

# 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

```
1  package com.powernode.private05;
2
3  class Teacher{
4      String name = "zs";
5      int age = 23;
6      char sex = '男';
7
8  }
9  public class Test {
10     public static void main(String[] args) {
11         Teacher teacher = new Teacher();
12         teacher.sex = '女';
13         System.out.println(teacher.sex);
14         /**
15          * 其他类拿到了实例变量赋值不合法
16          * 可以使用封装来解决
17          */
18         teacher.sex = 'X';
19         System.out.println(teacher.sex);
20
21     }
22 }
```

## 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

```
1  package com.powernode.private06;
2
3  class Teacher{
4      String name = "zs";
5      int age = 23;
6      //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
7      private char sex = '男';
8      //2.提供set方法赋值: 注意: 命名规范: set + 字段名称首字母大写
9      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
10         if (newSex != '男' && newSex != '女') {//如果性别不是男且不是女
11             System.out.println("性别不合法");
12         }else{//性别是男或者是女
13             sex = newSex;
14         }
15     }
16     //3.提供get方法获得值
17     public char getSex(){
18         return sex;
19     }
20 }
21 public class Test {
22     public static void main(String[] args) {
23         Teacher teacher = new Teacher();
24         //teacher.sex = 'X';
25         teacher.setSex('女');
26         System.out.println(teacher.getSex());
27     }
28 }
```

```

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

```

1  package com.powernode.private07;
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      public void setSex(char newSex){
9          //第一种方案:
10         /* if (newSex != '男' && newSex != '女') {
11             System.out.println("性别不合法");
12         } else {
13             sex = newSex;
14         } */
15         //第二种方案: (需要修改数据类型--放弃)
16         //sex = newSex != '男' && newSex != '女' ? "性别不合法" : newSex;
17         //第三种方案:
18         if (newSex != '男' && newSex != '女') {
19             System.out.println("性别不合法");
20             return; //结束了方法
21         }
22         sex = newSex;
23
24
25     }
26     //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

```

String name = "zs"; no usages           name: "zs"
int age = 23; no usages                 age: 23
//1.封装: private
private char sex = '男'; 1 usage        sex: '男' 3006
//2.set方法赋值
public void setSex(char sex){ 1 usage    sex: '女'
    if (sex != '男' && sex != '女') {
        System.out.println("性别不合法");
        return; // 结束了方法
    }
    sex = sex; sex: '女' 22899
}

```

Diagram illustrating variable scope and usage:

- `sex` is a private instance variable (1 usage).
- `setSex(char sex)` is a public method (1 usage).
- Inside `setSex`, the parameter `sex` is a local variable.
- The `if` statement checks if the local `sex` is not '男' and not '女'.
- The `return` statement ends the method.
- After the method call, the instance variable `sex` is updated to '女' (22899).

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     /**

```

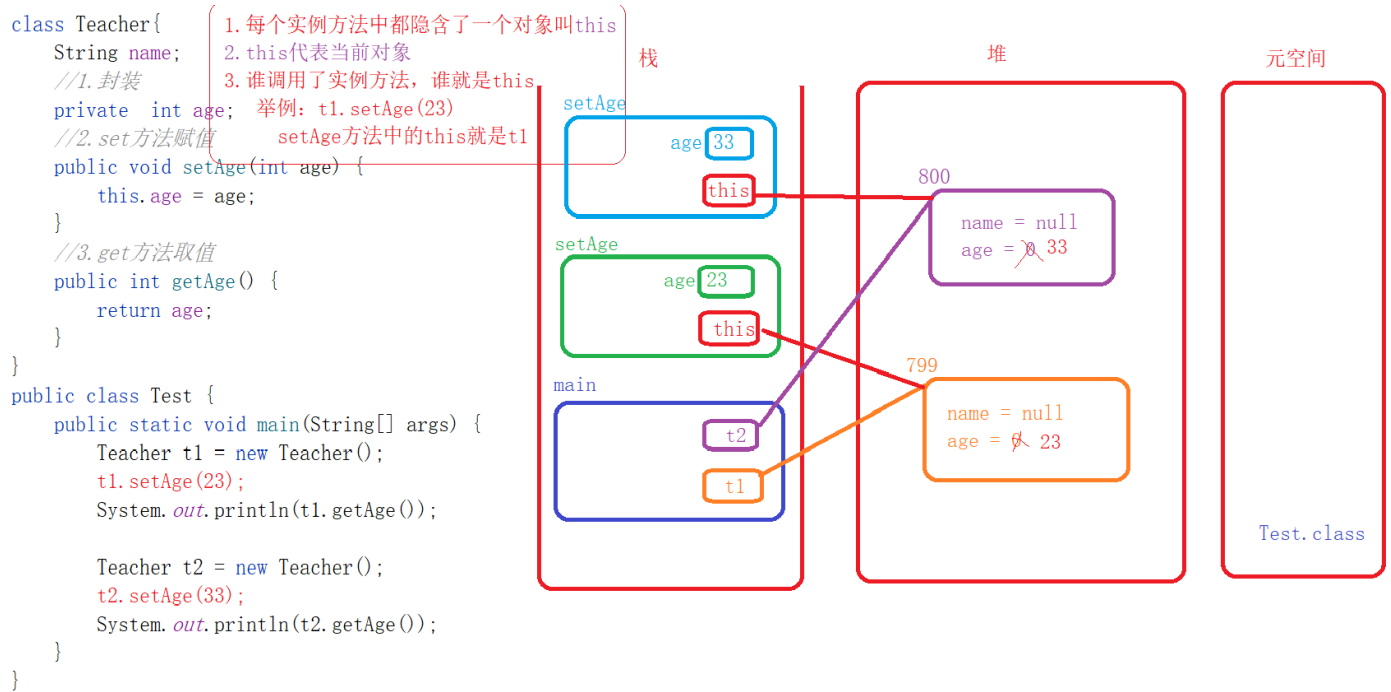
```

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

```

1  package com.powernode.private09;
2  class Teacher{
3      String name;
4      //1.封装
5      private int age;
6      //2.set方法赋值
7      public void setAge(int age) {
8          this.age = age;
9      }
10     //3.get方法取值
11     public int getAge() {
12         return age;
13     }
14 }
15 public class Test {
16     public static void main(String[] args) {
17         Teacher t1 = new Teacher();
18         t1.setAge(23);
19         System.out.println(t1.getAge());
20
21         Teacher t2 = new Teacher();
22         t2.setAge(33);
23         System.out.println(t2.getAge());
24     }
25 }

```

## 7. 权限修饰符

代码块

```
1  package com.powernode.private10;
2  class Teacher{
3      /**
4       * - 权限修饰符一共有四个（先知道两个）
5       *   1.private(权限最小，只能本类中访问)
6       *   2.public (权限最大)
7       *   3.private 可以修饰：
8       *       1.字段
9       *       2.方法
10      *   4.Modify:单词，修饰
11      *   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
12      */
13      private String name;
14
15      public void setName(String name) {
16          this.name = name;
17      }
18      //只能本类访问
19      private String getName() {
20          return name;
21      }
22  }
23  public class Test {
24  }
```

## 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

```
1  package com.powernode.private11;
2  class Dog {
3      private String name = "旺财";
4      private int age = 23;
5      private char sex = '女';
6
7      public void setName(String name){
8          this.name = name;
9      }
10     public String getName(){
11         return name;
12     }
13
14     public void setAge(int age) {
15         this.age = age;
16     }
17
18     public int getAge() {
19         return age;
20     }
21
22     public void setSex(char sex) {
23         this.sex = sex;
24     }
25
26     public char getSex() {
27         return sex;
28     }
29 }
30 public class Test {
31     public static void main(String[] args) {
32         Dog dog = new Dog();
33         dog.setName("汪汪");
34         dog.setSex('男');
35         System.out.println(dog.getName());
36         System.out.println(dog.getSex());
37     }
38 }
39 }
```

## 9. 链式调用



```
1 package com.powernode.private12;
2
3 class User{
4     private String uname;
5     private int password;
6
7     /*public void setUname(String uname) {
8         this.uname = uname;
9     }
10
11     public void setPassword(int password) {
12         this.password = password;
13     }*/
14     public User setUname(String uname) {
15         this.uname = uname;
16         return this;
17     }
18
19     public User setPassword(int password) {
20         this.password = password;
21         return this;
22     }
23 }
24
25 public class Test {
26     public static void main(String[] args) {
27         /* User user = new User();
28         user.setUname("zs");
29         user.setPassword(123);*/
30         /*User user = new User();
31         user.setUname("zs").setPassword(123);*/
32         //使用链式调用可以解决，匿名对象只用一次的问题
33         new User().setUname("zs").setPassword(123);
34     }
35 }
```

```
package com.powernode.private12;
```

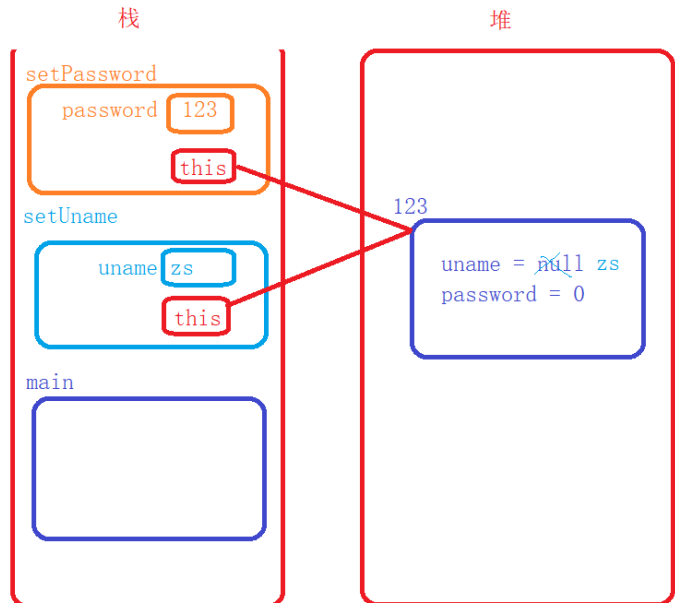
```
class User{
    private String uname;
    private int password;

    public User setName(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().setName("zs").setPassword(123);
    }
}
```

this对象  
就是user对象



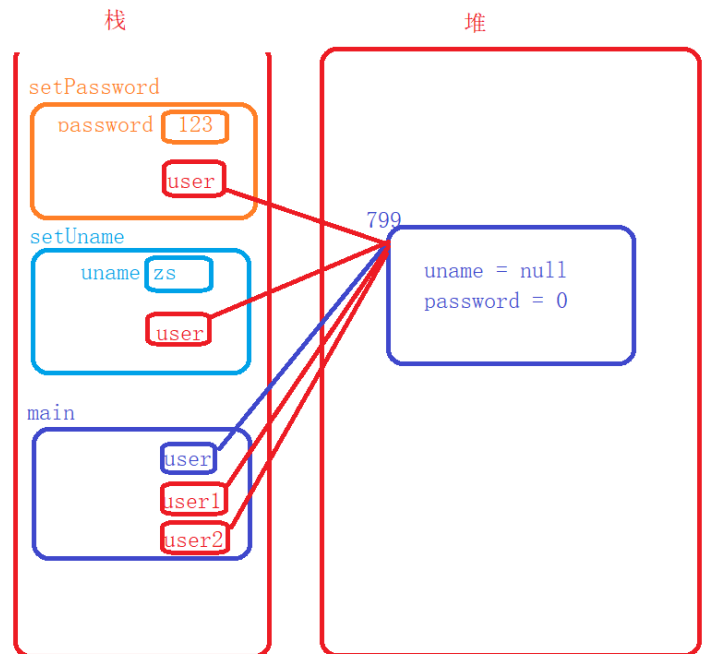
## 10. 链式调用（对象传递方式）

```
class User{
    private String uname;
    private int password;
    public User setName(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.setName("zs", user);
        User user2 = user1.setPassword(123, user);

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



### 代码块

```
1 package com.powernode.private13;
2
3 class User{
4     private String uname;
5     private int password;
```

```
6
7
8     public User setUsername(String uname, User user) {
9         this.uname = uname;
10        return user;
11    }
12
13    public User setPassword(int password, User user) {
14        this.password = password;
15        return user;
16    }
17 }
18
19 public class Test {
20     public static void main(String[] args) {
21         /* User user = new User();
22         user.setUsername("zs");
23         user.setPassword(123);*/
24         /* User user = new User();
25         User user1 = user.setUsername("zs", user);
26         User user2 = user1.setPassword(123, user);
27
28         System.out.println(user == user1);
29         System.out.println(user1 == user2);*/
30         User user = new User();
31         user.setUsername("zs", user).setPassword(123, user);
32
33     }
34 }
```