7.1 类与对象

7.1.1 养猫猫问题引入

张老太养了两只猫猫：一只名字叫小白，今年 3 岁，毛色为白色；另一只叫小花，今年 100 岁，毛色为花色。我们希望编写一个程序，当用户输入小猫的名字时，能显示该猫的名字、年龄和颜色；若用户输入的小猫名错误，则显示 “张老太没有这只猫猫” 。这个问题看似简单，却能很好地引出面向对象编程的相关知识。通过解决这个问题，我们可以逐步理解如何更高效、合理地组织和管理程序中的数据与操作。

7.1.2 使用现有技术解决（Object01.java 示例）

1. 单独定义变量解决

我们可以为每只猫的信息分别定义变量，如：

```
// 第 1 只猫信息
String cat1Name = "小白";
int cat1Age = 3;
String cat1Color = "白色";
// 第 2 只猫信息
String cat2Name = "小花";
int cat2Age = 100;
String cat2Color = "花色";
```

这样做虽然能实现记录猫的信息，但将一只猫的信息拆解开来，不利于数据的整体管理。当猫的数量增多或者需要对猫的信息进行更多操作时，代码会变得混乱且难以维护。

2. 使用数组解决

也可以使用数组来存储猫的信息，例如：

```
// 第 1 只猫信息
String[] cat1 = {"小白", "3", "白色"};
String[] cat2 = {"小花", "100", "花色"};
```

然而，这种方式存在一些缺点。首先，从数组中无法直接体现出数据的类型，比如年龄应该是整数类型，但在数组中只是字符串形式；其次，只能通过数组下标来获取信息，变量名字和内容的对应关系不明确，代码可读性差；最后，数组不能体现猫的行为，比如猫的跑动、进食等行为无法通过数组来表示。

7.1.3 现有技术解决的缺点分析

- **不利于数据管理**：单独定义变量时，数据分散，难以对同一类事物（如猫）的信息进行统一管理和操作。例如，要统计所有猫的平均年龄，需要分别获取每只猫的年龄变量，代码繁琐。
- **效率低**：当需要对大量猫的信息进行处理时，如查找特定条件的猫，单独变量和数组的方式都需要逐个遍历和判断，效率低下。而且数组在处理复杂逻辑时，代码的可读性和可维护性差，进一步影响开发和维护效率。

正是因为现有的这些技术不能完美地解决新的需求，Java 设计者引入了类与对象（OOP，Object - Oriented Programming）的概念。

7.1.4 使用面向对象的方式解决养猫问题

首先，我们定义一个猫类 `Cat` ，它是一个自定义的数据类型，代码如下：

```
// 定义一个猫类 Cat -> 自定义的数据类型
class Cat {
    // 属性/成员变量
    String name; // 名字
    int age; // 年龄
    String color; // 颜色
    // 这里也可以添加更多属性，比如体重
    // double weight;
}
```

然后在 `main` 方法中，使用面向对象的方式创建猫对象并操作：

```
public class Object01 {
    // 编写一个 main 方法
    public static void main(String[] args) {
        // 实例化一只猫[创建一只猫对象]
        // 老韩解读
        // 1. new Cat() 创建一只猫(猫对象)
        // 2. Cat cat1 = new Cat(); 把创建的猫赋给 cat1
        // 3. cat1 就是一个对象
        Cat cat1 = new Cat();
        cat1.name = "小白";
        cat1.age = 3;
        cat1.color = "白色";
        // 创建了第二只猫，并赋给 cat2
        // cat2 也是一个对象(猫对象)
        Cat cat2 = new Cat();
        cat2.name = "小花";
        cat2.age = 100;
        cat2.color = "花色";
        // 怎么访问对象的属性呢
        System.out.println("第 1 只猫信息" + cat1.name
            + " " + cat1.age + " " + cat1.color);
        System.out.println("第 2 只猫信息" + cat2.name
            + " " + cat2.age + " " + cat2.color);
    }
}
```

7.1.5 一个程序就是一个世界，有很多事物（对象 [属性，行为]）

在现实世界中，存在各种各样的事物，每个事物都有自己的属性和行为。比如猫有名字、年龄、颜色等属性，也有跑动、进食、睡觉等行为。在程序中，我们可以将这些事物抽象为对象，对象的属性用变量来表示，对象的行为用方法来表示（在后续章节会详细介绍方法相关内容）。通过这种方式，程序可以更贴近现实世界的逻辑，使代码更易于理解和维护。

7.1.6 类与对象的关系示意图

类是对一类事物的抽象描述，它定义了这类事物共有的属性和行为。例如，“猫类” 定义了所有猫都可能具有的名字、年龄、颜色等属性，以及可能的行为。对象则是类的具体实例，是实实在在存在的个体。比如前面创建的 `cat1` 和 `cat2` 就是 “猫类” 的两个具体对象，它们具有 “猫类” 所定义的属性，但属性值是具体的（如 `cat1` 的名字是 “小白” ）。可以用以下示意图来表示类与对象的关系：
（此处可插入一个简单的类与对象关系示意图，如：画一个矩形框代表类，里面写上类的属性和行为；从类框引出两个小椭圆，分别代表两个对象，在对象椭圆中标注具体的属性值 ）

7.1.7 快速入门 - 面向对象的方式解决养猫问题（回顾 Object01.java 代码）

通过前面的代码示例，我们可以看到使用面向对象编程解决养猫问题的基本步骤：

1. 定义类：定义 `Cat` 类，确定类的属性（如 `name`、`age`、`color` ）。
2. 创建对象：使用 `new` 关键字创建 `Cat` 类的对象，如 `Cat cat1 = new Cat();` 。
3. 操作对象：通过对象名。属性名的方式对对象的属性进行赋值和访问，如 `cat1.name = "小白";` 和 `System.out.println(cat1.name);` 。

7.1.8 类和对象的区别和联系

- 区别
    - 类是抽象的，概念的，代表一类事物，比如人类、猫类等，它是一种数据类型，就像一个模板，定义了这类事物的属性和行为规范，但本身并不对应具体存在的实体。
    - 对象是具体的，实际的，代表一个具体事物，是类的一个实例。它具有类所定义的属性，并且属性有具体的值，是实实在在存在于程序运行过程中的个体。
- **联系**
  类是对象的模板，对象是类的具体体现。对象是根据类来创建的，一个类可以创建多个对象，这些对象都具有类所定义的属性和行为，但属性值可以不同。例如，从 “猫类” 可以创建出无数只具体的猫对象，每只猫对象的名字、年龄、颜色等属性值可能不一样。

7.1.9 对象在内存中存在形式（重要）

要深入理解面向对象编程，了解对象在内存中的存在形式至关重要。在 Java 中，内存主要分为以下几个区域：

1. **栈**：一般存放基本数据类型（如 `int`、`double` 等）以及局部变量。栈的特点是数据的存储和读取速度快，遵循先进后出的原则。
2. **堆**：存放对象（如 `Cat` 类创建的对象）以及数组等。堆中的数据存储空间较大，且可以动态分配。对象在堆中存储时，会为对象的每个属性分配相应的存储空间，并进行初始化。
3. **方法区**：包含常量池（存放常量，比如字符串常量）以及类加载信息。类加载信息包括类的结构信息（如属性、方法等），在程序运行过程中，类的信息只会被加载一次到方法区。

以创建 `Person` 类对象为例，`Person p = new Person();` 的内存分配流程如下：

1. 先加载 `Person` 类信息（属性和方法信息，只会加载一次）到方法区。
2. 在堆中分配空间，进行默认初始化（根据数据类型的默认值规则，如 `int` 类型属性默认值为 0，`String` 类型属性默认值为 `null` 等 ）。
3. 把堆中对象的地址赋给栈中的变量 `p`，此时 `p` 就指向了堆中的对象。
4. 进行指定初始化，比如 `p.name = "jack"; p.age = 10;` ，修改对象属性的默认值为指定的值。

7.1.10 属性 / 成员变量 / 字段

基本介绍

从概念或叫法上看，成员变量 = 属性 = field（字段），即成员变量是用来表示对象属性的，在本教材授课中，统一称为属性。例如，在 `Car` 类中：

```
class Car {
    String name; // 属性，成员变量，字段 field
    double price;
    String color;
    String[] master; // 属性可以是基本数据类型，也可以是引用类型(对象，数组)
}
```

属性是类的一个组成部分，其类型一般可以是基本数据类型（如 `int`、`double`、`boolean` 等），也可以是引用类型（如对象、数组）。比如前面定义猫类中的 `int age` 就是属性。

注意事项和细节说明

1. **属性的定义语法**：访问修饰符 属性类型 属性名；。这里简单介绍访问修饰符，它用于控制属性的访问范围，有四种访问修饰符：`public`（公共的，可被任何类访问）、`protected`（受保护的，在同一个包内或子类中可访问）、默认（不写修饰符时，在同一个包内可访问）、`private`（私有的，只能在本类中访问），后面会详细介绍。
2. **属性的定义类型**：属性的定义类型可以为任意类型，包含基本类型或引用类型。例如，既可以定义 `int` 类型的属性表示数量，也可以定义 `String` 类型的属性表示名称，还可以定义自定义类的对象作为属性（如 `Car` 类中可以有一个 `Driver` 类对象作为属性，表示司机）。
3. **属性的默认值**：属性如果不赋值，有默认值，规则和数组一致。具体来说：`int` 类型默认值为 0，`short` 类型默认值为 0，`byte` 类型默认值为 0，`long` 类型默认值为 0，`float` 类型默认值为 0.0，`double` 类型默认值为 0.0，`char` 类型默认值为 `\u0000`，`boolean` 类型默认值为 `false`，`String` 类型默认值为 `null`。以下是一个案例演示：

```
public class PropertiesDetail {
    // 编写一个 main 方法
    public static void main(String[] args) {
        // 创建 Person 对象
        // p1 是对象名(对象引用)
        // new Person() 创建的对象空间(数据) 才是真正的对象
        Person p1 = new Person();
        // 对象的属性默认值，遵守数组规则:
        // int 0, short 0, byte 0, long 0, float 0.0,double 0.0, char \u0000,
boolean false, String null
        System.out.println("\n 当前这个人的信息");
        System.out.println("age=" + p1.age + " name=" + p1.name + " sal=" +
p1.sal + " isPass=" + p1.isPass);
    }
}
class Person {
    // 四个属性
    int age;
    String name;
    double sal;
    boolean isPass;
}
```

7.1.11 如何创建对象

1. **先声明再创建**：先声明一个对象变量，然后再使用 `new` 关键字创建对象并赋值给该变量。例如：

```
Cat cat; // 声明对象 cat
cat = new Cat(); // 创建
```

这种方式适用于需要在不同时机进行对象声明和创建的场景，比如根据某些条件判断后再创建对象。
\2. **直接创建**：直接使用 `new` 关键字声明并创建对象。例如：

```
Cat cat = new Cat();
```

这种方式更为简洁，适用于大多数情况下直接创建对象并使用的场景。

7.1.12 如何访问属性

基本语法

使用对象名。属性名的方式来访问对象的属性。例如：

```
Cat cat = new Cat();
cat.name; // 访问 cat 对象的 name 属性
cat.age; // 访问 cat 对象的 age 属性
cat.color; // 访问 cat 对象的 color 属性
```

类和对象的内存分配机制（重要）

我们通过一个思考题来深入理解类和对象的内存分配机制。定义一个人类 `Person` （包括名字、年龄），代码如下：

```
public class Object03 {
    public static void main(String[] args) {
        Person p = new Person();
        p.name = "jack";
        p.age = 10;
    }
}
class Person {
    String name;
    int age;
}
```

在这个例子中：

1. 首先，加载 `Person` 类信息（包括属性 `name` 和 `age` 以及可能存在的方法信息）到方法区，这一步在程序运行过程中对于同一个类只会执行一次。
2. 然后，在堆中为 `Person` 对象分配空间，并按照属性类型的默认值规则进行默认初始化，即 `name` 属性初始化为 `null`，`age` 属性初始化为 0 。
3. 接着，在栈中创建变量 `p`，并将堆中 `Person` 对象的地址赋值给 `p`，此时 `p` 就指向了堆中的对象。

4. 最后，通过 `p.name = "jack"; p.age = 10;` 对对象的属性进行指定初始化，修改默认值为我们指定的值。

7.1.13 类和对象的内存分配机制（练习题及内存布局图分析）

练习题：定义一个 `Dog` 类，包含 `name`（字符串类型）、`age`（整数类型）、`weight`（双精度浮点数类型）属性，在 `main` 方法中创建两个 `Dog` 对象 `dog1` 和 `dog2`，分别为它们的属性赋值并输出。请分析并画出内存布局图。

代码示例：

```
public class DogExercise {
    public static void main(String[] args) {
        Dog dog1 = new Dog();
        dog1.name = "大黄";
        dog1.age = 5;
        dog1.weight = 15.5;

        Dog dog2 = new Dog();
        dog2.name = "小黑";
        dog2.age = 3;
        dog2.weight = 10.0;

        System.out.println("dog1 的信息：name=" + dog1.name + ", age=" + dog1.age
+ ", weight=" + dog1.weight);
        System.out.println("dog2 的信息：name=" + dog2.name + ", age=" + dog2.age
+ ", weight=" + dog2.weight);
    }
}
class Dog {
    String name;
    int age;
    double weight;
}
```

内存布局图分析：

1. 程序运行时，首先加载 `Dog` 类信息到方法区，包括 `name`、`age`、`weight` 属性的定义以及类可能包含的方法信息（此处未定义方法，后续会学习方法相关内容）。
2. 当执行 `Dog dog1 = new Dog();` 时，在堆中为 `dog1` 对象分配空间，对属性进行默认初始化（`name` 为 `null`，`age` 为 0，`weight` 为 0.0），然后在栈中创建变量 `dog1`，并将堆中 `dog1` 对象的地址赋值给它。接着通过 `dog1.name = "大黄"; dog1.age = 5; dog1.weight = 15.5;` 对 `dog1` 对象的属性进行指定初始化。
3. 当执行 `Dog dog2 = new Dog();` 时，同样在堆中为 `dog2` 对象分配空间并默认初始化，在栈中创建变量 `dog2` 并赋值地址，再进行指定初始化。
   （此处可插入一个内存布局图，展示栈、堆、方法区的位置，在方法区标注 `Dog` 类信息，在堆中画出两个 `Dog` 对象并标注各自属性值，在栈中画出 `dog1` 和 `dog2` 变量并表示出它们指向堆中对象的关系

课后作业

一、选择题

1. 以下关于类和对象的说法，正确的是（ ）
   A. 类是具体的，对象是抽象的
   B. 对象是类的模板
   C. 类是对象的模板，对象是类的实例
   D. 类和对象没有关系
2. 在 Java 中，对象通常存放在（ ）中
   A. 栈 B. 堆 C. 方法区 D. 寄存器
3. 定义类的属性时，如果不赋值，int 类型属性的默认值是（ ）
   A. null B. 0 C. 0.0 D. false
4. 以下哪种方式是正确创建 `Cat` 类对象的方式（ ）
   A. `Cat cat;` B. `cat = new Cat();`
   C. `Cat cat = new Cat();` D. `new Cat;`
5. 假设定义了 `class Dog { String name; int age; }` ，以下访问 `Dog` 对象属性正确的是（ ）
   A. `Dog dog; dog.name = "旺财";`
   B. `Dog dog = new Dog(); dog.name = "旺财";`
   C. `Dog dog = new Dog; dog.name = "旺财";`
   D. `Dog.name = "旺财";`
6. 类的属性可以是以下哪种类型（ ）
   A. 只能是基本数据类型
   B. 只能是引用类型
   C. 既可以是基本数据类型，也可以是引用类型
   D. 不能是数组类型
7. 以下访问修饰符中，访问权限最大的是（ ）
   A. `private` B. `protected` C. `public` D. 默认
8. 当创建一个对象时，Java 内存分配的流程顺序是（ ）
   ① 在堆中分配空间并默认初始化 ② 加载类信息 ③ 进行指定初始化 ④ 把堆中对象地址赋给栈中变量
   A. ①②③④ B. ②①④③ C. ②①③④ D. ①②④③
9. 关于类和对象，以下说法错误的是（ ）
   A. 一个类可以创建多个对象
   B. 对象的属性值可以不同
   C. 类的属性和行为在方法区中只加载一次
   D. 对象的属性只能在创建对象时赋值
10. 以下对于数组解决养猫问题缺点的描述，错误的是（ ）
    A. 数据类型体现不出来
    B. 可以直观体现猫的行为
    C. 只能通过下标获取信息，变量名和内容对应关系不明确
    D. 不利于数据管理

二、填空题

1. 类是 ____ 的，对象是 ____ 的。
2. 在 Java 中，对象存放在 ____ 中，基本数据类型的局部变量存放在 ____ 中。
3. 定义一个 `Person` 类，包含 `name`（字符串类型）和 `age`（整数类型）属性，定义属性的代码为 `String name; int age;` ，其中 `name` 和 `age` 被称为类的 ____ 。
4. 创建对象的两种方式分别是 ____ 和 ____ 。
5. 访问对象属性的基本语法是 ____ 。
6. 类的属性如果不赋值，`boolean` 类型的默认值是 ____ ，`String` 类型的默认值是 ____ 。

7. 定义类时，属性的定义语法为 ____ 。
8. 面向对象编程中，对象具有 ____ 和 ____ 。
9. 当使用 `new` 关键字创建对象时，首先会加载 ____ 信息。
10. 解决张老太养猫问题，使用传统单独定义变量的方式不利于 ____ 。

三、判断题

1. 类是对象的具体实例，对象是类的模板。（ ）
2. 在 Java 中，所有数据都存放在堆中。（ ）
3. 类的属性如果不赋值，都有默认值。（ ）
4. 先声明对象变量再创建对象和直接创建对象的效果是完全一样的。（ ）
5. 访问对象属性时，只能在创建对象之后进行。（ ）
6. 类的属性只能是基本数据类型，不能是引用类型。（ ）
7. 方法区主要存放对象实例。（ ）
8. 一个类只能创建一个对象。（ ）
9. 用数组解决养猫问题时，能很好地体现数据类型。（ ）
10. 面向对象编程能更方便地管理和操作数据。（ ）

四、简答题

1. 简述类和对象的区别与联系。
2. 说明 Java 中创建对象的步骤和内存分配机制。
3. 请列举出使用数组解决养猫问题存在的缺点。
4. 解释类的属性的定义语法以及属性默认值的规则。
5. 为什么 Java 设计者要引入类与对象的概念？
6. 阐述面向对象编程相对于传统编程方式（如单独定义变量、使用数组）的优势。
7. 访问对象属性的基本语法是什么？请举例说明。
8. 类的属性可以有哪些类型？请分别举例。
9. 描述在定义类时，访问修饰符的作用和种类。
10. 以养猫问题为例，说明如何用面向对象编程解决实际问题。

五、编程题

1. 定义一个 `Book` 类，包含 `name`（书名，字符串类型）、`price`（价格，双精度浮点数类型）、`pageNum`（页数，整数类型）属性，在 `main` 方法中创建一个 `Book` 对象，并为其属性赋值后输出。
2. 定义一个 `Student` 类，包含 `name`（姓名）、`age`（年龄）、`score`（成绩）属性，创建两个 `Student` 对象，分别为他们赋值并比较谁的成绩更高，输出成绩较高的学生姓名。
3. 定义一个 `Circle` 类，包含 `radius`（半径，双精度浮点数类型）属性，编写方法计算圆的面积并在 `main` 方法中创建 `Circle` 对象，计算并输出其面积。（提示：圆面积公式为 $S=\pi**r2$ ，π 取值 3.14 ）
4. 定义一个 `Animal` 类，包含 `name`（名字）、`type`（种类）属性，在 `main` 方法中创建一个 `Animal` 对象数组，包含 3 个动物对象，并为每个对象赋值后遍历输出。
5. 定义一个 `Car` 类，包含 `brand`（品牌）、`color`（颜色）、`speed`（速度，整数类型）属性，创建一个 `Car` 对象，先设置速度为 0 ，然后将速度增加 50 ，最后输出汽车品牌、颜色和当前速度。
6. 定义一个 `Person` 类，包含 `name`（姓名）、`age`（年龄）、`gender`（性别）属性，编写一个方法 `introduce` 用于输出个人信息（格式为：我叫 [姓名]，今年 [年龄] 岁，性别 [性别] ），在 `main` 方法中创建 `Person` 对象并调用该方法。

7. 定义一个 `Rectangle` 类，包含 `length`（长，双精度浮点数类型）和 `width`（宽，双精度浮点数类型）属性，编写方法计算矩形的周长和面积，在 `main` 方法中创建 `Rectangle` 对象，计算并输出其周长和面积。
8. 定义一个 `Phone` 类，包含 `brand`（品牌）、`price`（价格）、`memory`（内存，字符串类型，如"128GB"）属性，创建两个 `Phone` 对象，比较它们的价格高低并输出价格较高的手机品牌。
9. 定义一个 `Flower` 类，包含 `name`（花名）、`color`（颜色）、`petalNum`（花瓣数量，整数类型）属性，在 `main` 方法中创建一个 `Flower` 对象列表（使用数组或集合，此处建议使用数组），添加 4 种花对象并遍历输出。
10. 定义一个 `Employee` 类，包含 `name`（姓名）、`salary`（工资，双精度浮点数类型）、`department`（部门）属性，编写方法 `raiseSalary` 用于给员工涨薪（涨薪幅度为 10%），在 `main` 方法中创建 `Employee` 对象，调用涨薪方法后输出涨薪后的工资。