第一章 类和对象

主要内容

- 类和对象的关系 **重点**
- 类图 **熟悉**
- 定义和调用方法 **重点**
- 成员变量和成员方法 **重点** **难点**
- this 关键字 **重点** **难点**
- 定义和调用构造方法 **重点**

课程目标

- 掌握类和对象的关系
- 熟悉类图的使用
- 掌握方法的定义及调用
- 掌握成员变量和局部变量的区别
- 掌握构造方法的定义及调用

第一节 类和对象

1. 类的由来

人们在日常生活中，经常会将具有相同特征或者相同行为的事物归为一类。在Java中，用来描述这类事物的就是Java类，Java类就是这样诞生的。Java 是一门以类为组织单元的语言，我们定义的Java类就是一种Java数据类型，该数据类型属于引用数据类型。

2. 如何定义类

语法

```
public class 类名{

}
```

示例

```
//人类
public class Person {

}
```

类定义好了，如何填充类中的内容?

找出人类共同的特征：姓名，性别，年龄

```java
public class Person {

    public String name; //姓名

    public String sex; //性别

    public int age; //年龄
}

```

找出人类共同的行为：吃饭、睡觉、工作

在类中如何描述人类的行为?

在Java中，使用方法来描述行为，方法的定义语法如下：

```
//[]中内容表示可有可无
访问修饰符  返回值类型  方法名([参数列表]){
    [return 返回值;]
}
```

那么Person类的定义应该如下：

```java
public class Person {

    public String name; //姓名

    public String sex; //性别

    public int age; //年龄

    public void eat(){
        System.out.println("人吃饭");
    }

    public void sleep(){
        System.out.println("人睡觉");
    }

    public void work(){
        System.out.println("人工作");
    }
}
```

定义一个计算器类，计算器能够对两个数字进行加减乘除。

分析

a. 计算器能够接收两个数字和一个运算符

b.计算器能够进行计算

```java
public class Calculator {

    public double number1; //接收的数字1
```

```
    public double number2; //接收的数字2

    public String operator; //接收的运算符

    public void calculate(){//计算
        switch (operator){
            case "+":
                System.out.println(number1 + number2);
                break;
            case "-":
                System.out.println(number1 - number2);
                break;
            case "*":
                System.out.println(number1 * number2);
                break;
            case "/":
                System.out.println(number1 / number2);
                break;
        }
    }
}
```

3. 类图

类图用于描述类的结构，与流程图一样，简单直观，容易理解

public修饰的属性和方法前需要使用'+'，private修饰的属性和方法前需要使用'-'

可以使用processon网站进行绘制类图

4. 类和对象的关系

解释说明

类是描述多个事物的共有特征和行为的一个抽象体。而对象是一个具体的事物，每一个属性和每一个行为都是具体的。类是对象的集合体。类是用来构建具体的对象的。

语法

```
类名 对象名 = new 类名();
对象名.属性名 = 属性值;
```

示例

```
public class PersonTest {

    public static void main(String[] args) {
        //这里p称为对象名，跟数组名一样，本质都是变量。只是在面向对象中称之为对象名
        Person p = new Person();//构建了一个具体的人，只是这个人目前还没有名字，性别和年龄
        p.name = "刘德华";
        p.sex = "男";
        p.age = 53;
    }
}

public class CalculatorTest {
```

```
13
14      public static void main(String[] args) {
15          Calculator c = new Calculator(); //构建了一个具体的计算器
16          c.number1 = 10;
17          c.number2 = 5;
18          c.operator = "*";
19      }
20  }
```

类既然是一类事物的共同特征和行为的描述，那么一个类应该可以描述多个事物，因此类也可以创建多个对象。

示例

```
1   public class PersonTest {
2
3       public static void main(String[] args) {
4           //这里p称为对象名，跟数组名一样，本质都是变量。只是在面向对象中称之为对象名
5           Person p = new Person();//构建了一个具体的人，只是这个人目前还没有名字，性别
    和年龄
6           p.name = "刘德华";
7           p.sex = "男";
8           p.age = 53;
9
10          Person p1 = new Person();
11          p1.name = "张学友";
12          p1.sex = "男";
13          p1.age = 52;
14
15          Person p2 = new Person();
16          p2.name = "黎明";
17          p2.sex = "男";
18          p2.age = 45;
19      }
20  }
```

结论

类是对多个事物的抽象描述，描述的是他们的共同特征和行为举止。但需要注意的是：**类中描述的共同特征，在对象创建出来之后是跟随对象走的。行为举止也是一样，属于对象。**

练习

请使用类和对象的相关知识描述汽车类和宠物类，并构建具体的对象。

分析：

汽车的特征： 品牌 型号 价格

汽车的行为：启动 加速 刹车

```
1   public class Car {
2
3       public String brand; //品牌
4
5       public String type; //型号
6
7       public double price; //价格
```

```
8
9       public void start(){
10          System.out.println("汽车启动");
11      }
12
13      public void speedUp(){
14          System.out.println("汽车加速");
15      }
16
17      public void stop(){
18          System.out.println("汽车刹车");
19      }
20  }
21  public class CarTest {
22
23      public static void main(String[] args) {
24          Car c = new Car();
25          c.brand = "奥迪";
26          c.type = "A8";
27          c.price = 100000;
28      }
29  }
```

宠物的特征：名字 健康值

宠物的行为：玩耍

```
1   public class Pet {
2
3       public String name; //宠物名字
4
5       public int health; //宠物的健康值
6
7       public void play(){
8           System.out.println("宠物在玩耍");
9       }
10  }
11  public class PetTest {
12
13      public static void main(String[] args) {
14          Pet p = new Pet();
15          p.name = "哈巴狗";
16          p.health = 100;
17      }
18  }
```

第二节 成员变量和成员方法

1. 成员变量

解释说明

在类中定义的变量就是成员变量。成员变量顾名思义是属于成员（具体的对象、具体的事物）的，成员变量有初始值。

成员变量的初始值：

引用数据类型的初始值都是 null ，整数都是0，浮点数都是0.0，boolean类型是false，char类型是'\u0000'

访问成员变量的语法

```
对象名.属性名;
```

示例

```
public class PersonTest {

    public static void main(String[] args) {
        //这里p称为对象名，跟数组名一样，本质都是变量。只是在面向对象中称之为对象名
        Person p = new Person();//构建了一个具体的人，只是这个人目前还没有名字，性别和年龄
        System.out.println(p.name + "\t" + p.sex + "\t" + p.age);
        p.name = "刘德华";
        p.sex = "男";
        p.age = 53;

        System.out.println(p.name + "\t" + p.sex + "\t" + p.age);

        Person p1 = new Person();
        p1.name = "张学友";
        p1.sex = "男";
        p1.age = 52;
        System.out.println(p1.name + "\t" + p1.sex + "\t" + p1.age);

        Person p2 = new Person();
        p2.name = "黎明";
        p2.sex = "男";
        p2.age = 45;
        System.out.println(p2.name + "\t" + p2.sex + "\t" + p2.age);
    }
}
```

练习

利用对象的属性展示汽车类和宠物类的信息

```
public class CarTest {

    public static void main(String[] args) {
        Car c = new Car();
        c.brand = "奥迪";
        c.type = "A8";
        c.price = 100000;
        System.out.println(c.brand + "\t" + c.type + "\t" + c.price);
    }
}

public class PetTest {

    public static void main(String[] args) {
        Pet p = new Pet();
        p.name = "哈巴狗";
```

```
        p.health = 100;
        System.out.println(p.name + "\t" + p.health);
    }
}
```

2. 成员方法

解释说明

在类中定义的方法就是成员方法。成员方法顾名思义是属于成员（具体的对象、具体的事物）的。

调用成员方法的语法

```
//[]中内容可有可无
对象名.方法名([参数列表]);
```

示例

```
/**
 * 人类
 */
public class Person {

    public String name; //姓名

    public String sex; //性别

    public int age; //年龄

    public void eat(){
        System.out.println(age + "岁的" + sex + "性同志" +name + "在吃饭");
    }

    public void sleep(){
        System.out.println(age + "岁的" + sex + "性同志" +name + "在睡觉");
    }

    public void work(){
        System.out.println(age + "岁的" + sex + "性同志" +name + "在工作");
    }
}

public class PersonTest {

    public static void main(String[] args) {
        //这里p称为对象名，跟数组名一样，本质都是变量。只是在面向对象中称之为对象名
        Person p = new Person();//构建了一个具体的人，只是这个人目前还没有名字，性别和年龄
        System.out.println(p.name + "\t" + p.sex + "\t" + p.age);
        p.name = "刘德华";
        p.sex = "男";
        p.age = 53;
        System.out.println(p.name + "\t" + p.sex + "\t" + p.age);
        p.eat();
        p.sleep();
```

```
        p.work();

        Person p1 = new Person();
        p1.name = "张学友";
        p1.sex = "男";
        p1.age = 52;
        System.out.println(p1.name + "\t" + p1.sex + "\t" + p1.age);
        p1.eat();
        p1.sleep();
        p1.work();


        Person p2 = new Person();
        p2.name = "黎明";
        p2.sex = "男";
        p2.age = 45;
        System.out.println(p2.name + "\t" + p2.sex + "\t" + p2.age);
        p2.eat();
        p2.sleep();
        p2.work();
    }
}
```

练习

利用对象的方法展示汽车类和宠物类的信息

```
public class Car {

    public String brand; //品牌

    public String type; //型号

    public double price; //价格

    public void start(){
        System.out.println("汽车启动");
    }

    public void speedUp(){
        System.out.println("汽车加速");
    }

    public void stop(){
        System.out.println("汽车刹车");
    }

    public void show(){//展示
        System.out.println(brand + "\t" + type + "\t" + price);
    }
}

public class CarTest {

    public static void main(String[] args) {
        Car c = new Car();
        c.brand = "奥迪";
```

```
        c.type = "A8";
        c.price = 100000;
        c.show();


        Car c1 = new Car();
        c1.brand = "大众";
        c1.type = "保时捷卡宴";
        c1.price = 150000;
        c1.show();
    }
}
```

3. 成员变量和局部变量

解释说明

局部变量就是在方法内部定义的变量。局部变量没有初始值，因此，局部变量在使用之前必须完成初始化操作。当局部变量与成员变量同名时，局部变量的优先级更高。

示例

```
public void show(){//展示
    String name;//局部变量，没有初始值
    System.out.println(name + brand + "\t" + type + "\t" + price);
}

public void show(){//展示
    //因此局部变量的作用范围更小，就在局部变量所定义的方法内，
    //因此局部变量在方法内的优先级要高于成员变量
    String brand = "奔驰";
    System.out.println(brand + "\t" + type + "\t" + price);
}
```

4. this 关键字

思考

在方法中，如果局部变量和成员变量同名，此时又想使用成员变量，怎么办呢?

此时需要使用this关键字来解决。

this关键字表示的是当前对象（使用new创建出来的对象）

示例

```
public class Car {

    public String brand; //品牌

    public String type; //型号

    public double price; //价格

    public void start(){
        System.out.println("汽车启动");
    }
```

```
12
13      public void speedUp(){
14          System.out.println("汽车加速");
15      }
16
17      public void stop(){
18          System.out.println("汽车刹车");
19      }
20
21      public void show(){//展示
22          //因此局部变量的作用范围更小，就在局部变量所定义的方法内，
23          //因此局部变量在方法内的优先级要高于成员变量
24          String brand = "奔驰";
25          System.out.println(this.brand + "\t" + type + "\t" + price);
26      }
27  }
28
29  public class CarTest {
30
31      public static void main(String[] args) {
32          Car c = new Car(); // this => c
33          c.brand = "奥迪";
34          c.type = "A8";
35          c.price = 100000;
36          c.show();
37
38
39          Car c1 = new Car(); // this => c1
40          c1.brand = "大众";
41          c1.type = "保时捷卡宴";
42          c1.price = 150000;
43          c1.show();
44      }
45  }
```

this还可以用来调用成员的方法

```
1   /**
2    * 人类
3    */
4   public class Person {
5
6       public String name; //姓名
7
8       public String sex; //性别
9
10      public int age; //年龄
11
12      public void eat(){
13          System.out.println(age + "岁的" + sex + "性同志" +name + "在吃饭");
14          work(); //相当于 this.work();
15      }
16
17      public void sleep(){
18          System.out.println(age + "岁的" + sex + "性同志" +name + "在睡觉");
19          work(); //相当于 this.work();
20      }
```

```
    public void work(){
        System.out.println(age + "岁的" + sex + "性同志" +name + "在工作");
    }
}

```

第三节 构造方法

1. 概念

构造方法是一种特殊的方法，主要用于创建对象以及完成对象的属性初始化操作。构造方法不能被对象调用。

2. 语法

```
//[]中内容可有可无
访问修饰符 类名([参数列表]){

}
```

3. 示例

```
public class Car {

    public String brand; //品牌

    public String type; //型号

    public double price; //价格


    public Car(){
        this.brand = "默认品牌";
        this.type = "默认型号";
        this.price = 5000;
    }

    public void start(){
        System.out.println("汽车启动");
    }

    public void speedUp(){
        System.out.println("汽车加速");
    }

    public void stop(){
        System.out.println("汽车刹车");
    }

    public void show(){//展示
        //因此局部变量的作用范围更小，就在局部变量所定义的方法内，
        //因此局部变量在方法内的优先级要高于成员变量
        String brand = "奔驰";
        System.out.println(this.brand + "\t" + type + "\t" + price);
```

```
    }
}

```

思考：在Car类中没有定义构造方法的时候，我们也可以这么使用，为什么？

```
public class Car {

    public String brand; //品牌

    public String type; //型号

    public double price; //价格

    public Car(){

    }

    public void start(){
        System.out.println("汽车启动");
    }

    public void speedUp(){
        System.out.println("汽车加速");
    }

    public void stop(){
        System.out.println("汽车刹车");
    }

    public void show(){//展示
        //因此局部变量的作用范围更小，就在局部变量所定义的方法内，
        //因此局部变量在方法内的优先级要高于成员变量
        String brand = "奔驰";
        System.out.println(this.brand + "\t" + type + "\t" + price);
    }
}

```

来自官方的说明

[类和对象]

```
You don't have to provide any constructors for your class, but you must be careful when doing this. The compiler automatically provides a no-argument, default constructor for any class without constructors.

你不必为类提供任何构造方法，但是在执行此操作时必须要小心。 编译器会自动为任何没有构造方法的类提供无参数的默认构造方法。
```