Java语言基础

【Day06】

Java 面向对象编程

OOP

面向对象编程

- 万物皆对象。
- 面向对象指以属性和行为的观点去分析现实生活中的事物。
- 面向对象编程指先以面向对象的思想进行分析，然后使用面向对象的编程语言进行表达的过程。
- 面向对象编程是软件产业化发展的需求。
- 理解面向对象的思想精髓(封装、继承、多态)，至少掌握一种编程语言。

Java 类、对象和引用

class object reference

类和对象的概念

- 类是一个概念(名词)抽象的定义。简单说就是分类。
- 类定义了该类型对象的数据结构，称之为 成员变量 ，同时，也定义了一些可以被调用的功能，称之为 成员方法。
- 类是用于构建对象的模板，对象的实质就是内存中一块存储区域，其数据结构由定义它的类来决定。

类和成员变量的定义

- Java语言中，类的成员变量就是属性。属性的定义可以使用如下语法：

```
class 类名 {
    成员变量类型 成员变量名称;
    …… ……
}
```

```
class Point {
    int x;
    int y;
}
```

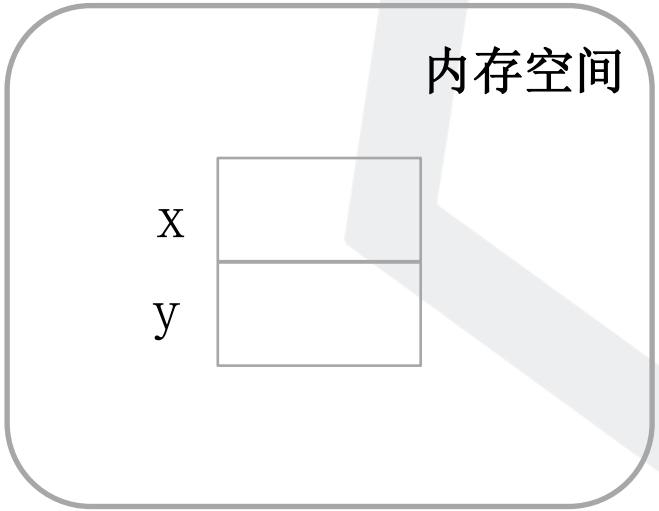
Point类定义了所有对象都应该具有的数据结构。

对象的创建

- 当一个类的定义存在后，可以使用new运算创建该类的对象。对象创建的过程一般称为类的实例化。
- new 类名();

new Point();

这条语句使用new关键字创建了一个Point类的对象，JVM为该对象开辟了内存空间，按照Point的定义，该对象中存储有两个int类型的成员变量。



引用的定义

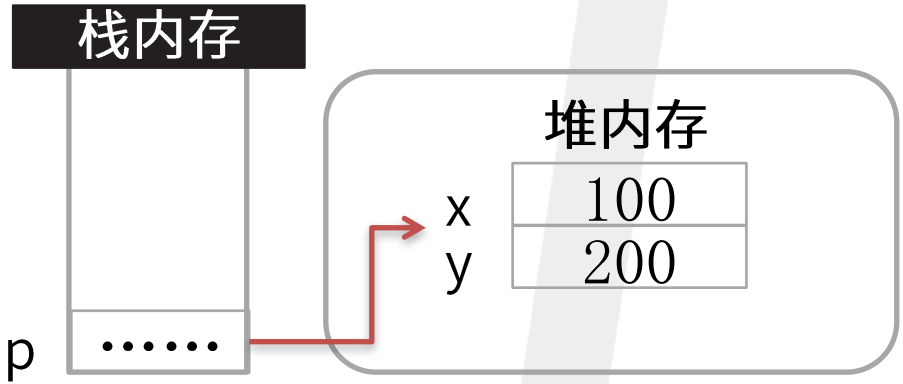
- 除8种基本类型之外，用类名（接口、数组）声明的变量称为引用类型变量，简称“引用”。
- 引用类型变量中存储的是某个对象在内存中的地址信息。
- 引用的功能在于访问对象。引用类型变量按照如下方式声明：
- 类名 引用类型变量名

```
Point p = new Point();
```

用某个类名声明的引用类型变量可以存储该类对象的地址信息，通常称为“指向该类的对象”。当一个引用类型变量指向该类的对象时，就可以通过这个变量对对象实施访问。

项目案例

```
class Point{
    int x;
    int y;
}
Point p = new Point();
p.x = 100;
p.y = 200;
```



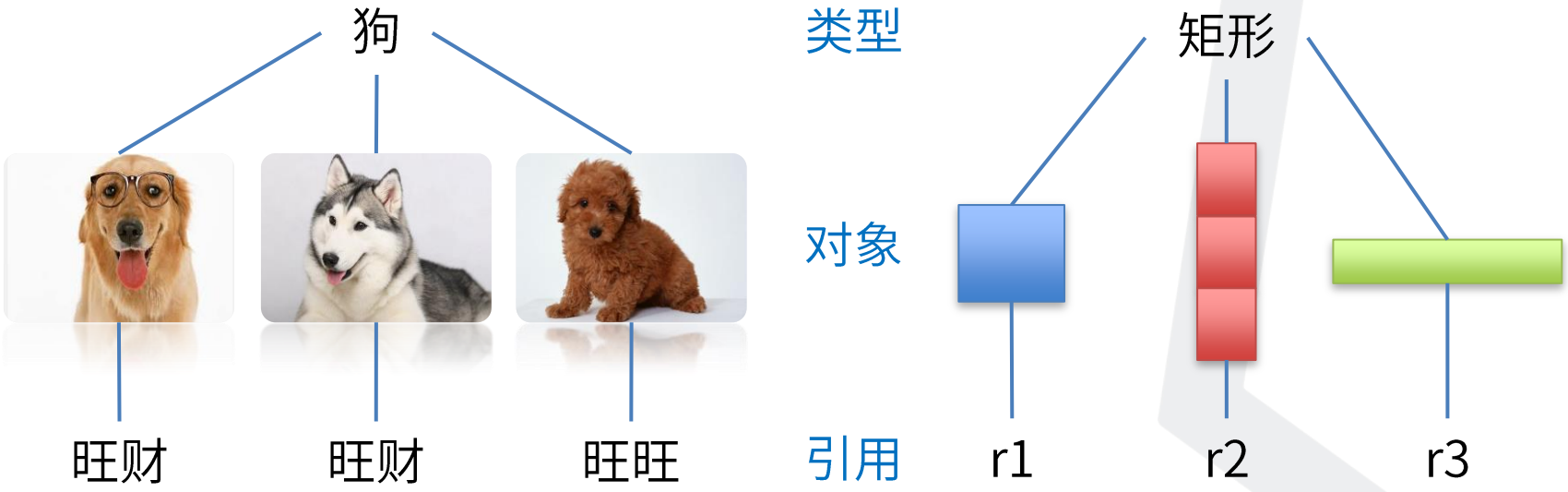
第1条语句等号右边创建了一个Point类型的对象，存放在堆内存中。等号的左边声明了一个名为p的Point类型的变量。存储在“栈”中，赋值表达式将刚刚创建对象的地址信息赋值给p。此后可以通过变量p访问该对象。第2、3条语句体现了通过引用类型变量访问对象的方式；
p.x 表示的语义为“p所指向对象的x”。

成员变量的初始化

- 对象创建后，其成员变量可以按照默认的方式初始化；
- 对象成员变量的默认初始化值规则如下表所示。

成员变量的类型	默认初始值
数值类型（byte、short、int、long、float、double）	0
boolean型	false
char型	\u0000
引用类型	null

类 对象 引用 作用域



Java 成员方法

method

成员方法的定义

- 类中除了定义成员变量，还可以定义方法。
- 方法用于封装一段特定逻辑功能，主要要素有：方法名、参数列表和返回值
- 具体格式如下：

```
class 类名 {
    返回值类型 方法名称(参数列表) {
        …… …… ……
    }
}
```

```
class Point{
    int x; int y;
    void up(int dy) {
        y-=dy;
    }
}
```

方法要素—返回值

- 方法调用结束后可以返回一个数据，称之为返回值。
- 方法在声明时必须指定返回值的数据类型。
- 通过return语句返回，return语句的作用在于结束方法且将数据返回。如果方法没有返回值，则将返回值类型声明为void。

```
return a+b;
```

return语句返回该表达式的值

```
return;
```

对于返回值为void的方法也可以使用return语句，此时该语句的作用仅仅在于结束方法调用。

方法要素—参数列表

- 方法的参数是指：在调用时传递给方法，需要被方法处理的数据。
- 在方法定义时，需要声明该方法所需要的参数变量。
- 在方法调用时，会将实际参数值传递给方法的参数变量。必须保证传递参数的类型和个数符合方法的声明。

```
int max(int a, int b) { …… …… }

int res = max(5,6);
```

方法调用

- 方法的调用必须通过某个对象的引用。当通过某个对象的引用调用方法时，方法中涉及的成员变量就是该对象的成员变量。

```
class Point {
    int x;
    int y;
    public void up(int dy) {
        y -= dy;
    }
}
```

```
Point p1 = new Point();
Point p2 = new Point();
p1.x = 100; p1.y = 200;
p2.x = 300; p2.y = 400;
p1.up(2);
p2.up(2);
```

p1.up()的语义在于：针对p1所指向的对象调用方法up()；
此时，方法up()中所涉及的x和y是p1所指向对象的x和y；
而p2.up()运行时方法up()中所涉及x和y是p2所指向对象的x和y。

方法调用的实例

```
class Point  {
  int x;
  int y;

  int distance(Point p) {
     int dx = x - p.x;
     int dy = y - p.y;
     return (int) Math.sqrt(dx * dx + dy * dy);
  }
}
```

```
Point p1 = new Point();
p1.x = 100; p1.y = 200;
Point p2 = new Point();
p2.x = 400; p2.y = 400;
int res = p1.distance(p2);
System.out.println(dres);   500
```

JVM内存结构-方法区

- 该空间用于存放类的信息。Java程序运行时，首先会通过类装载器载入类文件的字节码信息，经过解析后将其装入方法区。在方法区保存类的各种信息

```
Point p = new Point();
```

Point类首先被装载到JVM的方法区，包括类的基本信息和方法定义等。

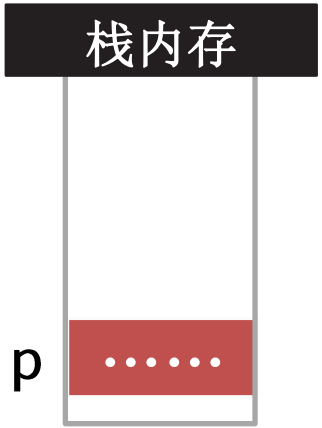
方法区
class Point {…}

JVM内存结构-栈区

- 栈用于存放程序运行过程当中所有的局部变量。一个运行的Java程序从开始到结束会有多次方法的调用。JVM会为每一个方法的调用在栈中分配一个对应的空间，这个空间称为该方法的栈帧。一个栈帧对应一个正在调用中的方法，栈帧中存储了该方法的参数、局部变量等数据。当某一个方法调用完成后，其对应的栈帧将被清除。

```
Point p = new Point();
```

引用类型的变量p在栈中存放。

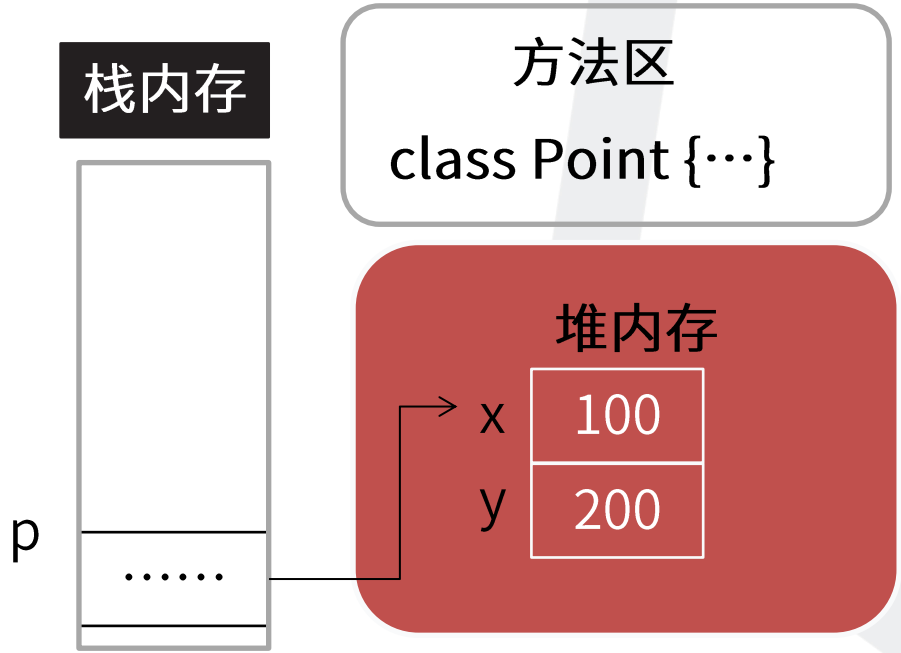


JVM内存结构-堆区

- JVM会在其内存空间中开辟一个称为“堆”的存储空间，这部分空间用于存储使用new关键字创建的对象。

Point p = new Point()

创建了一个Point类型的对象，存放在堆中。



总结与答疑

技术资料

IT兄弟连

ITXDL.CN

变态严管　让学习成为一种习惯