

第八章 类的继承

1. 没有继承存在的问题

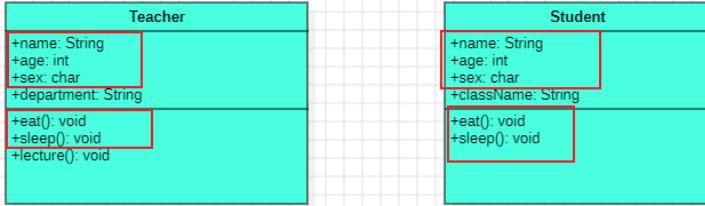
代码块

```
1 package com.powernode.extends09;
2
3 public class Teacher {
4     public String name;
5     public int age;
6     public char sex;
7     public String department;
8
9     public void eat(){
10         System.out.println("Teacher.eat");
11     }
12     public void sleep(){
13         System.out.println("Teacher.sleep");
14     }
15     public void lecture(){
16         System.out.println("Teacher.lecture");
17     }
18 }
19
20
```

代码块

```
1 package com.powernode.extends09;
2
3 public class Student {
4     public String name;
5     public int age;
6     public char sex;
7     public String className;
8
9     public void eat(){
10         System.out.println("Student.eat");
11     }
12     public void sleep(){
13         System.out.println("Student.sleep");
14     }
15 }
```

```
14      }
15  }
```



重复定义的成员

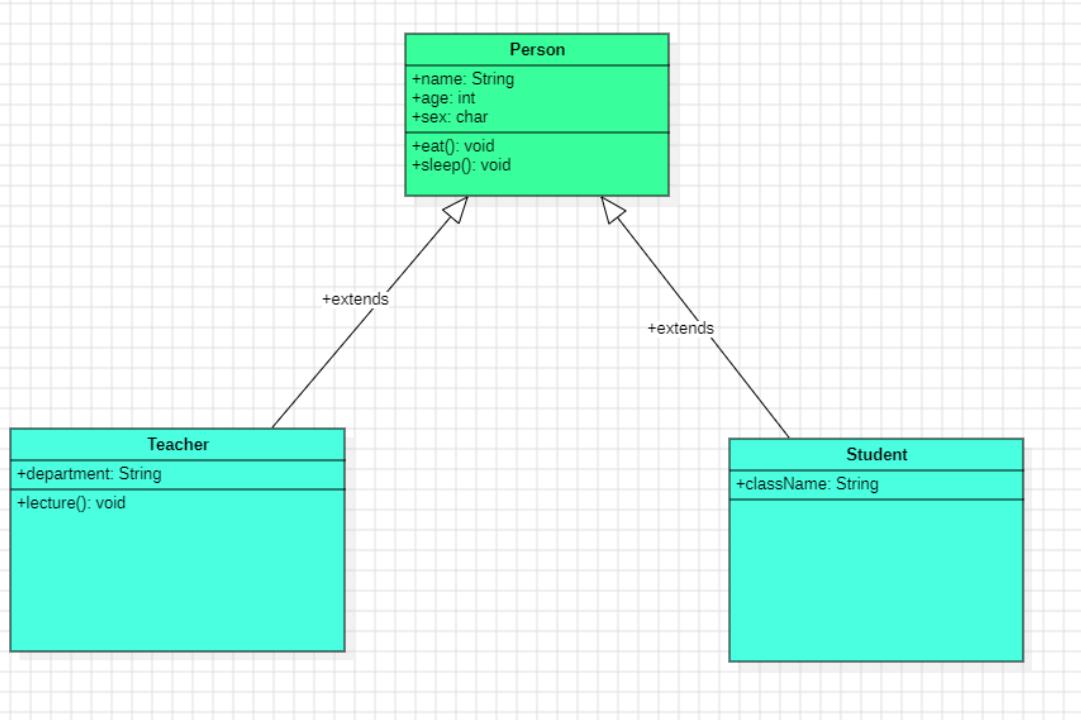
1. 属性
2. 方法

1. Teacher类和Student类的属性和方法，有重复定义
2. 可以使用继承来解决

2. 继承的概述

1. 继承的关键字是：extends
 - a. 父类（超类）
 - b. 子类
2. 继承的语法：
 - a. 子类 extends 父类
3. 继承的作用
 - a. 减少代码的重复编写
 - b. 继承可以使程序结构更加清晰，方便后期维护

3. 继承解决代码冗余



代码块

```

1 package com.powernode.extends10;
2
3 public class Person {
4     public String name;
5     public int age;
6     public char sex;
7
8     public void eat(){
9         System.out.println("Teacher.eat");
10    }
11    public void sleep(){
12        System.out.println("Teacher.sleep");
13    }
14 }
15 //Teacher 继承了 Person, 那么Teacher就拥有了Person的公有属性和方法
16 class Teacher extends Person{
17     public String department;
18
19     public void lecture(){
20         System.out.println("Teacher.lecture");
21     }
22 }
23 class Student extends Person{
24
25     public String className;
26
27
28 }

```

4. 使用继承下来的公有成员

代码块

```
1 package com.powernode.extends11;
2
3 public class Person {
4     public String name = "zs";
5     public int age = 23;
6     public char sex = '男';
7
8     public void eat(){
9         System.out.println("Teacher.eat");
10    }
11    public void sleep(){
12        System.out.println("Teacher.sleep");
13    }
14 }
15 //Teacher 继承了 Person, 那么Teacher就拥有了Person的公有属性和方法
16 class Teacher extends Person{
17     public String department = "讲师编号";
18
19     public void lecture(){
20         System.out.println("Teacher.lecture");
21     }
22 }
23 class Student extends Person{
24
25     public String className = "班级名称";
26 }
27 class Test{
28     public static void main(String[] args) {
29         //1. 使用子类继承下来的属性
30         Teacher t1 = new Teacher();
31         System.out.println(t1.name);
32         System.out.println(t1.age);
33         System.out.println(t1.sex);
34         //2. 使用子类继承下来的方法
35         t1.sleep();
36         t1.eat();
37         //3. 访问自己的属性和方法
38         System.out.println(t1.department);
39         t1.lecture();
40     }
41 }
```

5. 继承中的私有成员

- <https://docs.oracle.com/javase/tutorial/java/landl/subclasses.html>
- 官网文档解释私有成员的继承问题

代码块

```

1  Private Members in a Superclass
2  A subclass does not inherit the private members of its parent class. However,
3  if the superclass has public or protected methods for accessing its private
4  fields,
5  these can also be used by the subclass.
6  A nested class has access to all the private members of its enclosing
7  class—both fields and methods. Therefore, a public or protected nested class
8  inherited
9
10 超类中的私有成员
11 子类不会继承父类的私有成员。但是，如果超类拥有用于访问其私有字段的公共或受保护方法，那么子
12 类也可以使用这些方法。
13 嵌套类可以访问其外围类的所有私有成员（包括字段和方法）。因此，子类继承的公共或受保护嵌套类
14 可以间接访问超类的所有私有成员。
15

```

代码块

```

1  package com.powernode.extends12;
2
3  public class Person {
4      public String name = "zs";
5      //私有成员：只能本类访问
6      private int age = 23;
7
8      public int getAge() {
9          return age;
10     }
11
12     public void setAge(int age) {
13         this.age = age;

```

```
14     }
15 }
16 class Teacher extends Person{
17
18 }
19 class Test{
20     public static void main(String[] args) {
21         Teacher t1 = new Teacher();
22         System.out.println(t1.name);
23         //System.out.println(t1.age);
24         t1.setAge(33);
25         System.out.println(t1.getAge());
26     }
27 }
```

6. 方法重写（方法覆盖）

代码块

```
1 package com.powernode.extends13;
2
3 public class Person {
4     public void sleep(){
5         System.out.println("Person.躺着睡");
6     }
7 }
8
9 /**
10 * 1.Teacher 继承了 Person, Teacher就拥有了sleep方法
11 * 2.但是午休办公室没有床, 老师午休趴着睡
12 * 3.子类继承父类的方法, 不能满足子类的需求
13 * 4.这种情况, 我们可以私用【方法重写】来实现子类个性化的需求
14 *
15 */
16 class Teacher extends Person{
17     public void sleep(){
18         System.out.println("趴着睡");
19     }
20 }
21 class Student extends Person{
22
23 }
24 class Test{
25     public static void main(String[] args) {
```

```

26     Teacher teacher = new Teacher();
27     /**
28      * 1.方法访问的规则:
29      *   1.子类如果有, 访问子类的
30      *   2.子类如果没有, 访问父类的
31      *   3.父类也没有, 报错
32      */
33     teacher.sleep();
34
35     Student student = new Student();
36     student.sleep();
37
38 }
39 }
```

7. Override的作用和重写的规则

代码块

```

1 package com.powernode.extends14;
2
3 public class Person {
4     public void sleep(){
5         System.out.println("躺着睡");
6     }
7 }
8 class Teacher extends Person{
9     /**
10      * 1.@Override的作用
11      *   1.约束了该方法, 必须是重写父类的
12      *   2.具体作用:
13      *     1.避免重写写错
14      *     2.提高可读性
15      * 2.重写的规则 (会背)
16      *   1.方法重写, 两个子类必须有父子关系
17      *   2.返回类型, 方法名称和参数列表和父类一样
18      *   3.子类的访问权限 >= 父类的访问权限
19      *     访问权限的从大到小: public > protected > default(不写) > private
20      *   4.private 修饰的方法不可以被重写
21      *   5.static 修饰的方法不可以被重写
22      *   6.异常: 子类 <= 父类 (异常专题讲解)
23      *   7.在实际的工作中: 通常情况下: 把父类的方法重写一份, 只改方法体
24      *
25      */
26 @Override
27 public void sleep() {
```

```
28         System.out.println("趴着睡");
29     }
30 }
```

8. 子类属性和父类同名

代码块

```
1 package com.powernode.extends15;
2
3 public class Person {
4     public String name = "父类name";
5 }
6 class Teacher extends Person{
7     public String name = "子类name";
8 }
9 class Test{
10     public static void main(String[] args) {
11         Teacher teacher = new Teacher();
12         System.out.println(teacher.name);
13     }
14 }
```

9. super调用无参构造器

代码块

```
1 package com.powernode.extends16;
2
3 public class Person {
4     public String name = "父类name";
5     public Person(){
6         System.out.println("Person.Person");
7     }
8 }
9 class Teacher extends Person{
10     public String name = "子类name";
11     /**
12      * 1.Teacher 没有任何构造器
13      * 2.编译器会在编译时添加默认构造器
14      * 3.默认构造器的第一行会【默认添加super()】
15      * 4.super(), 调用父类无参构造器
16      * 5.调用'super()'必须是构造函数体中的第一条语句
17     */
18 }
```

```

18     Teacher(){
19         //System.out.println("Teacher.Teacher");
20         super(); //调用'super()'必须是构造函数体中的第一条语句
21     }
22 }
23 class Test{
24     public static void main(String[] args) {
25         /**
26             * 创建子类对象时，先创建父类对象
27             */
28         Teacher teacher = new Teacher();
29     }
30 }

```

10. this和super调用构造器

* 3. 构造器的第一行如果没有this调用了重载构造器
 * 4. 那么编译器会添加super(),

```

/*
 * 3. 构造器的第一行如果没有this调用了重载构造器
 * 4. 那么编译器会添加super(),
 */
Teacher(){
    this(age: 20);
}
Teacher(int age){
    //super();
    this.age = age;
}

class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
    }
}

```

```

public class Person { 1 usage 1 inheritor
    public Person(){
        System.out.println("Person.Person");
    }
}

```

代码块

```

1 package com.powernode.extends17;
2
3 public class Person {
4     public Person(){
5         System.out.println("Person.Person");
6     }
7 }
8 class Teacher extends Person{
9     int age;
10
11     /**

```

```
12     * 1.构造器的第一行如果有this调用重载构造器
13     * 2.那么编译器不会添加super(),调用父类构造器
14     * 3.构造器的第一行如果没有this调用了重载构造器
15     * 4.那么编译器会添加super(),调用父类构造器
16     */
17 Teacher(){
18     this(20);
19 }
20 Teacher(int age){
21     //super();
22     this.age = age;
23 }
24 }
25 class Test{
26     public static void main(String[] args) {
27         Teacher teacher = new Teacher();
28     }
29 }
```

11. super调用父类有参构造器

代码块

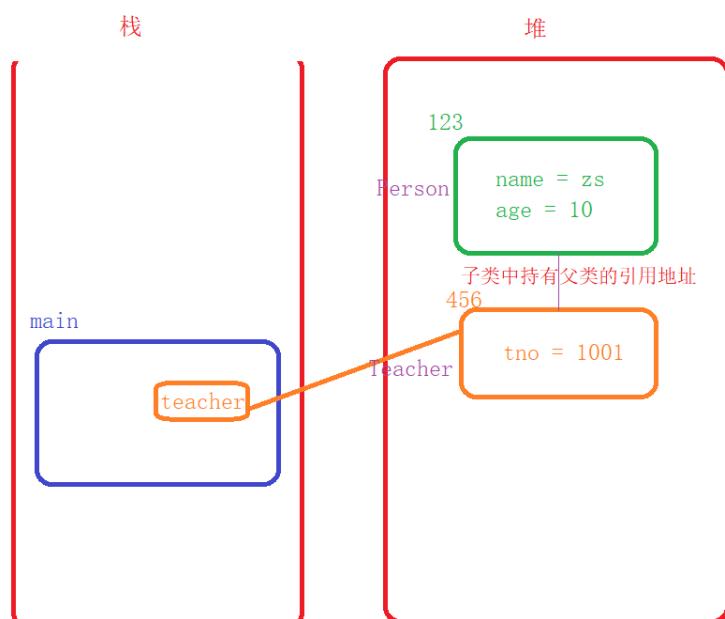
```
1 package com.powernode.extends18;
2
3 public class Person {
4     private String name;
5     private int age;
6
7     public Person(String name, int age) {
8         this.name = name;
9         this.age = age;
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public int getAge() {
17        return age;
18    }
19 }
20 class Teacher extends Person{
21     private int tno;
```

```

22     /*Teacher(){
23         super();
24     }*/
25     //可以理解为有三个属性，父类的两个要借助子类构造器初始化
26     Teacher(String name,int age,int tno){
27         super(name,age);
28         this.tno = tno;
29     }
30
31     public int getTno() {
32         return tno;
33     }
34 }
35 class Test{
36     public static void main(String[] args) {
37         Teacher teacher = new Teacher("zs",23,1001);
38         System.out.println(teacher.getName());
39         System.out.println(teacher.getAge());
40         System.out.println(teacher.getTno());
41     }
42 }
```

```

public class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
class Teacher extends Person{
    private int tno;
    Teacher(String name, int age, int tno) {
        super(name, age);
        this.tno = tno;
    }
}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher("zs", 23, 1001);
    }
}
```



代码块

```

1 package com.powernodeextends19;
2
3 public class Person {
4     private String name;
```

```

5     private int age;
6
7     public Person(String name, int age) {
8         this.name = name;
9         this.age = age;
10    }
11
12 }
13 class Teacher extends Person{
14     private int tno;
15
16     Teacher(String name,int age,int tno){
17         super(name,age);
18         this.tno = tno;
19     }
20
21 }
22 }
23 class Test{
24     public static void main(String[] args) {
25         Teacher teacher = new Teacher("zs", 23, 1001);
26     }
27 }
```

12. 子类持有父类引用

- 每个实例方法中都持有
 - this代表当前对象
 - super代表父类对象

代码块

```

1 package com.powernode.extends20;
2
3 public class Person {
4     private String name;
5     private int age;
6
7     public Person(String name, int age) {
8         this.name = name;
9         this.age = age;
10    }
11
12     public void m1(){
13         System.out.println("Person.m1");
```

```
14     }
15     public String getDetails(){
16         return "姓名: " + name + "\t年龄: " + age;
17     }
18 }
19 class Teacher extends Person{
20     private int tno;
21
22     Teacher(String name,int age,int tno){
23         super(name,age);
24         this.tno = tno;
25     }
26
27     @Override
28     public void m1() {
29         //super.m1();父类的m1方法
30         //this.m1();//子类的m1方法
31         System.out.println("Teacher.m1");
32     }
33
34     @Override
35     public String getDetails() {
36         return super.getDetails() + "\t编号: " + tno;
37     }
38 }
39 class Test{
40     public static void main(String[] args) {
41         Teacher teacher = new Teacher("zs", 23, 1001);
42         teacher.m1();
43         System.out.println(teacher.getDetails());
44     }
45 }
```