第八章 类的继承

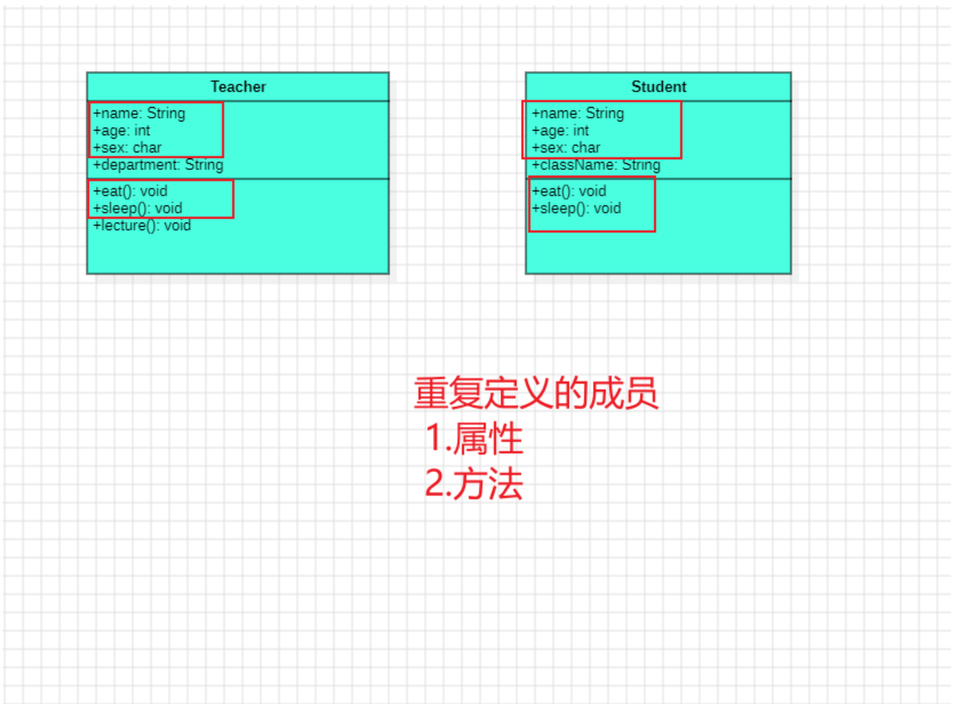
1. 没有继承存在的问题

代码块

```java
package com.powernode.extends09;

public class Teacher {
    public String name;
    public int age;
    public char sex;
    public String department;

    public void eat(){
        System.out.println("Teacher.eat");
    }
    public void sleep(){
        System.out.println("Teacher.sleep");
    }
    public void lecture(){
        System.out.println("Teacher.lecture");
    }
}


```

代码块

```java
package com.powernode.extends09;

public class Student {
    public String name;
    public  int age;
    public char sex;
    public String className;

    public void eat(){
        System.out.println("Student.eat");
    }
    public void sleep(){
        System.out.println("Student.sleep");
        }
    }
```

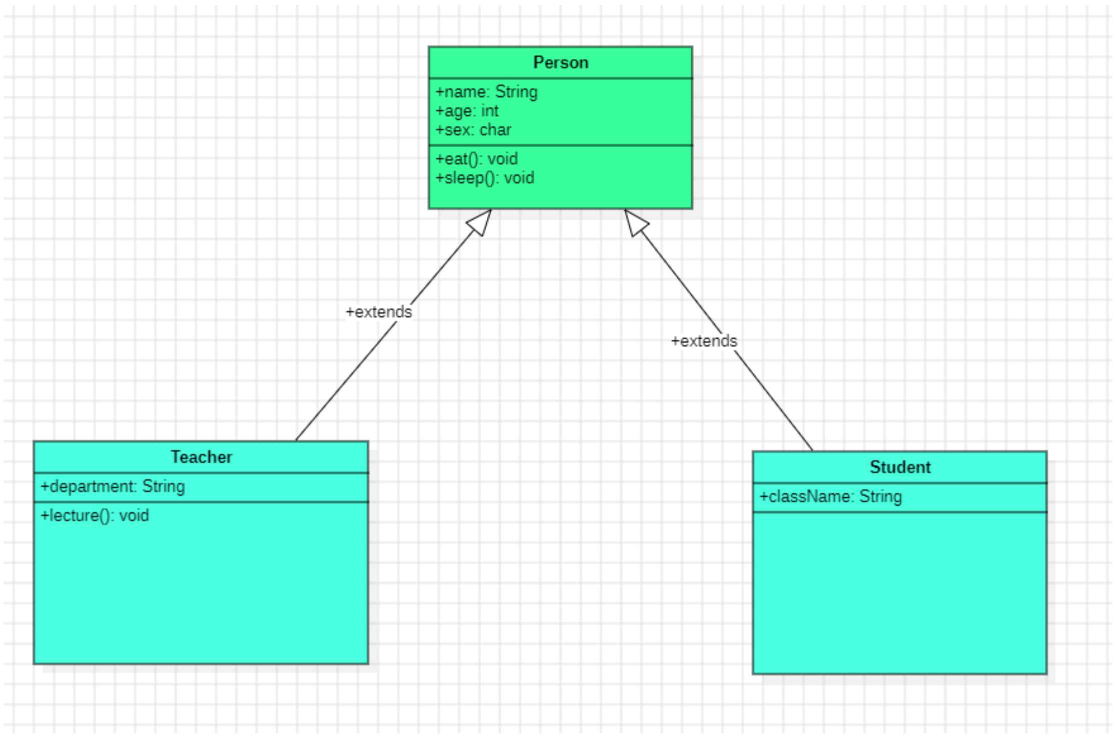


1. Teacher类和Student类的属性和方法，有重复定义
2. 可以使用继承来解决

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

```java
package com.powernode.extends10;

public class Person {
    public String name;
    public int age;
    public char sex;

    public void eat(){
        System.out.println("Teacher.eat");
    }
    public void sleep(){
        System.out.println("Teacher.sleep");
    }
}
//Teacher 继承了 Person，那么Teacher就拥有了Person的公有属性和方法
class Teacher extends  Person{
    public String department;

    public void lecture(){
        System.out.println("Teacher.lecture");
    }
}
class Student extends Person{

    public String className;


}
```

4. 使用继承下来的公有成员

代码块

```java
package com.powernode.extends11;

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

    public void eat(){
        System.out.println("Teacher.eat");
    }
    public void sleep(){
        System.out.println("Teacher.sleep");
    }
}
//Teacher 继承了 Person，那么Teacher就拥有了Person的公有属性和方法
class Teacher extends  Person{
    public String department = "讲师编号";

    public void lecture(){
        System.out.println("Teacher.lecture");
    }
}
class Student extends Person{

    public String className = "班级名称";
}
class Test{
    public static void main(String[] args) {
        //1.使用子类继承下来的属性
        Teacher t1 = new Teacher();
        System.out.println(t1.name);
        System.out.println(t1.age);
        System.out.println(t1.sex);
        //2.使用子类继承下来的方法
        t1.sleep();
        t1.eat();
        //3.访问自己的属性和方法
        System.out.println(t1.department);
        t1.lecture();

    }
}
```

5. 继承中的私有成员

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

代码块

```
Private Members in a Superclass
A subclass does not inherit the private members of its parent class. However,
if the superclass has public or protected methods for accessing its private fields,
these can also be used by the subclass.
A nested class has access to all the private members of its enclosing
class—both fields and methods. Therefore, a public or protected nested class inherited
by a subclass has indirect access to all of the private members of the superclass.


超类中的私有成员
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，那么子类也可以使用这些方法。

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可以间接访问超类的所有私有成员。


```

代码块

```java
package com.powernode.extends12;

public class Person {
    public String name = "zs";
    //私有成员：只能本类访问
    private int age = 23;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
class Teacher extends Person{

}
class Test{
    public static void main(String[] args) {
        Teacher t1 = new Teacher();
        System.out.println(t1.name);
        //System.out.println(t1.age);
        t1.setAge(33);
        System.out.println(t1.getAge());
    }
}
```

6. 方法重写（方法覆盖）

代码块

```java
package com.powernode.extends13;

public class Person {
    public void sleep(){
        System.out.println("Person.躺着睡");
    }
}

/**
 * 1.Teacher 继承了 Person，Teacher就拥有了sleep方法
 * 2.但是午休办公室没有床，老师午休趴着睡
 * 3.子类继承父类的方法，不能满足子类的需求
 * 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
 *
 */
class Teacher extends Person{
    public void sleep(){
        System.out.println("趴着睡");
    }
}
class Student extends Person{

}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
        /**
         * 1.方法访问的规则:
         *    1.子类如果有，访问子类的
         *    2.子类如果没有，访问父类的
         *    3.父类也没有，报错
         */
        teacher.sleep();

        Student student = new Student();
        student.sleep();

    }
}
```

7. Override的作用和重写的规则

代码块

```
package com.powernode.extends14;

public class Person {
    public  void sleep(){
        System.out.println("躺着睡");
    }
}
class Teacher extends Person{
    /**
     * 1.@Override的作用
     *    1.约束了该方法，必须是重写父类的
     *    2.具体作用:
     *      1.避免重写写错
     *      2.提高可读性
     * 2.重写的规则（会背）
     *    1.方法重写，两个子类必须有父子关系
     *    2.返回类型，方法名称和参数列表和父类一样
     *    3.子类的访问权限 >= 父类的访问权限
     *        访问权限的从大到小: public > protected > default(不写)  > private
     *    4.private 修饰的方法不可以被重写
     *    5.static 修饰的方法不可以被重写
     *    6.异常: 子类 <= 父类（异常专题讲解）
     *    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
     *
     */
    @Override
    public  void sleep() {
```

```java
            System.out.println("趴着睡");
        }
    }
```

8. 子类属性和父类同名

代码块

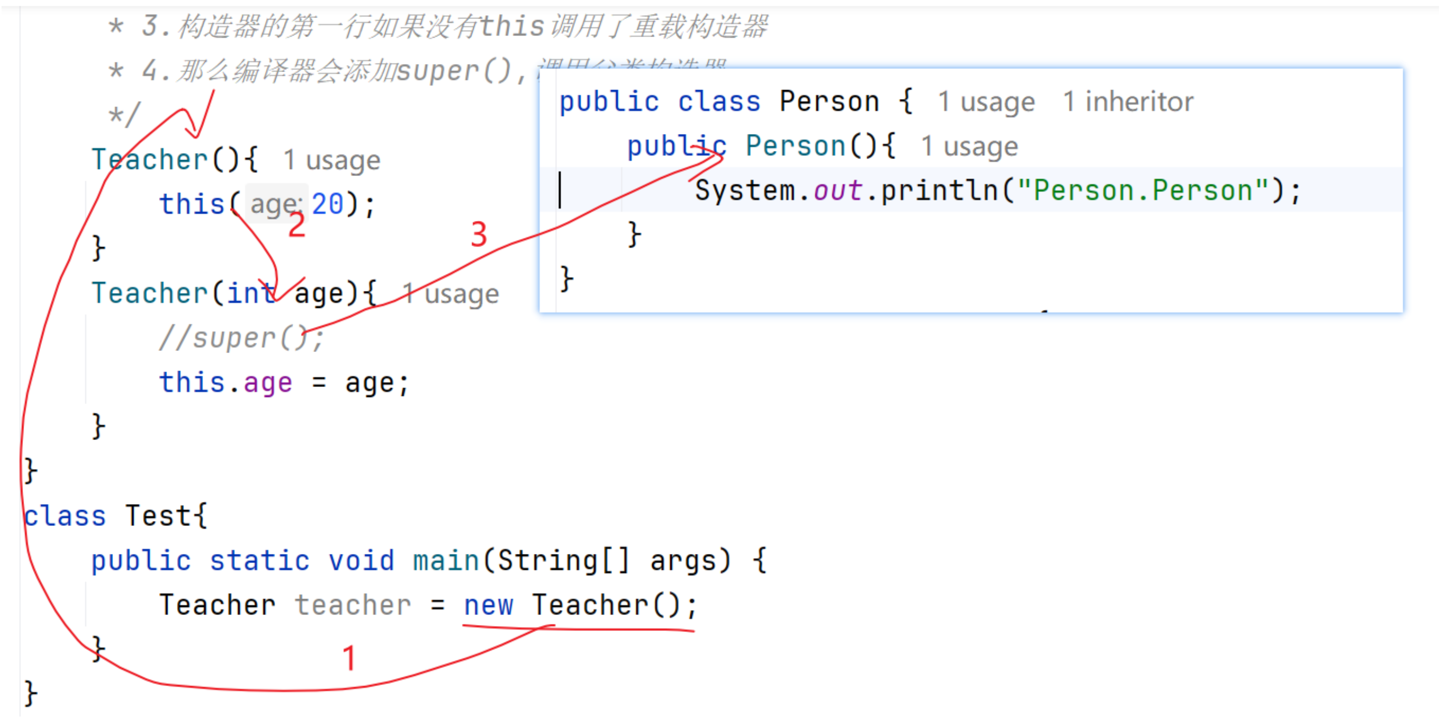
```java
package com.powernode.extends15;

public class Person {
    public String name = "父类name";
}
class Teacher extends  Person{
    public String name = "子类name";
}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
        System.out.println(teacher.name);
    }
}
```

9. super调用无参构造器

代码块

```java
package com.powernode.extends16;

public class Person {
    public String name = "父类name";
    public Person(){
        System.out.println("Person.Person");
    }
}
class Teacher extends  Person{
    public String name = "子类name";
    /**
     * 1.Teacher 没有任何构造器
     * 2.编译器会在编译时添加默认构造器
     * 3.默认构造器的第一行会【默认添加super()】
     * 4.super(),调用父类无参构造器
     * 5.调用‘super()’必须是构造函数体中的第一条语句
     */
    Teacher(){
        //System.out.println("Teacher.Teacher");
        super();//调用‘super()’必须是构造函数体中的第一条语句
    }
}
class Test{
    public static void main(String[] args) {
        /**
         * 创建子类对象时，先创建父类对象
         */
        Teacher teacher = new Teacher();
    }
}
```

10. this和super调用构造器



```java
     * 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(),调用父类构造器
     */
    Teacher(){  1 usage
        this( age: 20);
    }
    Teacher(int age){  1 usage
        //super();
        this.age = age;
    }
}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
    }
}
```

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

代码块

```java
package com.powernode.extends17;

public class Person {
    public Person(){
        System.out.println("Person.Person");
    }
}
class Teacher extends Person{
    int age;

    /**
```

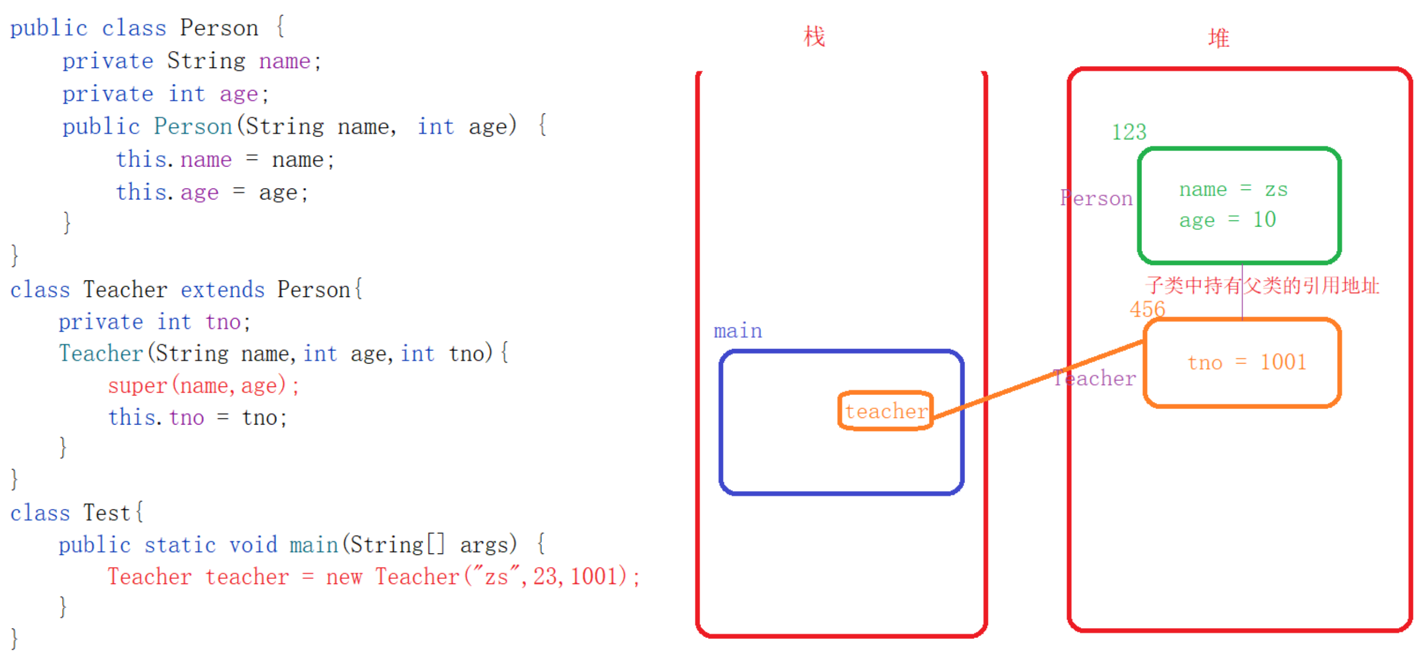
```
         * 1.构造器的第一行如果有this调用重载构造器
         * 2.那么编译器不会添加super(),调用父类构造器
         * 3.构造器的第一行如果没有this调用了重载构造器
         * 4.那么编译器会添加super(),调用父类构造器
         */
        Teacher(){
            this(20);
        }
        Teacher(int age){
            //super();
            this.age = age;
        }
    }
    class Test{
        public static void main(String[] args) {
            Teacher teacher = new Teacher();
        }
    }
```

11. super调用父类有参构造器

代码块

```
package com.powernode.extends18;

public class Person {
    private String name;
    private int age;

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

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
class Teacher extends Person{
    private int tno;
```

```java
    /*Teacher(){
        super();
    }*/
    //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
    Teacher(String name,int age,int tno){
        super(name,age);
        this.tno = tno;
    }

    public int getTno() {
        return tno;
    }
}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher("zs",23,1001);
        System.out.println(teacher.getName());
        System.out.println(teacher.getAge());
        System.out.println(teacher.getTno());
    }
}
```



```java
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

```java
package com.powernode.extends19;

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);
    }
}
```

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

```java
package com.powernode.extends20;

public class Person {
    private String name;
    private int age;

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

   public void m1(){
       System.out.println("Person.m1");
    }
    public String getDetails(){
        return "姓名：" + name + "\t年龄：" + age;
    }
}
class Teacher extends Person{
    private int tno;

    Teacher(String name,int age,int tno){
        super(name,age);
        this.tno = tno;
    }

    @Override
    public void m1() {
        //super.m1();父类的m1方法
        //this.m1();//子类的m1方法
        System.out.println("Teacher.m1");
    }

    @Override
    public String getDetails() {
        return super.getDetails() + "\t编号：" + tno;
    }
}
class Test{
    public static void main(String[] args) {
        Teacher teacher = new Teacher("zs", 23, 1001);
        teacher.m1();
        System.out.println(teacher.getDetails());
    }
}
```