目录

[Day05.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637306)

[1     回顾](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637307)

[2     构造方法重载](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637308)

[练习1   构造方法重载](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637309)

[3     this](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637310)

[3.1   this.xxxx](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637311)

[3.2   this(...)](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637312)

[4     方法的重载 Overload](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637313)

[5     面向对象](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637314)

[5.1       类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637315)

[5.2       实例](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637316)

[5.3       引用变量](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637317)

[5.4       构造方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637318)

[5.5   this](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637319)

[5.6       方法重载 Overload](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637320)

[6     数组](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637321)

[6.1       数组的创建（3种语法）](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637322)

[练习2   数组](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637323)

[6.2       数组的长度属性](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637324)

[6.3       数组遍历](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637325)

[练习3   最小值的位置下标](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637326)

[练习4   商品录入查询](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637327)

[7     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day05.htm#_Toc5637328)

**Day05.  Java**

**package** day0408;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.print("整数a:");

**int** a = **new** Scanner(System.***in***).nextInt();

    System.***out***.print("整数b:");

**int** b = **new** Scanner(System.***in***).nextInt();

    //把a,b的值传递到 zdgys() 求最大公约数

**int** y = *zdgys*(a, b);

    //把a,b的值传递到 zxgbs() 求最小公倍数

**long** s = *zxgbs*(a, b);

    System.***out***.println("最大公约数："+y);

    System.***out***.println("最小公倍数："+s);

}

**private** **static** **int** zdgys(**int** a, **int** b) {

**int** min = a<b ? a : b;

**for** (**int** i = min; i >= 1; i--) {

        //a,b都能被 i 整除

**if**(a%i==0 && b%i==0) {

**return** i;

        }

    }

**return** 1;//不可能执行，为了编译通过

}

**private** **static** **long** zxgbs(**int** a, **int** b) {

**int** max = a>b ? a : b;

**for** (**int** i = max; ;i += max) {

        //i能被a和b整除

**if**(i%a==0 && i%b==0) {

**return** i;

        }

    }

}

}

**1   回顾**

  数据类型

  基本类型

  引用类型

  八种基本类型

  byte 1,short 2,int 4,long 8

  float 4,double 8

  char 2

  boolean 1

  字面值规则（5条）

  3453，int

  byte a = 127, byte

  3.14, double

  L, F, D

  0b, 0x, 0, \u

  运算规则（5条）

  3/2, 1

  byte a = 2;

a = a+a;//错

a += a; //复合的赋值运算，自动转型

a = 2+2;//编译成：4

  Integer.MAX\_VALUE+1,负数最小值

  2-1.9, 0.1000000000000009

4.35\*100, 434.9999999999999994

  Infinity 无穷大

3.14/0

NaN, Not a Number

Math.sqrt(-6)

  运算符

  +-\*/

  +

  %

  == != > >= < <=

  && || !

  ++

--

a++ 先取值，再自增

++a 先自增，再取值

  1 ? 2 : 3

  =

  += -= ...

  ()

  流程控制

  if-else if-else

  switch-case-default

  byte,short,char,int

  enum

  jdk1.7, String

  for

  while

  do-while

  break

  中断、跳出循环和 switch

  continue

  继续进入循环的下一轮

  循环命名

outer: for(...) {

   for(...) {

      break outer;

      continue outer;

}

}

  对象

  类，模板

  对象、实例，

  从模板创建的具体实例

  实例是数据的打包

  构造方法

  新建对象时实行

  不定义，有默认构造

  构造方法一般用来给属性赋值

  this

  当前对象

  正在调用的对象

**2   构造方法重载**

一个类中，可以定义多个不同参数的构造方法

class A {

public A() {

}

public A(int a) {

}

public A(double d) {

}

public A(int a, double d, String s) {

}

}

new A();

new A(3.14);

new A(5, 3.14, "abc")

**练习1  构造方法重载**

项目：day0501\_构造方法重载

类：day0501.Test1

             Student

Student

**package** day0501;

**public** **class** Student {

**int** id;

String name;

String gender;

**int** age;

**public** Student() {

}

**public** Student(**int** id,String name) {

**this**(id, name, **null**);

}

**public** Student(**int** id,String name,String gender) {

    //把接收的参数数据，传递到下面的构造方法完成赋值运算

    //重载的构造方法之间调用

**this**(id, name, gender, 0);

}

**public** Student(**int** id,String name,String gender,**int** age) {

**this**.id = id;

**this**.name = name;

**this**.gender = gender;

**this**.age = age;

}

**public** String toString() {

**return** id+", "+name+", "+gender+", "+age;

}

}

Test1

**package** day0501;

**public** **class** Test1 {

/\*

 \* Student类

 \* {

 \*     id,

 \*     name,

 \*     gender,

 \*     age,

 \*     toString()

 \* }

 \*

 \* s1 = {

 \*     id: 0,

 \*     name: null,

 \*     gender: null,

 \*     age: 0

 \* }

 \* s2 = {

 \*     id: 6,

 \*     name: "张三",

 \*     gender: null,

 \*     age: 0

 \* }

 \* s3 = {

 \*     id: 6,

 \*     name: "张三",

 \*     gender: "男",

 \*     age: 0

 \* }

 \* s4 = {

 \*     id: 6,

 \*     name: "张三",

 \*     gender: "男",

 \*     age: 23

 \* }

 \*/

**public** **static** **void** main(String[] args) {

    Student s1 = **new** Student();

    Student s2 = **new** Student(6,"张三");

    Student s3 = **new** Student(6,"张三","男");

    Student s4 = **new** Student(6,"张三","男",23);

    System.***out***.println(s1.toString());

    System.***out***.println(s2.toString());

    System.***out***.println(s3.toString());

    System.***out***.println(s4.toString());

}

}

**3   this**

**3.1     this.xxxx**

this，当前对象，正在调用的对象

**3.2     this(...)**

重载的构造方法之间调用

必须是首行代码

**4   方法的重载 Overload**

同名不同参

println()

println(int)

println(double)

println(String)

....

**5   面向对象**

**5.1     类**

模板、图纸

类在第一次用到时，会被加载到内存的“方法区”

**5.2     实例**

从模板创建的具体实例

新建实例时，在“堆内存”中新分配内存空给给这一个实例

**5.3     引用变量**

理解成“遥控器”

引用类型的变量，保存一个实例的内存地址

引用变量的特殊值：null，空，不保存任何实例的地址

**5.4     构造方法**

新建实例时执行的一个特殊方法

**5.5     this**

this.xxx

this(...)

**5.6     方法重载 Overload**

同名不同参

**6   数组**

用来存放一组数据的数据结构

**6.1     数组的创建（3种语法）**

int[] a = new int[6];

  新建 int[] 数组，长度6

  6个位置都存默认值 0

  数组的起始地址值，保存到变量a

int[] a = {6,2,6,8};

  直接初始化数组数据

a = new int[]{7,3,8,1,7,9,3,1};

  为存在的数组变量赋值，直接初始化数据，要添加类型

**练习2  数组**

项目：day0502\_数组

类：day0502.Test1

**package** day0502;

**import** java.util.Arrays;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.println("\n--------------------");

*f1*();

    System.***out***.println("\n--------------------");

*f2*();

    System.***out***.println("\n--------------------");

*f3*();

}

**private** **static** **void** f1() {

    /\*

     \* 先新建 String[]，长度5

     \* 再把数组地址保存到变量a

     \* a = [null, null, null, "abc", null]

     \*                         3

     \* 在下标3位置放入 "abc"

     \*/

    String[] a = **new** String[5];

    //Arrays.toString(a)

    //把数组中的数据，连接成字符串

    System.***out***.println(Arrays.*toString*(a));

    a[3] = "abc";

    System.***out***.println(Arrays.*toString*(a));

}

**private** **static** **void** f2() {

**int**[] a = {6,2,7,3,7,3,1,5};

    System.***out***.println(Arrays.*toString*(a));

    a = **new** **int**[]{7,2,9,3};

    System.***out***.println(Arrays.*toString*(a));

}

**private** **static** **void** f3() {

**int**[] a = **new** **int**[5];

    a[2] = 565;

    System.***out***.println(a[2]);

    System.***out***.println(a[2] \* 2);

    a[2]++;

    /\* 交换 2,3 位置的值

     \* [0, 0, 0, 0, 0]

     \*         2   3

     \* t = 566 \*/

**int** t = a[2];

    a[2] = a[3];

    a[3] = t;

    System.***out***.println(Arrays.*toString*(a));

}

}

**6.2     数组的长度属性**

a.length

  数组一旦创建，长度不可变

  最大下标 a.length - 1

  允许0长度的数组

**6.3     数组遍历**

从头到尾，依次访问每个位置

for(int i=0; i<a.length; i++) {

a[i]

}

**练习3  最小值的位置下标**

day0502\_数组

Test2

**package** day0502;

**import** java.util.Arrays;

**import** java.util.Random;

**public** **class** Test2 {

**public** **static** **void** main(String[] args) {

    //调用 suiJi() 产生一个乱序数组，存到变量a

**int**[] a = *suiJi*();

    System.***out***.println(Arrays.*toString*(a));

    //把数组，传递到 f() 方法求最小值的位置下标

**int** index = *f*(a);

    System.***out***.println(

         "最小值的位置下标："+index);

}

**private** **static** **int** f(**int**[] a) {

    /\*

     \* a = [92, 7, 45, 74, 22, 58, 2, 96, 43, 19]

     \*

     \* 假设第一个值就是最小的

     \* 最小值位置下标 m = 0

     \*

     \* m = 6

     \* a = [92, 7, 45, 74, 22, 58, 2, 96, 43, 19]

     \*      0   1  2   3   4   5   6  7   8   9

     \*

     \* i循环从下标1向后找更小值

     \* 找到更小值，把下标保存到m

     \*/

**int** m = 0;

**for** (**int** i = 1; i < a.length; i++) {

        //a[i] 比 a[m] 更小

**if** (a[i] < a[m]) {

           m = i;//更小值的下标，存到m

        }

    }

**return** m;

}

**private** **static** **int**[] suiJi() {

    /\*

     \* 新建 int[] 数组，长度10

     \* a = [0,0,0,0,0,0,0,0,0,0]

     \*

     \* 遍历填入 100 内随机整数

     \* a = [67,21,7,56,99,78,12,88,54,28]

     \*

     \* 把数组的地址返回到调用位置

     \*/

**int**[] a = **new** **int**[10];

**for** (**int** i = 0; i < a.length; i++) {

        a[i] = **new** Random().nextInt(100);//[0, 100)

    }

**return** a;

}

}

**练习4  商品录入查询**

项目：day0503\_商品录入查询

类：day0503.Test1

             Item

Item 类

{

   id,

   name,

   price,

   number

}

Item[]

items = [{...}, {...}, {...}, {...}, {...}]

**package** day0503;

**import** java.util.Scanner;

**public** **class** Test1 {

/\*

 \* 商品数组

 \* [{...}, {...}, {...}, {...}, {...}]

 \*/

**static** Item[] *items* = {

**new** Item(1,"iPhoneXS",8199,20),

**new** Item(2,"华为P30 pro",5799,80),

**new** Item(3,"小米8",2699,50),

**new** Item(4,"vivo NEX",3599,90),

**new** Item(5,"OPPO FIND",3099,60)

};

**public** **static** **void** main(String[] args) {

    /\*

     \* 1. 商品列表

     \* 2. 商品录入

     \* 3. 商品查询

     \* 4. 统计信息

     \* 5. 退出

     \* 选择：> 2

     \*/

    outer:

**while**(**true**) {

        //调用 caiDan() 方法显示菜单并获得选择的值

**int** c = *caiDan*();

        //判断选项值

**switch**(c) {

**case** 1: *f1*(); **break**;

**case** 2: *f2*(); **break**;

**case** 3: *f3*(); **break**;

**case** 4: *f4*(); **break**;

**case** 5: **break** outer;

        }

    }

}

**private** **static** **int** caiDan() {

    System.***out***.println("-------------------");

    System.***out***.println("1. 商品列表");

    System.***out***.println("2. 商品查询");

    System.***out***.println("3. 商品录入");

    System.***out***.println("4. 统计信息");

    System.***out***.println("5. 退出");

    System.***out***.println("-------------------");

    System.***out***.print("选择：> ");

**int** c = **new** Scanner(System.***in***).nextInt();

**return** c;

}

**private** **static** **void** f1() {

    /\*

     \* [{...}, {...}, {...}, {...}, {...}]

     \*    0      1      2      3      4

     \* 遍历显示商品信息

     \* items[0].id

     \* items[0].name

     \* items[0].toString()

     \*/

**for** (**int** i = 0; i < *items*.length; i++) {

        //println()方法的内部会自动调用对象的toString()方法

        System.***out***.println(*items*[i].toString());

    }

}

**private** **static** **void** f2() {

    /\*

     \* [{...}, {...}, {...}, {...}, {...}]

     \*    0      1      2      3      4

     \*

     \* 查询的商品名：a 存到变量 n

     \* n   items[i].name

     \* 判断商品名称中是否包含 n 子串

     \*

     \* s.indexOf(子串), 在s中寻找子串的起始位置

     \* s = "adfdabcabc"

     \* s.indexOf("bc")   1 下标值

     \* s.indexOf("xx")   -1 特殊值表示不存在

     \*/

    System.***out***.print("查询的商品名：");

    String n = **new** Scanner(System.***in***).nextLine();

    //遍历所有商品比对

**for** (**int** i = 0; i < *items*.length; i++) {

**if**(*items*[i].name.indexOf(n) >= 0) {

           System.***out***.println(*items*[i].toString());

        }

    }

}

**private** **static** **void** f3() {

    /\*

     \* [{xxx}, {xxx}, {xxx}, {xxx}, {...}]

     \*    0      1      2      3      4

     \*/

**for** (**int** i = 0; i < *items*.length; i++) {

        System.***out***.println(

           "录入第 "+(i+1)+" 件商品：");

        System.***out***.print("编号：");

*items*[i].id = **new** Scanner(System.***in***).nextInt();

        System.***out***.print("名称：");

*items*[i].name = **new** Scanner(System.***in***).nextLine();

        System.***out***.print("价格：");

*items*[i].price = **new** Scanner(System.***in***).nextDouble();

        System.***out***.print("数量：");

*items*[i].number = **new** Scanner(System.***in***).nextInt();

    }

    //重新显示列表

*f1*();

}

**private** **static** **void** f4() {

    /\*

     \* 商品总价

     \* 单价均价

     \* 最高总价

     \* 最高单价

     \*/

**double** spzj = 0;//商品总价

**double** djzj = 0;//单价总价

**double** zgzj = 0;//最高总价

**double** zgdj = 0;//最高单价

    //遍历数组

**for** (**int** i = 0; i < *items*.length; i++) {

        Item s = *items*[i];//访问数组中的对象，存到变量s

        spzj += s.price \* s.number;//商品总价

        djzj += s.price;//单价总价

**if**(s.price\*s.number > zgzj) { //最高总价

           zgzj = s.price\*s.number;

        }

**if**(s.price > zgdj) { //最高单价

           zgdj = s.price;

        }

    }

    System.***out***.println("商品总价："+spzj);

    System.***out***.println("单价均价："+(djzj/*items*.length));

    System.***out***.println("最高总价："+zgzj);

    System.***out***.println("最高单价："+zgdj);

}

}

**7   作业**

  重写

  day0501\_构造方法重载

  创建数组，顺序填入 1,2,3,4,....

打乱顺序

  i循环遍历数组

  随机定位一个 j 位置

  i和j位置的值交换

                 j

[5,3,7,4,1,6,2]

     i