第十章 对象的关联与Object类

1. 对象的关联

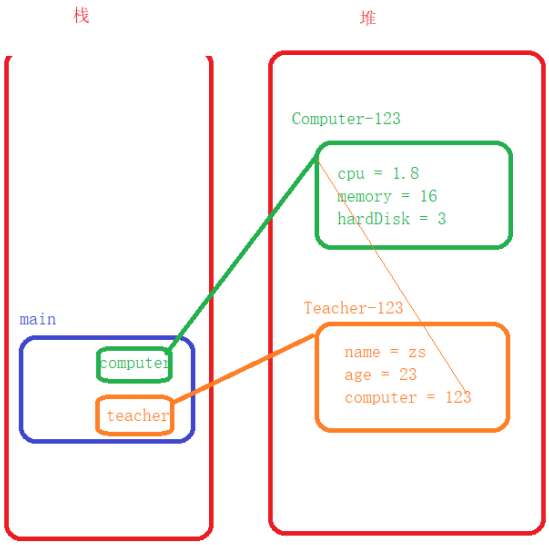
- 继承：父类和子类的关系，继承的关联关系比较高，耦合性比较高
- 关联：耦合性比较低
  - 一个类的对象，是另一个类的属性
  - 比如：老师有姓名，年龄，还有电脑

代码块

```
package com.powernode.relevancy11;

class Computer{
    private double cpu;
    private int memory;
    private int hardDisk;

    public Computer(double cpu, int memory, int hardDisk) {
        this.cpu = cpu;
        this.memory = memory;
        this.hardDisk = hardDisk;
    }

    public String getDetails(){
        return "cpu:" + cpu + "\t内存: " + memory + "\t硬盘: " + hardDisk;
    }
}
class Teacher {
    private String name;
    private int age;
    //老师有电脑
    private Computer computer;//基本类型的变量 叫变量 ，引用类型的变量 叫对象
    public Teacher(String name,int age,Computer computer){
        this.name = name;
        this.age = age;
        this.computer = computer;
    }
    public String getDetails(){
        return "姓名: " + name + "\t年龄: " + age + "\t电脑配置: " +
computer.getDetails();
    }
```

```
31  }
32  public class Test {
33      public static void main(String[] args) {
34          //1.创建电脑对象
35          Computer computer = new Computer(1.8,16,3);
36          //2.创建老师对象
37          Teacher teacher = new Teacher("zs",23,computer);
38          System.out.println(teacher.getDetails());
39      }
40  }
```

```
class Computer{
    private double cpu;
    private int memory;
    private int hardDisk;
    public Computer(double cpu, int memory, int hardDisk) {
        this.cpu = cpu;
        this.memory = memory;
        this.hardDisk = hardDisk;
    }
    public String getDetails(){
        return "cpu:" + cpu + "\t内存：" + memory + "\t硬盘：" + hardDisk;
    }
}
class Teacher {
    private String name;
    private int age;
    private Computer computer;//基本类型的变量 叫变量 ， 引用类型的变量 叫对象
    public Teacher(String name,int age,Computer computer){
        this.name = name;
        this.age = age;
        this.computer = computer;
    }
    public String getDetails(){
        return "姓名：" + name + "\t年龄：" + age + "\t电脑配置：" + computer.getDetails();
    }
}
public class Test {
    public static void main(String[] args) {
        //1.创建电脑对象
        Computer computer = new Computer(1.8,16,3);
        //2.创建老师对象
        Teacher teacher = new Teacher("zs",23,computer);
        System.out.println(teacher.getDetails());
```



2. Object类的概述

代码块

```
1  package com.powernode.object12;
2
3  public class Test {
4      //说明继承
5      @Override
6      public String toString() {
7          return super.toString();
8      }
9  }
```

- @Override 说明有父类
- Object类是所有类的父类
- 每个类都默认继承了Object

代码块

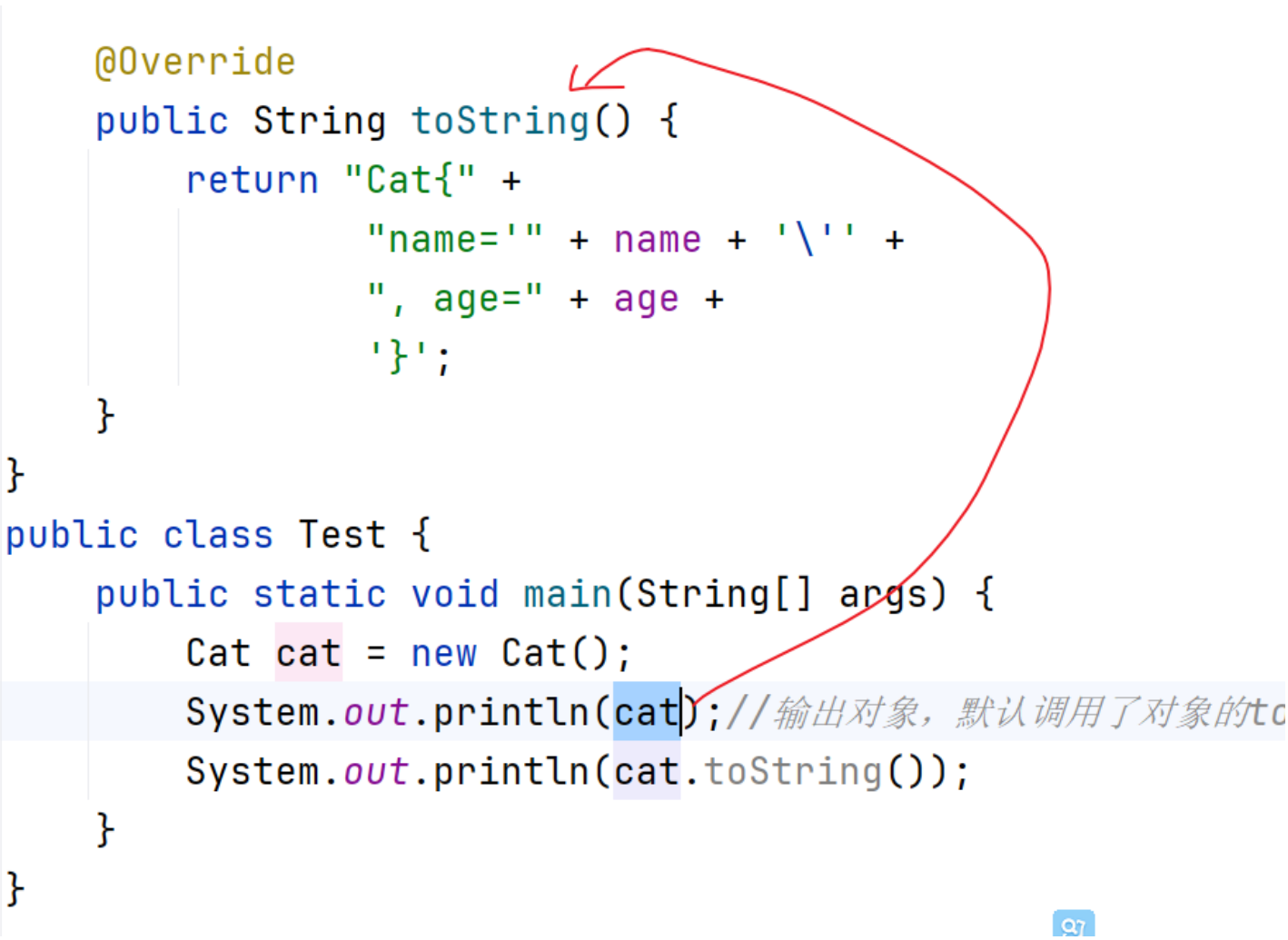
```
public class Test {
    //说明继承
    @Override
    public String toString() {
        return super.toString();
    }
}

等同于

public class Test extends Object {
    //说明继承
    @Override
    public String toString() {
        return super.toString();
    }
}
```

3. toString方法

代码块

```
package com.powernode.object13;

class Cat extends Object{
    private String name = "喵喵";
    private int age = 2;

    @Override
    public String toString() {
        return "Cat{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
public class Test {
```

```
16      public static void main(String[] args) {
17          Cat cat = new Cat();
18          System.out.println(cat);//输出对象，默认调用了对象的toString方法
19          System.out.println(cat.toString());
20      }
21  }
```



4. clone方法(现阶段不要求掌握)

代码块

```
1   package com.powernode.object14;
2
3   class Cat implements Cloneable{
4       private String name = "喵喵";
5       private int age = 2;
6
7       @Override
8       public String toString() {
9           return "Cat{" +
```

```
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}

public class Test {
    public static void main(String[] args) throws CloneNotSupportedException {
        Cat cat = new Cat();
        System.out.println(cat);
        Cat newCat = (Cat) cat.clone();
        System.out.println(newCat);

    }

}
```

5. hashCode

代码块

```
package com.powernode.object15;

import java.util.Objects;

class Cat {
    private String name;
    private int age;

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

    @Override
    public int hashCode() {
        //根据name和age生成hash码，如果两个对象的name和age相等，那么hash码也相等
        return Objects.hash(name,age);
    }
}
public class Test {
```

```
    public static void main(String[] args) {
        Cat cat1 = new Cat("喵喵",3);
        Cat cat2 = new Cat("喵喵",3);
        System.out.println("cat1 = " + cat1.hashCode());
        System.out.println("cat2 = " + cat2.hashCode());
        /**
         * 1.如果两个对象的内容相同，那么hashCode不一定相等
         * 2.如果两个对象的hash码相等，那么那么内容一定相等
         */
        //cat2 = cat1;
        System.out.println(cat1);
        System.out.println(cat2);
        System.out.println(cat1 == cat2);//false
    }
}
```

6. identityHashCode（了解）

代码块

```
package com.powernode.object16;

import java.util.Objects;

class Cat {
    private String name;
    private int age;

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

    @Override
    public int hashCode() {
        //根据name和age生成hash码，如果两个对象的name和age相等，那么hash码也相等
        return Objects.hash(name,age);
    }
}
public class Test {
    public static void main(String[] args) {
        Cat cat1 = new Cat("喵喵",3);
        Cat cat2 = new Cat("喵喵",3);
        System.out.println(cat1.hashCode());
        System.out.println(cat2.hashCode());

```

```
        //身份hashCode
        System.out.println(System.identityHashCode(cat1));
        System.out.println(System.identityHashCode(cat2));
        //== 可以理解为比较的是 身份hashCode,重写了hashCode,身份hash不变
        System.out.println(cat1 == cat2);
    }
}
```

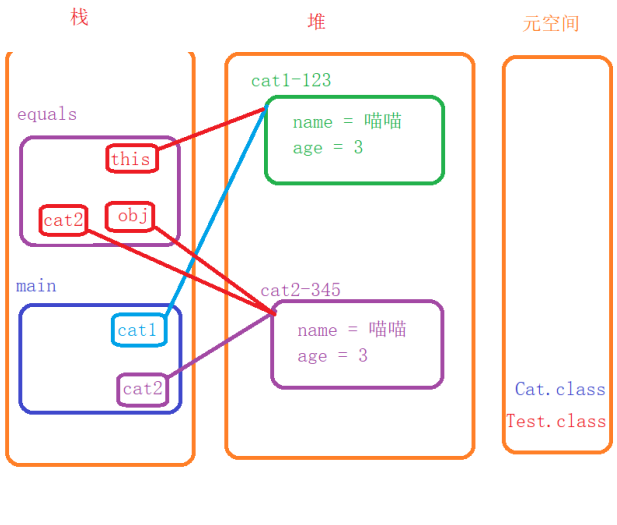
7. equals方法和内存分析

代码块

```
  package com.powernode.object17;

class Cat {
    private String name;
    private int age;

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

    /**
     * 1.比较两个对象的属性值是否相等
     * 2.先找到两个对象
     *    1.this:当前对象，谁调用了equals方法，谁就是当前对象，所以this = cat1
     *    2.obj: Object obj = cat2
     *
     */
    @Override
    public boolean equals(Object obj) {
        //this和obj的地址相等，说明指向堆中的同一个对象，内容一定相等
        if (this == obj) return true;
        //如果obj为null，那么this不为null，内容不具有比较性，直接返回false
        // this的字节码文件对象 不等于 obj的字节码文件对象，说明就不是一个类，不具有可不行
        if (obj == null || this.getClass() != obj.getClass())  return false;
        Cat cat2 = (Cat) obj;
        return age == cat2.age && name.equals(cat2.name);//字符串比较内容是否相等，使用equals
       // return this.age == cat2.age && this.name.equals(cat2.name);//字符串比较内容是否相等，使用equals
    }
}
class Dog{}
```

```java
public class Test {
    public static void main(String[] args) {
        Cat cat1 =  new Cat("喵喵",3);
        Cat cat2 =  new Cat("喵喵",3);
        //Dog dog = new Dog();
        /**
         * 1.比较两个对象的属性值是否相等
         * 2.使用 == 无法比较
         * 3.使用equals也无法比较，因为Object类的equals方法比较的也是地址
         * 4.父类的方法不满足子类的需求
         * 5.重写equals方法，在equals方法中比较属性值
         *
         */
        System.out.println(cat1.equals(cat2));
        /**
         * public boolean equals(Object obj) {
         *         return (this == obj);
         * }
         */
    }
}
```

```java
class Cat {
    private String name;
    private int age;
    public Cat(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public boolean equals(Object obj) {
        //this和obj的地址相等，说明指向堆中的同一个对象，内容一定相等
        if (this == obj) return true;
        if (obj == null || this.getClass() != obj.getClass())  return false;
        Cat cat2 = (Cat) obj;
        return age == cat2.age && name.equals(cat2.name);
    }
}
public class Test {
    public static void main(String[] args) {
        Cat cat1 =  new Cat("喵喵",3);
        Cat cat2 =  new Cat("喵喵",3);
        System.out.println(cat1.equals(cat2));
    }
}
```



作业

1:对象的关联

1. 编写Computer类，其属性为cpu，memory，hardDisk，提供构造器及相关方法，以及say方法用于自我描述。
2. 编写Student类，Computer对象作为Student类的属性，提供构造器及相关方法，以及say方法用于自我描述。

3. 编写Test类，在main方法中创建Student对象，调用say方法打印输出结果。

2.toString

1.编写Student类，定义属性name、age和address

2.覆盖Object的toString方法用于自我描述

3.提供构造器为实例变量赋值。

4.在Test类的main方法中，创建 Student 对象，并打印该对象。

5.再打印调用toString方法的返回内容，比较两个输出结果是否相同。为什么?

3.equals方法

1.定义Dog类，定义属性name、age

2.编写Test类，在main方法中创建两个 Dog对象

3.比较两个对象是否equals （Object类的）

4.重写equals比较对象的内容是否相等

4.hashCode方法

1.在Dog类重写hashCode改变父类的hash规则，使用name和age属性生成hash值