Tedu.cn 达内教育

Java语言基础

JSD BASICS DAY06

未来已来 | 教与学的革命

Tedu.cn 达内教育

技能回顾

- 嵌套循环：
  1) 循环中套循环
  2) 执行规则：外层循环走一次，内层循环走所有次
  3) break默认只能跳出当前一层循环

- 数组
  1) 相同数据类型元素的集合
  2) 操作：定义、初始化、访问、遍历、复制、排序

今日目标

01 了解方法的作用、概念

02 掌握方法定义的五要素

03 理解方法有参和无参的适用场景

04 理解方法有返回值和无返回值的适用场景

05 掌握return关键字的用法、方法的重载

06 预计代码量：课上140行、作业280行

01 方法

方法的意义

方法的意义

- 每种编程语言中都有方法的概念，也称为：函数、过程
- 作用:用于封装一段特定的业务逻辑功能----------()

知识讲解

```
System.out.println("hello");
int a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
System.arraycopy(a,1,b,0,4);
int[] b = Arrays.copyOf(a,6);
Arrays.sort(arr);
```

方法的意义(续)

- 建议：方法应该尽可能独立，一个方法只干一件事
- 方法可以被反复多次调用

知识讲解

```
System.out.println("hello");
int a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
System.arraycopy(a,1,b,0,4);
int[] b = Arrays.copyOf(a,6);
Arrays.sort(arr);
```

方法的意义(续)

- 好处：可以减少代码重复，有利于代码维护
- 何时用：只要是一个独立的业务功能，就可以封装到一个方法中

知识讲解

```
int[] arr = new int[10];
for(int i=0;i<arr.length;i++){
  arr[i] = (int)(Math.random()*100);
  System.out.println(arr[i]);
}
```

抽到一个方法中

day05包的ArrayDemo遍历
day05包的MaxOfArray找最中
day05包的ArrayDemo排序

代码重复

方法的定义

定义方法

- 定义方法的五要素 修饰词、返回值类型、方法名、参数列表、方法体

public static

知识讲解

方法1：public static
方法2：public static

定义方法的返回值类型

- 定义方法的五要素:修饰词、返回值类型、方法名、参数列表、方法体

知识讲解

```
System.out.println("hello");
System.arraycopy(a,1,b,0,4);
Arrays.sort(arr);
```

无返回值

```
int    a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
int[]  b = Arrays.copyOf(a,6);
```

有返回值

定义方法的返回值类型(续)

- 方法可以有返回值，也可以无返回值
  1) 无返回值:---------返回值类型统一写成void
  2) 有返回值:---------返回值类型写成特定的数据类型即可

知识讲解

```
System.out.println("hello");
System.arraycopy(a,1,b,0,4);
Arrays.sort(arr);
```

无返回值-------void

```
int    a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
int[]  b = Arrays.copyOf(a,6);
```

有返回值

定义方法的返回值类型(续)

知识讲解

- 何时有返回值？何时没有返回值？

  方法执行完之后：

  1) 若还需要用到方法中的数据----------有返回值

  2) 若不需要用到方法中的数据----------无返回值

```
System.out.println("hello");
System.arraycopy(a,1,b,0,4);
Arrays.sort(arr);
int    a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
int[]  b = Arrays.copyOf(a,6);
```

方法1: public static void

方法2: public static int

定义方法的方法名

知识讲解

- 定义方法的五要素:修饰词、返回值类型、方法名、参数列表、方法体

  方法名 → 见名知意

```
System.out.println("hello");
System.arraycopy(a,1,b,0,4);
Arrays.sort(arr);
int    a = scan.nextInt();
double b = scan.nextDouble();
double c = Math.random();
int[]  b = Arrays.copyOf(a,6);
```

方法1: public static void say

方法2: public static int sum

定义方法的参数列表

知识讲解

- 定义方法的五要素:修饰词、返回值类型、方法名、参数列表、方法体

```
System.out.      println("hello");
System.          arraycopy(a,1,b,0,4);
Arrays.          sort(arr);
int[]   b = Arrays.  copyOf(a,6);
```

有参数

```
int      a = scan.    nextInt();
double b = scan.    nextDouble();
double c = Math.   random();
```

无参数

定义方法的参数列表(续)

知识讲解

- 方法可以有参数，也可以无参数
- 有参可以使方法更加灵活

```
double c = Math.random();     //只能生成0.0到0.9999999999…之间的
假设random()有参：
  double c = Math.random(1,1000);    //生成1到1000之间的
  double c = Math.random(0,99);      //生成0到99之间的
  double c = Math.random(10,35);     //生成10到35之间的
```

方法1: public static void say ( )

方法2: public static int sum (int num1, int num2)

定义方法的方法体

- 定义方法的五要素:修饰词、返回值类型、方法名、参数列表、方法体

方法 → 具体的业务逻辑实现

知识讲解

```
方法1: public static void say()    {
            //业务逻辑实现…
        }
方法2: public static int sum( int num1, int num2 )    {
            //业务逻辑实现…
        }
```

方法的调用

调用方法

知识讲解

- 无返回值：方法名(有参传参);

```
public static void say( ) { //… }

调用：say();
```

return

知识讲解

- return的用法：

1. return 值 ； //1)结束方法的执行 //2)返回结果给调用方
   ----用在有返回值的方法中

```
public static int sum( int num1, int num2 ) {
    int num = num1+num2;
    return num;
}
```

调用方法(续)

知识讲解

- 无返回值：方法名(有参传参);

  public static void say( ) { //… }

  调用： say();

- 有返回值:数据类型 变量 = 方法名(有参传参);

  public static int sum(int num1, int num2) { //… }

  调用：int num = sum(5,6);

return(续)

知识讲解

- return的用法：

  1. return 值； //1)结束方法的执行 2)返回结果给调用方

     --------用在有返回值的方法中

  2. return; //1)结束方法的执行

     --------用在无返回值的方法中

方法的重载

方法的签名

知识讲解

- 方法的签名：方法名+参数列表

```
public static void say( ) { //… }
```

- Java规定：在同一类中，不允许出现签名完全相同的方法

```
public class Test{
    public static void say(){ … }
    public static void say(){ … }     //不允许
    public static int say(){ return 1; }     //不允许
}
```

方法的重载(overloading)

知识讲解

- 定义：发生在同一类中，方法名相同，参数列表不同
- 绑定：编译器在编译时会根据方法的签名自动绑定方法

```
public class Test{
    public static void say(){ ... }
    public static void say(String name){ ... }    //允许
}

say();  //调用无参say方法
say("zhangsan"); //调用String参数say方法
```

评委打分系统

课堂练习

- <<主持人大赛>>有N名评委给选手打分
- 选手的最终得分为：去掉最高分和最低分后的N-2名评委的平均分
- 要求：设计方法完成任务

总结 | SUMMARY

- 方法：封装一段特定的业务逻辑功能、尽可能独立，一个方法只干一件事
- 方法的定义：五要素

  修饰词 返回值类型 方法名(参数列表){ 方法体 }
- 方法的调用：

  1)无返回值：方法名(有参传参);

  2)有返回值：数据类型 变量 = 方法名(有参传参);
- 方法的重载：

  1) 发生在同一类中，方法名相同，参数列表不同

  2) 编译器编译时会根据方法的签名自动绑定调用方法

小T提示

- 方法的应用场景有哪些
- 方法的设计规则有哪些
- 方法重载的应用场景有哪些

Tedu.cn
达内教育

未来已来

AI助力IT技术的全新革命