {
        System.out.println("键盘敲烂，月薪过万");
    }
}
/*
    学生测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s = new Student();

        //使用对象
        System.out.println(s.name + "," + s.age);
```

```
        s.name = "林青霞";
        s.age = 30;

        System.out.println(s.name + "," + s.age);

        s.study();
        s.doHomework();
    }
}
```

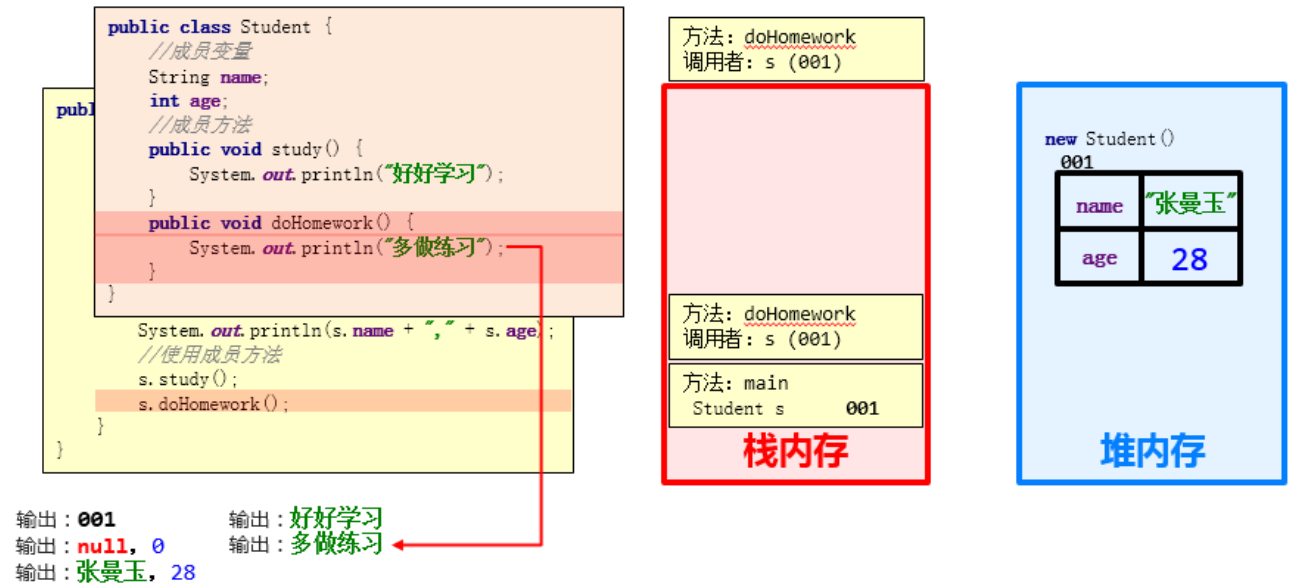
2. 对象内存图

2.1 单个对象内存图【理解】

- 成员变量使用过程

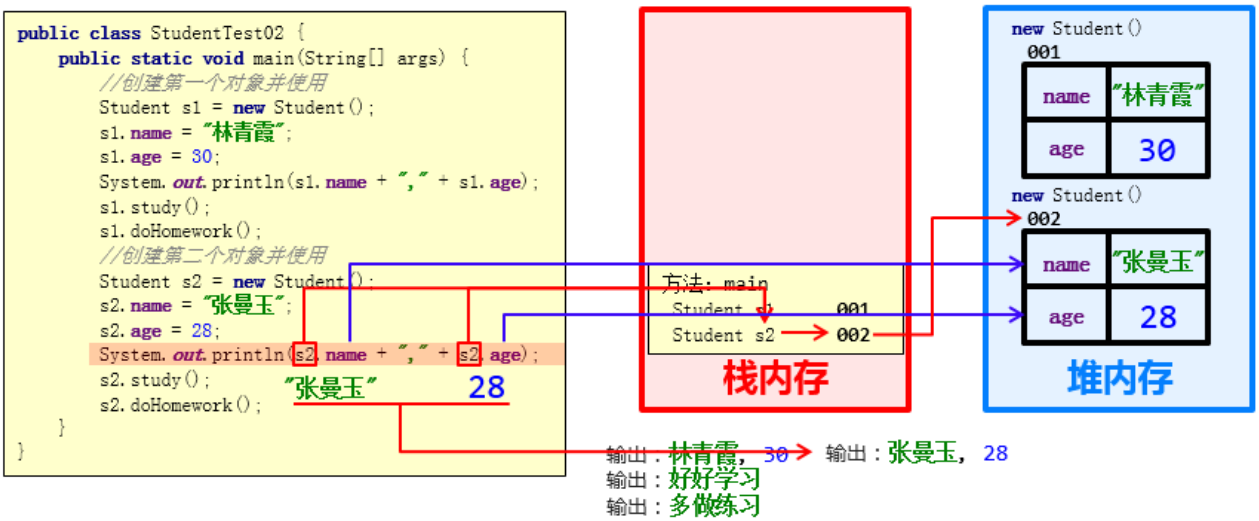


- 成员方法调用过程

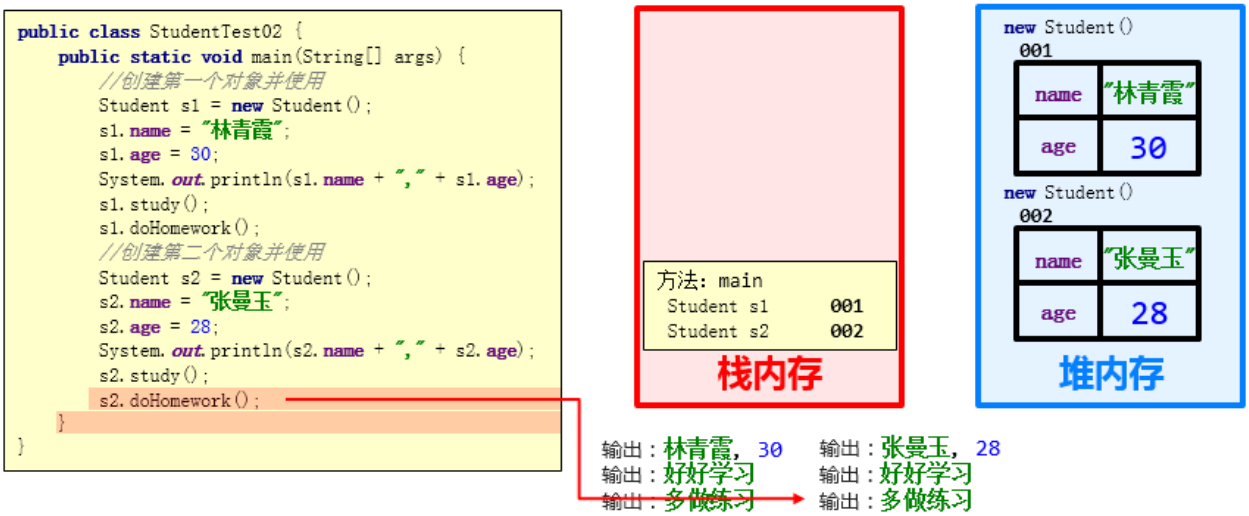


2.2 多个对象内存图【理解】

- 成员变量使用过程



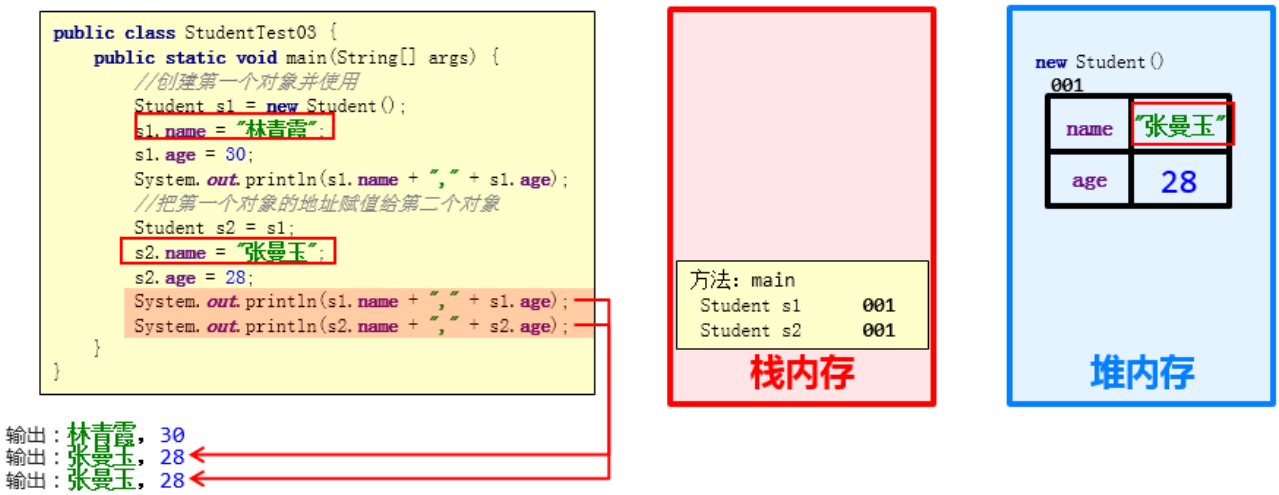
- 成员方法调用过程



- 总结：

  多个对象在堆内存中，都有不同的内存划分，成员变量存储在各自的内存区域中，成员方法多个对象共用的一份

2.3 多个对象指向相同内存图【理解】



- 总结

  当多个对象的引用指向同一个内存空间（变量所记录的地址值是一样的）

  只要有任何一个对象修改了内存中的数据，随后，无论使用哪一个对象进行数据获取，都是修改后的数据。

3. 成员变量和局部变量

3.1 成员变量和局部变量的区别【理解】

- 类中位置不同：成员变量（类中方法外）局部变量（方法内部或方法声明上）
- 内存中位置不同：成员变量（堆内存）局部变量（栈内存）
- 生命周期不同：成员变量（随着对象的存在而存在，随着对象的消失而消失）局部变量（随着方法的调用而存在，醉着方法的调用完毕而消失）
- 初始化值不同：成员变量（有默认初始化值）局部变量（没有默认初始化值，必须先定义，赋值才能使用）

4. 封装

4.1 private关键字【理解】

private是一个修饰符，可以用来修饰成员（成员变量，成员方法）

- 被private修饰的成员，只能在本类进行访问，针对private修饰的成员变量，如果需要被其他类使用，提供相应的操作
  - 提供“get变量名()”方法，用于获取成员变量的值，方法用public修饰
  - 提供“set变量名(参数)”方法，用于设置成员变量的值，方法用public修饰
- 示例代码：

```
/*
    学生类
 */
class Student {
    //成员变量
    String name;
    private int age;

    //提供get/set方法
    public void setAge(int a) {
        if(a<0 || a>120) {
            System.out.println("你给的年龄有误");
        } else {
            age = a;
        }
    }

    public int getAge() {
        return age;
    }

    //成员方法
    public void show() {
        System.out.println(name + "," + age);
```

```
    }
}
/*
    学生测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s = new Student();
        //给成员变量赋值
        s.name = "林青霞";
        s.setAge(30);
        //调用show方法
        s.show();
    }
}
```

4.2 private的使用【应用】

- 需求：定义标准的学生类，要求name和age使用private修饰，并提供set和get方法以及便于显示数据的show方法，测试类中创建对象并使用，最终控制台输出 林青霞，30
- 示例代码：

```
/*
    学生类
 */
class Student {
    //成员变量
    private String name;
    private int age;

    //get/set方法
    public void setName(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public void setAge(int a) {
        age = a;
    }

    public int getAge() {
        return age;
    }

    public void show() {
        System.out.println(name + "," + age);
    }
}
```

```
/*
    学生测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s = new Student();

        //使用set方法给成员变量赋值
        s.setName("林青霞");
        s.setAge(30);

        s.show();

        //使用get方法获取成员变量的值
        System.out.println(s.getName() + "---" + s.getAge());
        System.out.println(s.getName() + "," + s.getAge());

    }
}
```

4.3 this关键字【应用】

- this修饰的变量用于指代成员变量，其主要作用是（区分局部变量和成员变量的重名问题）
  - 方法的形参如果与成员变量同名，不带this修饰的变量指的是形参，而不是成员变量
  - 方法的形参没有与成员变量同名，不带this修饰的变量指的是成员变量

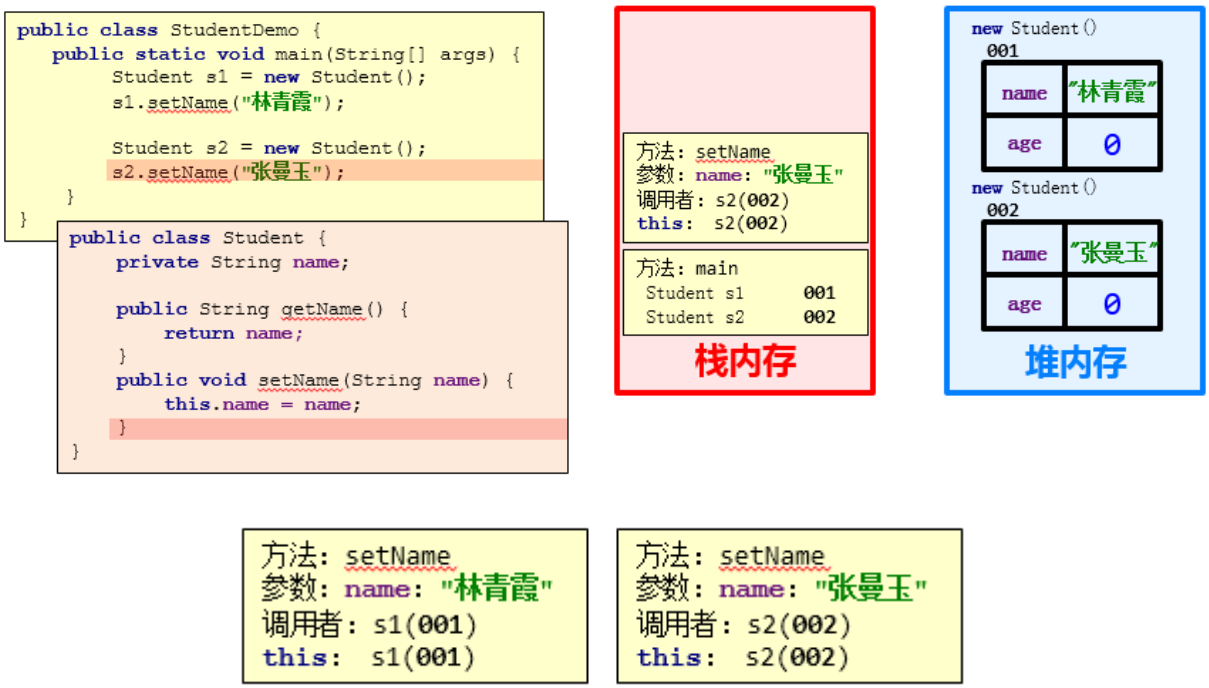
```
public class Student {
    private String name;
    private int age;

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

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void show() {
        System.out.println(name + "," + age);
    }
}
```

4.4 this内存原理【理解】

- this代表当前调用方法的引用，哪个对象调用的方法，this就代表哪一个对象
- 示例代码：

```
public class StudentDemo {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setName("林青霞");
        Student s2 = new Student();
        s2.setName("张曼玉");
    }
}
```

- 图解：



4.5 封装思想【理解】

1. 封装概述 是面向对象三大特征之一（封装，继承，多态）是面向对象编程语言对客观世界的模拟，客观世界里成员变量都是隐藏在对象内部的，外界是无法直接操作的
2. 封装原则 将类的某些信息隐藏在类内部，不允许外部程序直接访问，而是通过该类提供的方法来实现对隐藏信息的操作和访问 成员变量private，提供对应的getXxx()/setXxx()方法
3. 封装好处 通过方法来控制成员变量的操作，提高了代码的安全性 把代码用方法进行封装，提高了代码的复用性

5. 构造方法

5.1 构造方法概述【理解】

构造方法是一种特殊的方法

- 作用：创建对象 Student stu = **new Student();**
- 格式：

  public class 类名{

  修饰符 类名(参数) {

  }

  }
- 功能：主要是完成对象数据的初始化
- 示例代码：

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

    //构造方法
    public Student() {
        System.out.println("无参构造方法");
    }

    public void show() {
        System.out.println(name + "," + age);
    }
}
/*
    测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s = new Student();
        s.show();
    }
}
```

5.2 构造方法的注意事项【理解】

- 构造方法的创建

如果没有定义构造方法，系统将给出一个默认的无参数构造方法 如果定义了构造方法，系统将不再提供默认的构造方法

- 构造方法的重载

如果自定义了带参构造方法，还要使用无参数构造方法，就必须再写一个无参数构造方法

- 推荐的使用方式

无论是否使用，都手工书写无参数构造方法

- 重要功能！

可以使用带参构造，为成员变量进行初始化

- 示例代码

```
/*
    学生类
 */
class Student {
    private String name;
    private int age;

    public Student() {}

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

    public Student(int age) {
        this.age = age;
    }

    public Student(String name,int age) {
        this.name = name;
        this.age = age;
    }

    public void show() {
        System.out.println(name + "," + age);
    }
}
/*
    测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s1 = new Student();
        s1.show();

        //public Student(String name)
        Student s2 = new Student("林青霞");
        s2.show();

        //public Student(int age)
        Student s3 = new Student(30);
        s3.show();

        //public Student(String name,int age)
        Student s4 = new Student("林青霞",30);
        s4.show();
    }
}
```

5.3 标准类制作【应用】

- 需求：定义标准学生类，要求分别使用空参和有参构造方法创建对象，空参创建的对象通过setXxx赋值，有参创建的对象直接赋值，并通过show方法展示数据。
- 示例代码：

```
class Student {
    //成员变量
    private String name;
    private int age;

    //构造方法
    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    //成员方法
    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void show() {
        System.out.println(name + "," + age);
    }
}
/*
    创建对象并为其成员变量赋值的两种方式
        1:无参构造方法创建对象后使用setXxx()赋值
        2:使用带参构造方法直接创建带有属性值的对象
*/
public class StudentDemo {
    public static void main(String[] args) {
        //无参构造方法创建对象后使用setXxx()赋值
        Student s1 = new Student();
        s1.setName("林青霞");
        s1.setAge(30);
        s1.show();

        //使用带参构造方法直接创建带有属性值的对象
```

```
        Student s2 = new Student("林青霞",30);
        s2.show();
    }
}
```