



前端精品课程 开讲啦!

开讲人: _____



IOS



ANDROID



JAVA



.NET



类和对象

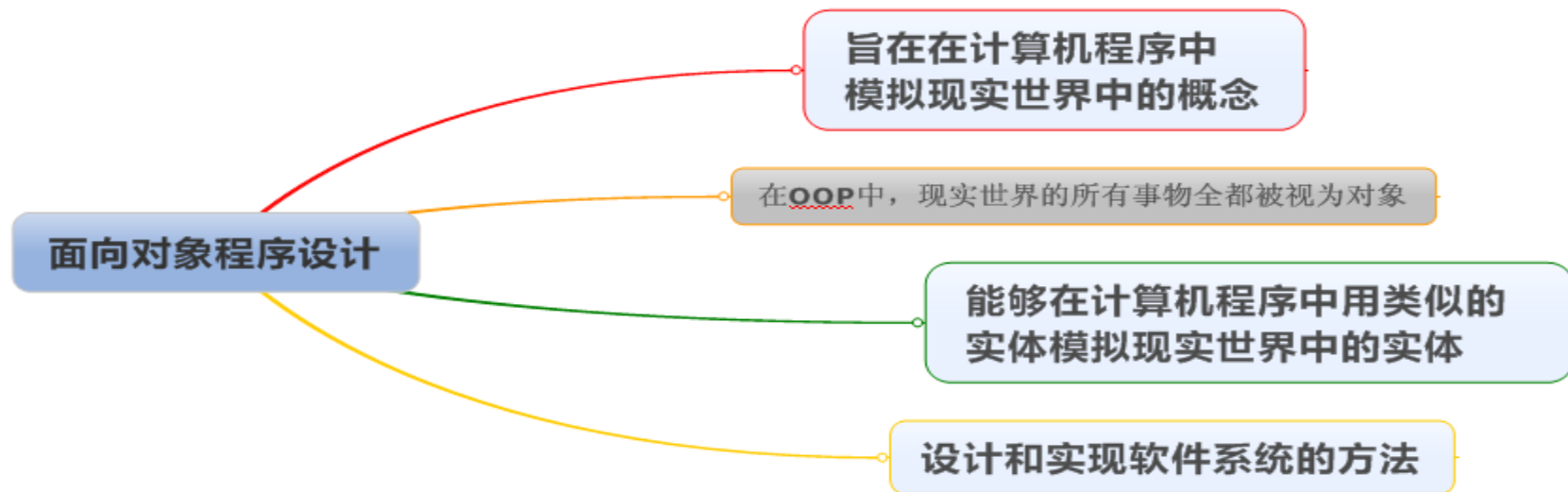
目标

- 理解对象
- 理解类
- 理解抽象和封装
- 创建类，属性和方法
- 掌握包的创建和导入
- 常用类
- 数组
- 算法

面向对象简介

- 面向对象程序设计（Object Oriented Programming, OOP）是当今主流的程序设计思想，如果要编写Java程序，您必须熟悉OOP。
- 它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。
- 面向对象有三个特点：封装、继承、多态。

面向对象程序设计



对象

- 面向对象编程的核心
- 表示现实世界中的实体
- 为计算机应用程序提供实用基础
- 完成特定任务

对象是客观存在的具体实体，具有明确定义的状态和行为。

对象的示例

狗狗 对象



姓名：奥巴
类型：哈士奇
年龄：3个月
体重：6千克

操作：
啃骨头
汪汪叫
到处撒尿

状态

行为

葫芦娃对象

姓名：红娃
年龄：25
体重：52千克

操作：
搬石头
打妖怪



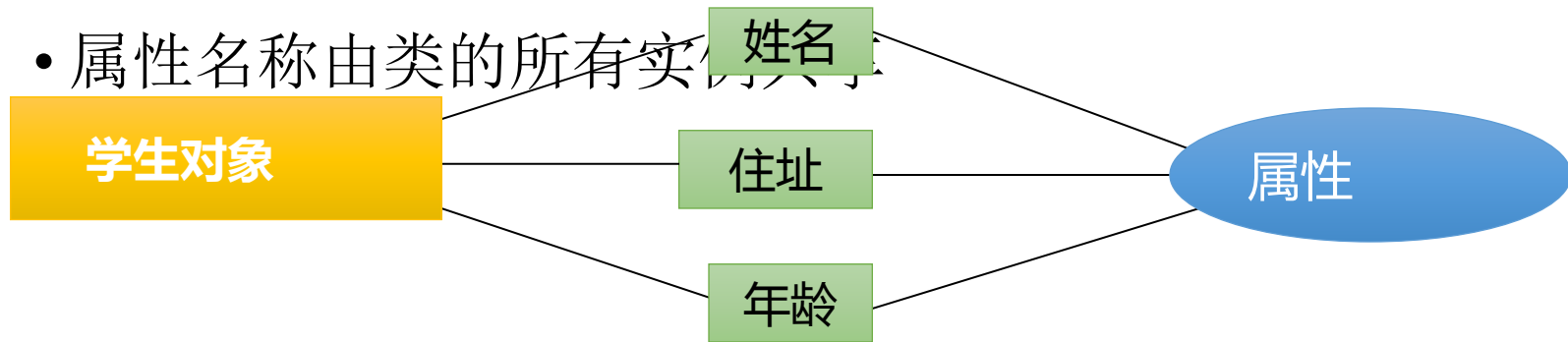
类

- 类以共同特性和操作定义实体
- 类是用于组合各个对象所共有操作和属性的一种机制

类是具有相同属性和行为的一组对象的集合

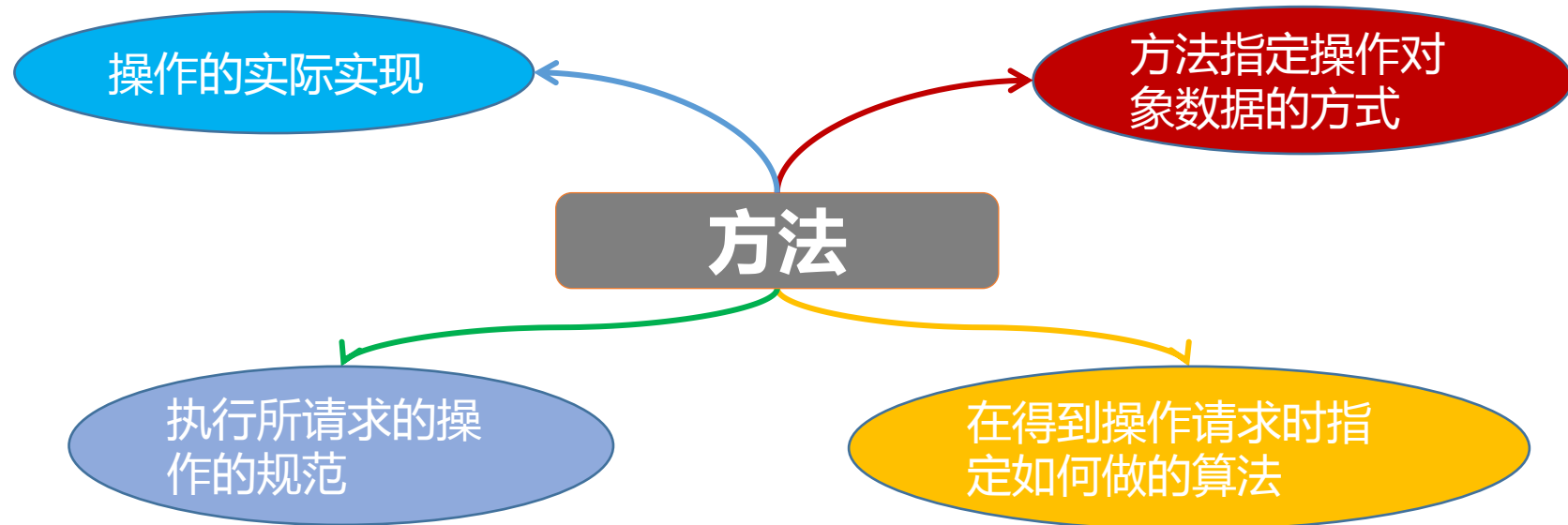
属性

- 事物的特性在类中表示为变量
- 每个对象的每个属性都拥有其特有的值
- 属性名称由类的所有实例共享



在类中表示对象或实体拥有的特性时称为属性

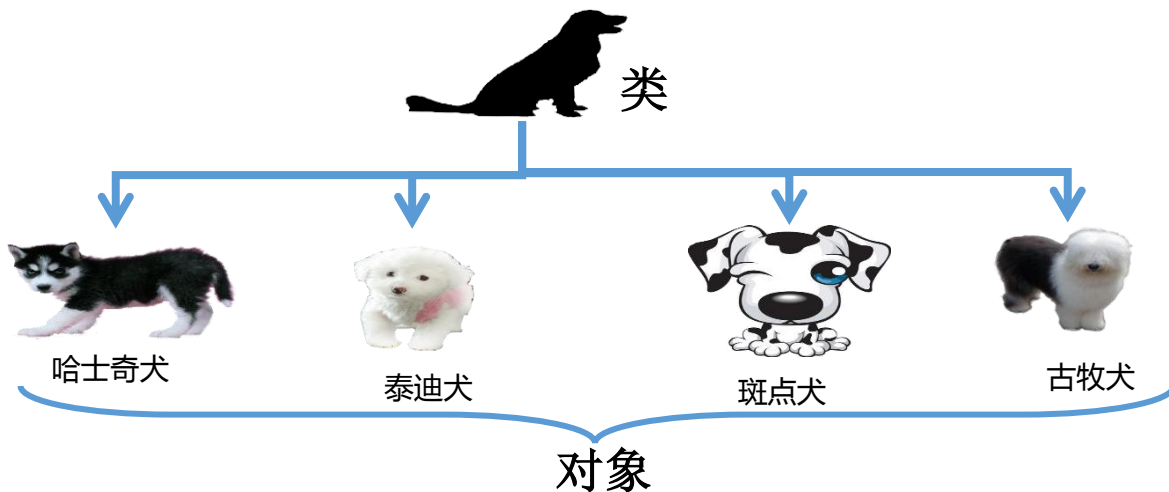
方法



对象执行的操作称为方法。

类和对象的关系

- 类是概念模型，定义对象的所有特性和所需的操作
- 对象是实际的实体,对象是真实的模型，所有属于同一个类的对象都具有相同的特性和操作
- 类是由对象抽象而来，对象是类的实例化



封装

- 信息隐藏,隐藏对象的实现细节, 简化开发
- 将东西包装在一起, 然后以新的完整形式呈现出来
 - 将方法和属性一起包装到一个单元中, 单元以类的形式实现

隐藏属性、方法或实现细节的过程称为封装。

在 Java 中实现类

语法:

```
[修饰符]  class <classname> {  
    <body of class>  
}
```

class 是创建类所使用的关键字,
<classname> 是类的名称,
<body of class> 包含属性和方法的声明。

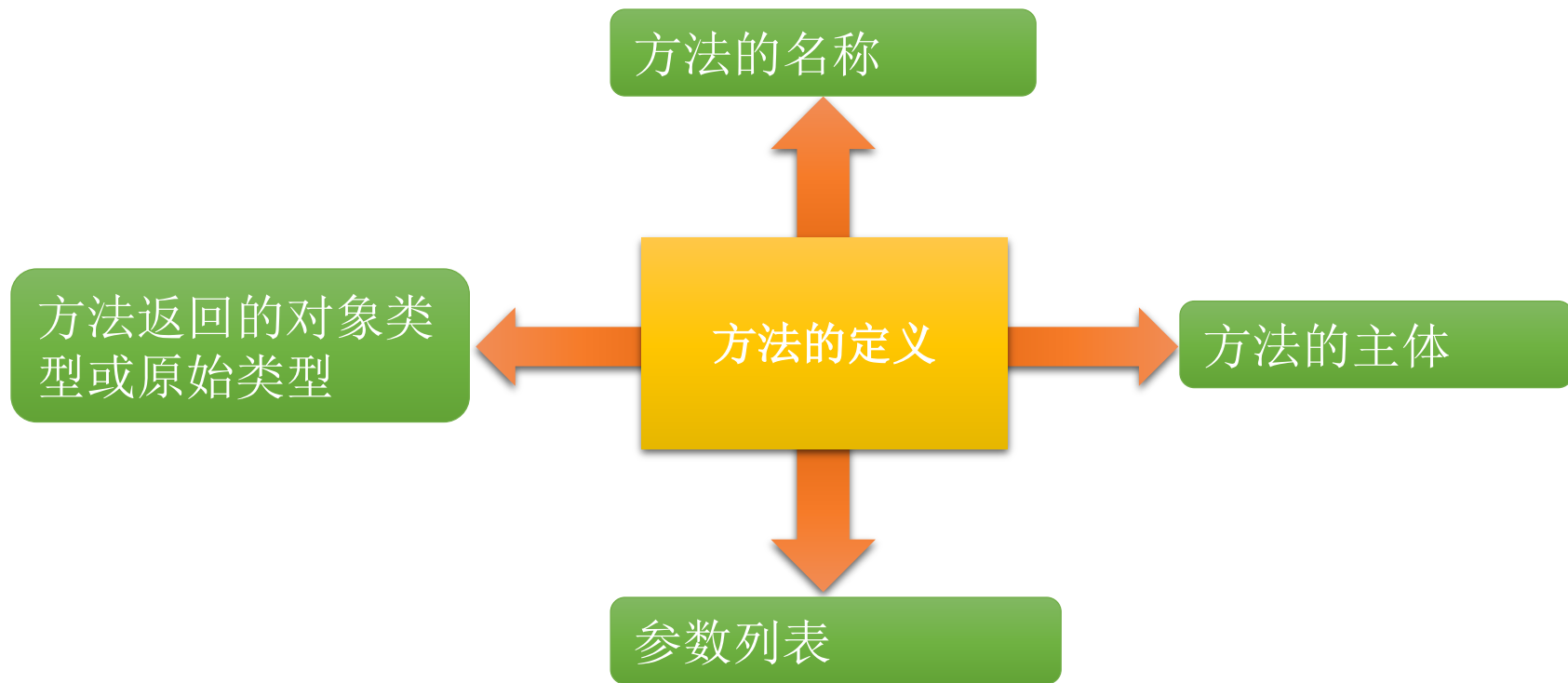
属性

- 属性也称为字段、实例域。它用于描述类的某些特点。属性的声明采用如下形式:
- <修饰符><属性类型><属性名> [=<初始值>];

如:

```
public class Dog{  
    public String name;  
    public int age;  
}
```

类中的方法



类中的方法

语法

```
[修饰符] <return type> <method name> (<type1> <arg1>, <type2> <arg3>,...) {  
    <set of statements>  
}
```

其中,

<return type> 是方法返回值的数据类型

<method name> 是用户自定义的方法名称

方法的参数列表是一组变量声明。

类中的方法

```
public class Person {  
    public String name;  
    private String sex;  
    private int age;  
    /*判断是不是儿童*/  
    public void isChildren () {  
  
        if(age < 18){  
            System.out.println("是儿童");  
  
        }else{  
            System.out.println("是成年人");  
        }  
    }  
    ....  
}
```



方法

属性和方法的访问

- 实例方法可使用圆点符号 “.” 来访问
 - 方法被调用的对象在圆点左边，而方法的名称在圆点右边

```
Person obj= new Person ();  
obj.name= “张三” ;  
obj.age=15;  
obj.isChildren() ;
```

this 关键字

- 用于任何实例方法内，指向当前对象
- this 的值指向对其调用当前方法的对象
- this 关键字可在需要当前类类型的对象引用时使用

this 关键字的示例

```
public class Point {  
    private int x;  
    private int y;  
    public void init (int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

对象的引用

```
public static void main (String args[]){  
    Point p = new Point ();  
    p.init (3,5); //此程序初始化 x = 3 和 y = 5  
}  
}
```

构造器

- 每当创建给定类的实例时就调用的方法
- 与类同名，但没有返回类型
- Java 为对象分配内存，初始化实例变量并调用构造器
- 两种构造器
 - 参数化构造器
 - 隐式构造器

参数化构造器的示例

```
public class SimpleDate {  
    private int month;  
    private int day;  
    private int year;  
  
    public SimpleDate(int m,int d,int y) {  
        month=m;  
        day=d;  
        year=y;  
        System.out.println("日期是 " + m + "/" + d + "/" + y + ".");  
    }  
  
    public static void main(String args[]) {  
        SimpleDate s1,s2;  
        s1=new SimpleDate(12,27,2013);  
        s2=new SimpleDate(3,5,2014);  
    }  
}
```

参数化构造器

包

- 包允许将类组合成较小的单元（类似文件夹），使其易于找到和使用相应的类文件
- 有助于避免命名冲突。在使用许多类时，类和方法的名称很难决定。有时需要使用与其他类相同的名称。包基本上隐藏了类并避免了名称上的冲突
- 包允许在更广泛的范围中保护类、数据和方法。可以在包中定义类，
“包将类名空间划分为更加容易管理的块，
包既是命名机制也是可见度控制机制”

创建包

```
package com.igeekhome.mypackage;
```

声明包

```
public class Calculate {  
    public double volume(double height, double width, double depth) {  
        .....  
    }  
    .....  
}
```

导入包

```
import com.igeekhome.mypackage.Calculate;
```

```
public class PackageDemo {  
    public static void main(String args[]){  
        Calculate calc=new Calculate( );  
        .....  
    }  
}
```

导入包

java.lang包，系统会自动导入，无需我们手动导入

包装类

- Java不将基本数据类型视作对象，也就是说基本数据类型不会被
视为对象处理，它只是一个值，但是有些时候我们需要使用对象

基本数据类型	包装类
boolean	java.lang.Boolean
byte	java.lang.Byte
char	java.lang.Character
short	java.lang.Short
int	java.lang.Integer
long	java.lang.Long
float	java.lang.Float
double	java.lang.Double

Math类

- Math类是Java的一个比较重要的类，它提供了各种数学函数。
 - Math类在java.lang包下，所以系统会自动导入。

例如:

```
double x = 100;
```

```
System.out.print(Math.sqrt(x)); //计算平方根，输出10
```

- Math还提供了很多常用的三角函数、指数函数和对数函数
例如:

```
Math.sin()
```

```
Math.cos()
```

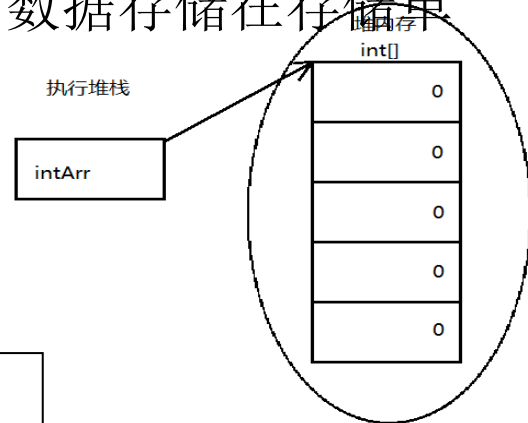
```
Math.tan()
```

数组

- 数组是一种数据结构，用于将相同数据类型的数据存储在存储单元中。
- 数组可以分为一维数组和多维数组
- 一维数组可以使用以下三种方式声明数组：

数据类型[] 标识符；

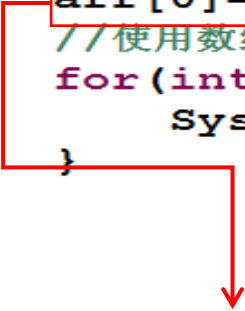
```
int [] arr1; //定义未初始化大小  
int [] arr2=new int[10]; //定义10个元素的数组  
int [] arr3={1,5,6,8,9,12}; //定义并赋予初值
```



数组的访问

- 数组有一个属性`length`可以得到数组的元素个数
- 数组内的元素可以使用下标来访问，Java自动为数组的每个元素

```
public static void main(String[] args) {  
    int[] arr={1,3,5,7,9}; //定义数组并赋予初值  
    arr[0]=-1; //修改数组的第一个元素的值为-1  
    //使用数组的下标，得到数组的元素并打印到控制台  
    for(int i=0;i<arr.length;i++){  
        System.out.println(arr[i]);  
    }  
}
```



arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
1	3	5	7	9

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
-1	3	5	7	9

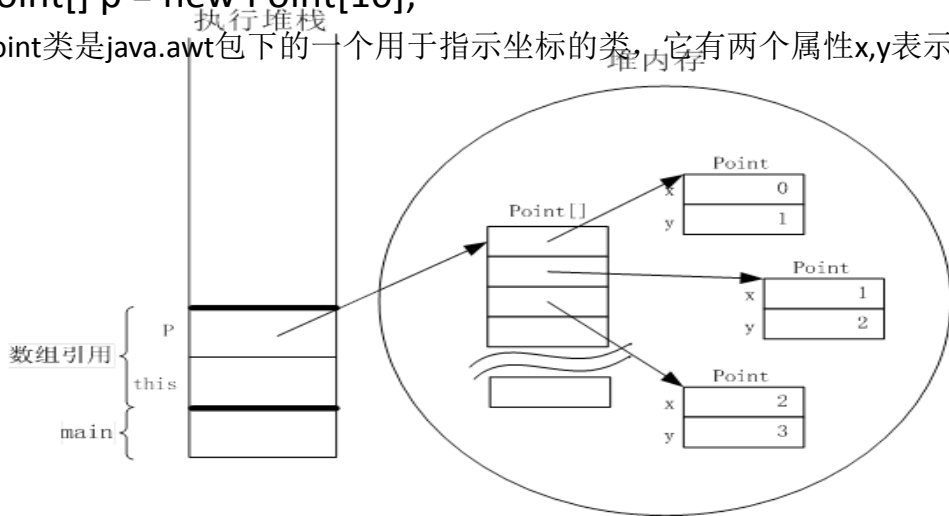
引用类型数组

- 创建引用数据类型的数组可以使用同样的语法:

例如:

```
Point[] p = new Point[10];
```

Point类是java.awt包下的一个用于指示坐标的类，它有两个属性x,y表示横坐标和纵坐标。



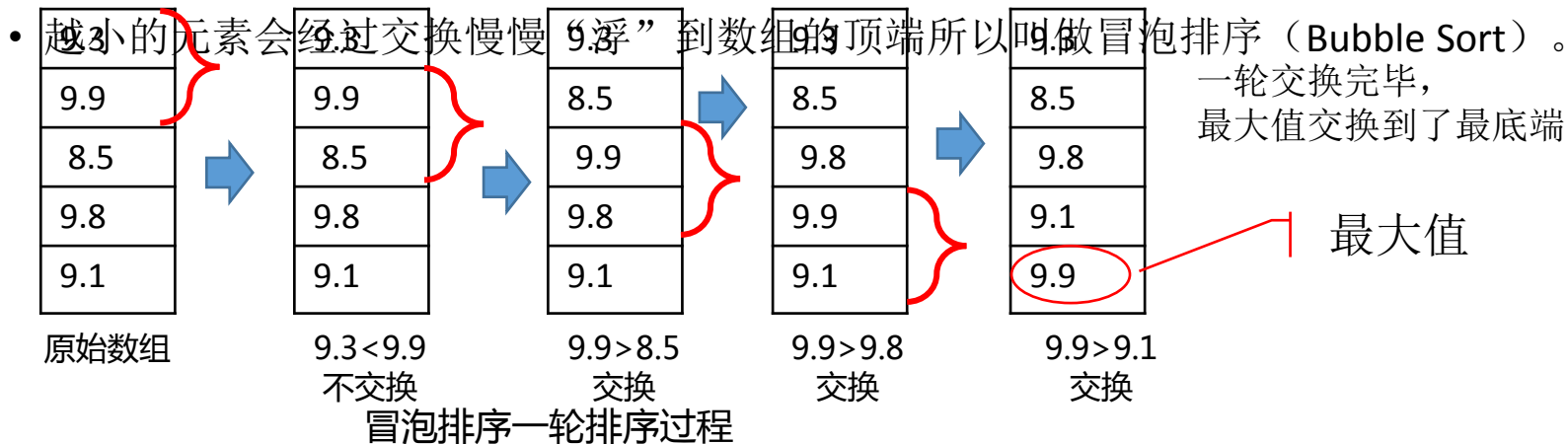
数组排序

- 很多情况下我们需要对数组中的值进行排序
- 常见的排序方法有冒泡排序、插入排序，选择排序和快速排序等

```
// 创建一个数组存放评委的评分  
double[] scores = { 9.3, 9.9, 8.5, 9.8, 9.1 };  
Arrays.sort(scores);
```

算法排序-冒泡排序

- 冒泡排序，是一种计算机科学领域的较简单的排序算法。
- 它重复地访问要排序的数组，一次比较两个元素，如果他们的顺序错误就把他们交换过来。



冒泡排序参考代码

```
/**
 * 冒泡排序
 * @param ary
 */
public static void bubbleSort(int[] ary){
    //控制循环比较的次数
    for(int i = 0; i < ary.length-1; i++){
        for(int j = 0; j < ary.length-i-1; j++){
            if(ary[j] > ary[j+1]){ //判断第j个元素是否比第j+1个大
                //交换原理
                int temp = ary[j];
                ary[j] = ary[j+1];
                ary[j+1] = temp;
            }
        }
    }
}
```


多维数组

- Java中能够创建引用数据类型的数组。我们知道数组也是一个引用数据类型，所以我们还能创建数组的数组（以及数组的数组的数组等等）。这样的数组形式叫做多维数组。

```
int[][] arr= new int[3][2]; //这就创建了一个3行2列的数组。  
arr[0][1]=100; //使用下标赋值:
```

arr[0][0]	arr[0][1]
0	100
arr[1][0]	arr[1][1]
0	0
arr[2][0]	arr[2][1]
0	0

总结

- 理解面向对象思想
- 创建类以及类的属性和方法
- 创建包和导入包
- 会使用Java的常用类
- 会使用正则表达式
- 会使用数组
- 掌握常见的算法排序



400-133-0510
www.igeekhome.com

© 极客营 版权所有 违者必究