第六章 类的封装

1. 为什么要学习封装

- 没有封装带来了安全隐患，实例变量赋值可能存在不合法

代码块

```
package com.powernode.private05;

class Teacher{
    String name = "zs";
    int age = 23;
    char sex = '男';

}
public class Test {
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
        teacher.sex = '女';
        System.out.println(teacher.sex);
        /**
         * 其他类拿到了实例变量赋值不合法
         * 可以使用封装来解决
         */
        teacher.sex = 'X';
        System.out.println(teacher.sex);

    }
}
```

2. 封装访问权限修饰符

1. 封装是一种访问权限的控制
2. 控制成员是否可以被其他类访问
3. 封装修饰符：private（私有的）

3. 封装来解决，赋值不合法问题

- 对字段封装，通常情况下的步骤（有一些情况，封装是不需要提供set或者get方法的）：

- 封装：private 修饰字段
- 赋值：*提供set方法赋值*
- 取值：*提供get方法获得值*

代码块

```
package com.powernode.private06;

class Teacher{
    String name = "zs";
    int age = 23;
    //1.封装字段：private 修饰符 修饰字段，使用private修饰后，该字段只能本类访问
    private char sex = '男';
    //2.提供set方法赋值：注意：命名规范：set + 字段名称首字母大写
    public void setSex(char newSex){//set访问内部需要用到外界的数据，所以要提供参数
        if (newSex != '男' && newSex != '女') {//如果性别不是男且不是女
            System.out.println("性别不合法");
        }else{//性别是男或者是女
            sex = newSex;
        }
    }
    //3.提供get方法获得值
    public char getSex(){
        return sex;
    }
}
public class Test {
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
        //teacher.sex = 'X';
        teacher.setSex('女');
        System.out.println(teacher.getSex());
    }
}
```

```
    int age = 23;  no usages
    //1.封装字段：private 修饰符 修饰字段，使用private修饰后，该字段只能本类访问
    private char sex = '男';  2 usages
    //2.提供set方法修饰值：注意：命名规范：set + 字段名称首字母大写
    public void setSex(char newSex){//set访问内部需要用到外界的数据，所以要提供参数  1 usage
        if (newSex != '男' && newSex != '女') {//如果性别不是男且不是女
            System.out.println("性别不合法");
        }else{//性别是男或者是女
            sex = newSex;
        }
    }
    //3.提供get方法获得值
    public char getSex(){  1 usage
        return sex;
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
        //teacher.sex = 'X';
        teacher.setSex('女');
        System.out.println(teacher.getSex());
    }
}
```

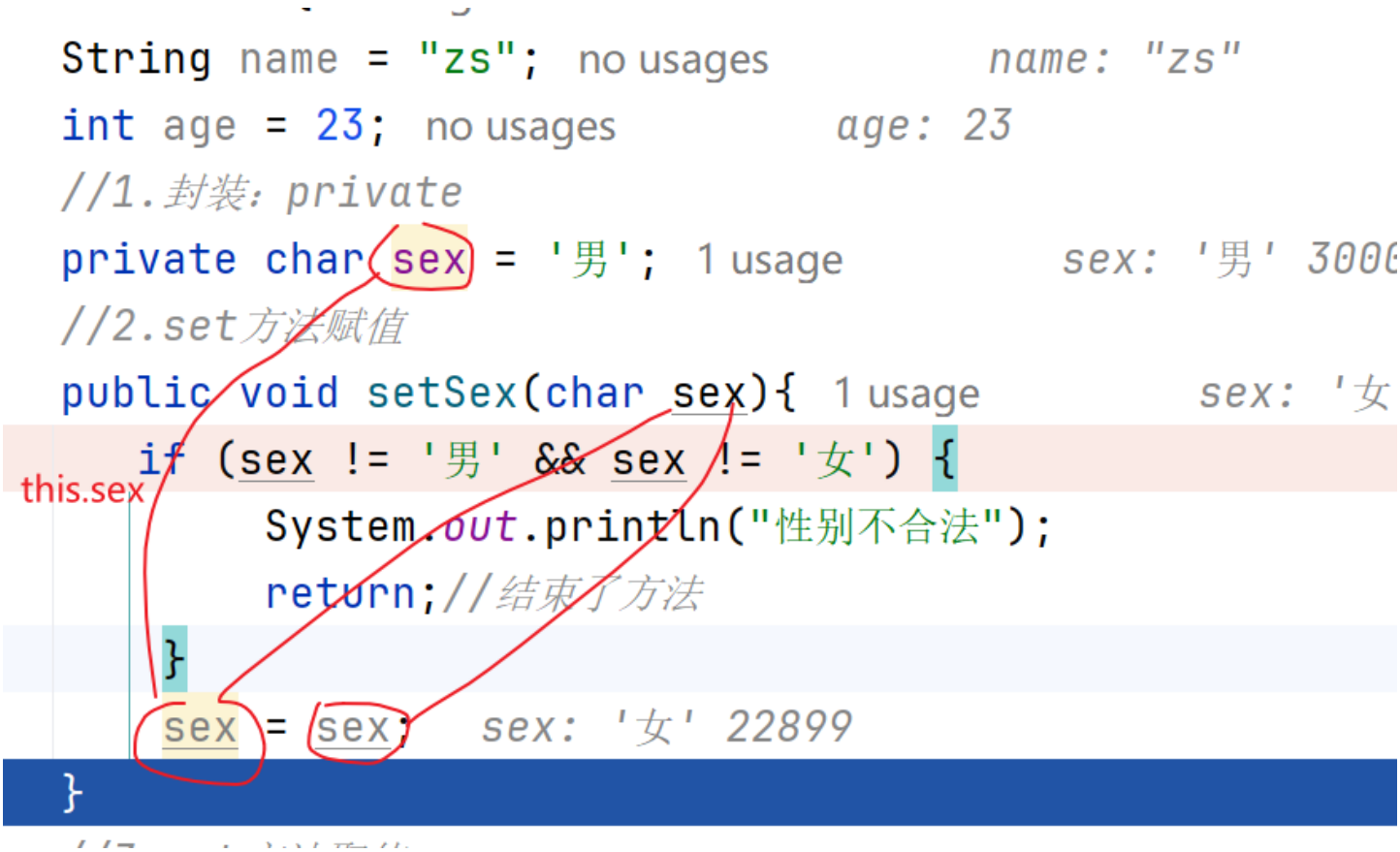
4. set方法优化

代码块

```
package com.powernode.private07;
class Teacher{
    String name = "zs";
    int age = 23;
    //1.封装: private
    private char sex = '男';
    //2.set方法赋值
    public void setSex(char newSex){
        //第一种方案:
       /* if (newSex != '男' && newSex != '女') {
            System.out.println("性别不合法");
        }else{
            sex = newSex;
        }*/
        //第二种方案:(需要修改数据类型--放弃)
        //sex = newSex != '男' && newSex != '女' ? "性别不合法" : newSex;
        //第三种方案:
        if (newSex != '男' && newSex != '女') {
            System.out.println("性别不合法");
            return;//结束了方法
        }
        sex = newSex;


    }
    //3.get方法取值
```

```
27      public char getSex() {
28          return sex;
29      }
30  }
31  public class Test {
32      public static void main(String[] args) {
33          Teacher teacher = new Teacher();
34          teacher.setSex('女');
35          System.out.println(teacher.getSex());
36      }
37  }
```

5. 使用this区分实例变量和局部变量



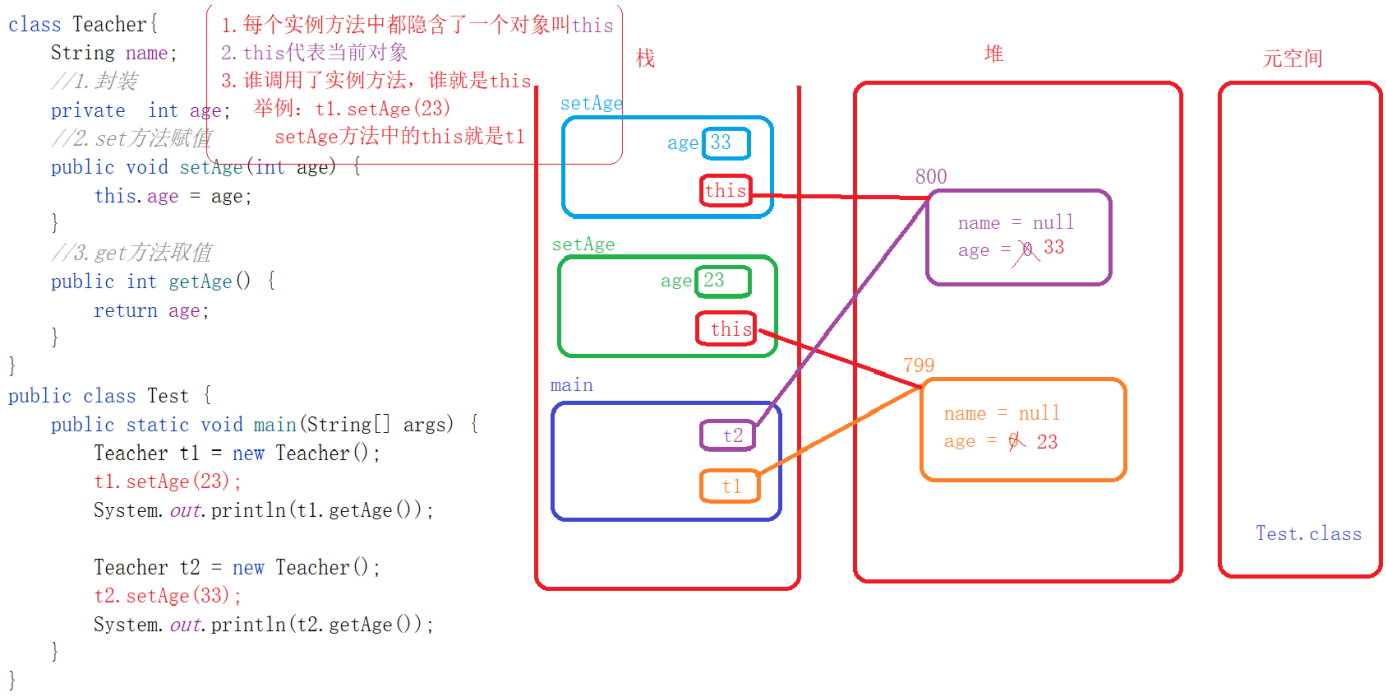
代码块

```
1   package com.powernode.private08;
2   class Teacher{
3       String name = "zs";
4       int age = 23;
5       //1.封装: private
6       private char sex = '男';
7       //2.set方法赋值
8
9       /**
```

```
10          * 1.局部变量：通过方法声明的变量称为局部变量
11          * 2.实例变量：类的属性
12          * 3.区分变量：可以是this.变量：表示实例变量
13          *
14          */
15      public void setSex(char sex){
16          if (sex != '男' && sex != '女') {
17              System.out.println("性别不合法");
18              return;//结束了方法
19          }
20          this.sex = sex;
21      }
22      //3.get方法取值
23      public char getSex() {
24          return sex;
25      }
26  }
27  public class Test {
28      public static void main(String[] args) {
29          Teacher teacher = new Teacher();
30          teacher.setSex('女');
31          System.out.println(teacher.getSex());
32      }
33  }
```

6. this为什么可以区分实例和局部变量

1. 每个实例方法中都隐含了一个this对象
2. this代表是当前对象，哪个对象调用了实例方法，那个对象就是this
3. 因此this可以区分实例变量和局部变量



代码块

```
package com.powernode.private09;
class Teacher{
    String name;
    //1.封装
    private  int age;
    //2.set方法赋值
    public void setAge(int age) {
        this.age = age;
    }
    //3.get方法取值
    public int getAge() {
        return age;
    }
}
public class Test {
    public static void main(String[] args) {
        Teacher t1 = new Teacher();
        t1.setAge(23);
        System.out.println(t1.getAge());

        Teacher t2 = new Teacher();
        t2.setAge(33);
        System.out.println(t2.getAge());
    }
}
```

7. 权限修饰符

代码块

```
package com.powernode.private10;
class Teacher{
    /**
     * - 权限修饰符一共有四个（先知道两个）
     *   1.private(权限最小，只能本类中访问)
     *   2.public （权限最大）
     *   3.private 可以修饰：
     *      1.字段
     *      2.方法
     *   4.Modify:单词，修饰
     *   5.权限修饰符：只要是修饰属性和方法的，也可以修饰类（private不可以）,不可以修饰局部变量
     */
    private String name;

    public void setName(String name) {
        this.name = name;
    }
    //只能本类访问
    private String getName() {
        return name;
    }
}
public class Test {
}
```

8. 随堂练习

1. 编写一个Dog类，提供name，age和sex属性并赋值
2. 使用private 对属性进行封装
3. 提供set方法修改属性值
4. 提供get方法获得属性值
5. 编写Test类
    a. 创建Dog对象
    b. 对name和sex进行修改（setName,setSex）

c. 获取修改过的属性值并输出

代码块

```
package com.powernode.private11;
class Dog {
    private String name = "旺财";
    private int age = 23;
    private char sex = '女';

    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 setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }
}
public class Test {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.setName("汪汪");
        dog.setSex('男');
        System.out.println(dog.getName());
        System.out.println(dog.getSex());

    }
}
```

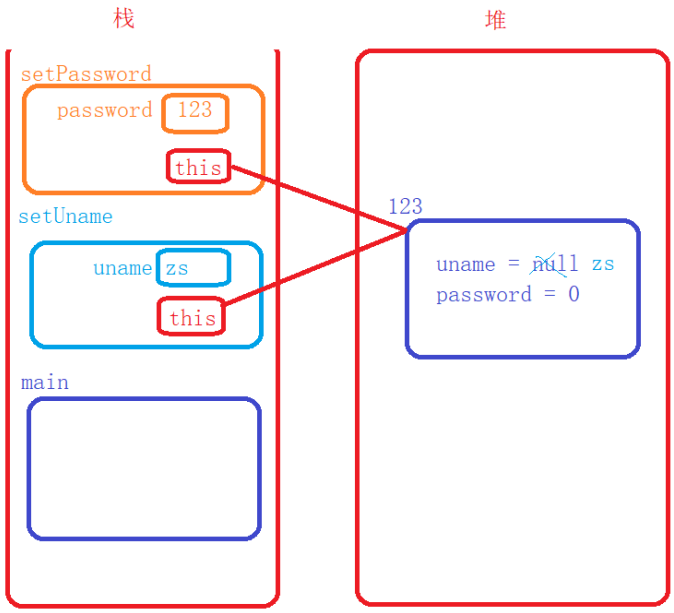
9. 链式调用

代码块

```
package com.powernode.private12;

class User{
    private String uname;
    private int password;

    /*public void setUname(String uname) {
        this.uname = uname;
    }

    public void setPassword(int password) {
        this.password = password;
    }*/
    public User setUname(String uname) {
        this.uname = uname;
        return this;
    }

    public User setPassword(int password) {
        this.password = password;
        return this;
    }
}

public class Test {
    public static void main(String[] args) {
       /* User user = new User();
        user.setUname("zs");
        user.setPassword(123);*/
        /*User user = new User();
        user.setUname("zs").setPassword(123);*/
        //使用链式调用可以解决，匿名对象只用一次的问题
        new User().setUname("zs").setPassword(123);
    }
}
```

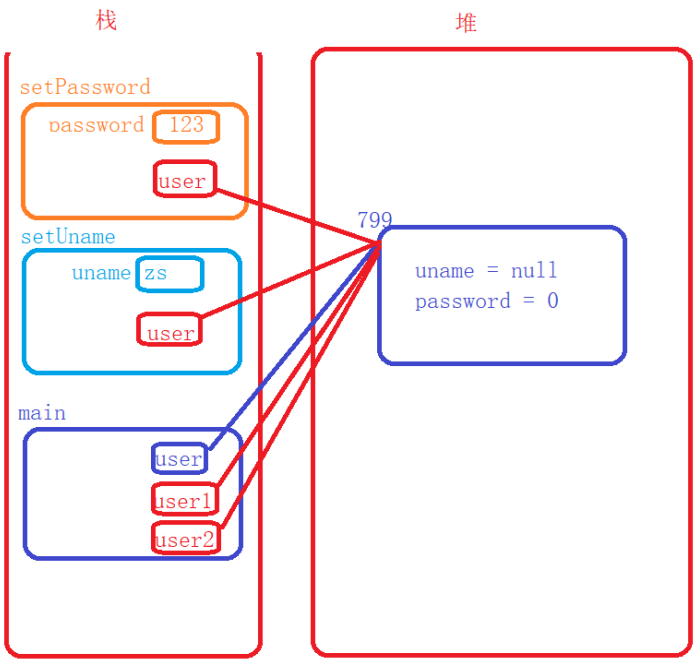
```java
package com.powernode.private12;

class User{
    private String uname;
    private int password;

    public User setUname(String uname) {
        this.uname = uname;
        return this;
    }

    public User setPassword(int password) {
        this.password = password;
        return this;
    }
}

public class Test {
    public static void main(String[] args) {
        new User().setUname("zs").setPassword(123);
    }
}
```

this对象
就是user对象



10. 链式调用（对象传递方式）

```java
class User{
    private String uname;
    private int password;
    public User setUname(String uname,User user) {
        this.uname = uname;
        return user;
    }

    public User setPassword(int password,User user) {
        this.password = password;
        return user;
    }
}

public class Test {
    public static void main(String[] args) {
        User user = new User();
        User user1 = user.setUname("zs", user);
        User user2 =user1.setPassword(123, user);

        System.out.println(user == user1);
        System.out.println(user1 == user2);

    }
}
```



代码块

```java
package com.powernode.private13;

class User{
    private String uname;
    private int password;
```

```java

    public User setUname(String uname,User user) {
        this.uname = uname;
        return user;
    }

    public User setPassword(int password,User user) {
        this.password = password;
        return user;
    }
}

public class Test {
    public static void main(String[] args) {
       /* User user = new User();
        user.setUname("zs");
        user.setPassword(123);*/
       /* User user = new User();
        User user1 = user.setUname("zs", user);
        User user2 = user1.setPassword(123, user);

        System.out.println(user == user1);
        System.out.println(user1 == user2);*/
        User user = new User();
        user.setUname("zs", user).setPassword(123, user);

    }
}
```