目录

[Day11.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330424)

[1     回顾](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330425)

[2     ArrayList（续）](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330426)

[2.1   ArrayList效率](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330427)

[练习1  ArrayList测试](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330428)

[3     LinkedList](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330429)

[3.1   LinkedList 效率](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330430)

[3.2       方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330431)

[3.3       队列操作 Queue，FIFO](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330432)

[3.4       栈操作 Stack，LIFO](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330433)

[练习2  LinkedList测试](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330434)

[练习3   丑数](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330435)

[4     HashMap](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330436)

[练习4  HashMap测试](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330437)

[练习5   字符串的字符统计](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330438)

[5     异常](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330439)

[5.1       错误信息：](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330440)

[5.2       异常的继承结构](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330441)

[5.3       异常捕获](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330442)

[练习6   异常](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330443)

[6     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day11.htm#_Toc6330444)

**Day11.  Java**

**1   回顾**

  数据类型

  基本类型，存值

  引用类型，存地址值

  基本类型

  字面值规则（5条）

  23442, 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; //int+int

  Integer.MAX\_VALUE+1

  2-1.9, 0.1000000000000009

4.35\*100, 434.99999999999994

BigDecimal

  Infinity, 无穷大, 3.14/0

NaN, Not a Number, Math.sqrt(-6)

  运算符

  +-\*/% == != > >= < <=

  && || !

  & | ^ ~ >>  >>>   <<

  ++ --

  ? :

  =

  += -= ^= >>= ...

  ()

  instanceof

  流程控制

  if

  switch

  byte,short,char,int

  enum

  jdk1.7, String

  for

  while

  do-while

  break

  continue

  循环命名

outer: for(...) {}

  数组

  int[] a = new int[6];

  int[] a = {4,3,6,1};

  a = new int[]{3,4,7,1,7};

  a.length， 不可变

  Arrays

  toString()

  sort()

  binarySearch()

  copyOf()

  System.arraycopy()

  二维数组

  int[][] a = new int[3][2];

  int[][] a = new int[3][];

  int[][] a = {{}, {}, {}};

  变量

  局部

  方法，或局部代码块

  必须手动初始化

  作用域

  成员

  类

  默认初始化

  访问范围，受访问能控制符的控制

  方法递归

  一步步简化，简化到最简，再倒推求出结果

  面向对象

  封装、继承、多态

  封装

  类，模板、图纸

  对象、实例

  引用，遥控器

  构造方法

  不定义，加默认

  常见作用：为成员变量赋值

  this

  this.xxx

  this引用当前对象

  this(...)

  本类的构造方法之间调用

  必须是首行代码

  方法重载

  同名不同参

  private

  隐藏，方便维护修改

  继承

  作用：代码重用、复用

  单继承

  子类对象

  父类对象和子类对象绑定，整体作为一个对象

  调用成员时，先找子类，再找父类

  super

  super.xxxx()

  重写时调用父类同一个方法代码

  super()

  super() 默认的隐含调用

  super(参数) 手动调用

  必须是首行代码

  方法重写

  继承的方法，在子类中重新定义从新编写

  覆盖、改写

  多态

  作用：一致的类型

  类型转换

  向上转型

  向下转型

  instanceof

  对真实类型和父类型判断都返回 true

  抽象类

  抽象方法作用：

  作为通用方法，在父类中定义

  要求子类必须实现该方法

  final

  常量，方法，类

  static

  静态属于类

  用类调用

  静态初始化块

  访问控制符

  public, protected, [default], private

  对象的创建过程

  10步

**2   ArrayList（续）**

方法

add(数据)

get(i)

remove(i)

remove(数据)

size()

**iterator()**

辅助新建一个迭代器对象

**2.1     ArrayList效率**

访问任意位置，效率高

增删数据，效率可能降低

**练习1  ArrayList测试**

项目：day1101\_ArrayList

类：day1101.Test1

**package** day1101;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    //插入随机整数，生成有序数字列表

    /\*

     \* ArrayList 代替数组

     \* \*）长度1.5倍自动增长

     \* \*）增删数据，数据复制移位，自动进行

     \*/

    ArrayList<Integer> list = **new** ArrayList<>();

    System.***out***.println("回车执行，按q退出");

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

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

**if**(s.equals("q")) {

**break**;

        }

**int** r = **new** Random().nextInt(1000);

        //Collections是集合的工具类

        //二分法查找，正数是插入点，负数是-(插入点+1)

**int** index = Collections.*binarySearch*(list, r);

**if**(index<0) {// -(插入点+1)

           index = -index-1;//得到插入点

        }

        list.add(index, r);//向index位置插入数据

        System.***out***.println(list);

    }

    //迭代器遍历集合

    //Iterator - 迭代器

    //新建迭代器，必须用集合的方法辅助创建

    Iterator<Integer> it = list.iterator();

    //当还有数据

**while**(it.hasNext()) {

        //取下一个

        Integer n = it.next();

        System.***out***.println(n);

    }

    //背着写迭代遍历

}

}

**3   LinkedList**

双向链表

a={

  value=8,

  prev=c,

  next=b

}

b={

  value=2,

  prev=a,

  next=c

}

c={

  value=2,

  prev=b,

  next=a

}

**3.1     LinkedList 效率**

两端效率高

**3.2     方法**

和ArrayList具有相同的方法

LinkedList 两端操作数据的方法

addFirst(数据)

addLast(数据)

getFirst()

getLast()

removeFirst()

removeLast()

**3.3     队列操作 Queue，FIFO**

offer()      addLast()

peek()       getLast()

poll()       removeFirst()

**3.4     栈操作 Stack，LIFO**

push()       addFirst()

pop()        removeFirst()

**练习2  LinkedList测试**

项目：day1102\_LinkedList

类：day1102.Test1

**package** day1102;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**public** **class** Test1 {

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

    /\*

     \* add(), get(), remove(), size()

     \*

     \* 添加 1000 万数据量

     \* [{1}, {1}, {1}, {1} ...]

     \*/

    LinkedList<Integer> list = **new** LinkedList<>();

**long** t;

    t = System.*currentTimeMillis*();

**for** (**int** i = 0; i < 100000; i++) {

        list.add(1);//添加1000万数据

    }

    System.***out***.println(System.*currentTimeMillis*()-t);

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

    t = System.*currentTimeMillis*();

    //访问头部和尾部

    System.***out***.println(list.getFirst());

    System.***out***.println(list.getLast());

    System.***out***.println(System.*currentTimeMillis*()-t);

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

    t = System.*currentTimeMillis*();

    System.***out***.println(list.get(50000));//中间

    System.***out***.println(System.*currentTimeMillis*()-t);

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

    t = System.*currentTimeMillis*();

**for** (**int** i = 0; i < list.size(); i++) {

        list.get(i);

    }

    System.***out***.println(System.*currentTimeMillis*()-t);

    System.***out***.println("\n--迭代器遍历---------------");

    t = System.*currentTimeMillis*();

    Iterator<Integer> it = list.iterator();

**while**(it.hasNext()) {

        it.next();

    }

    System.***out***.println(System.*currentTimeMillis*()-t);

}

}

**练习3  丑数**

被 2,3,5 整除的数

1也是丑数

2326520/2/2/2/3/3/5/5/5

求第 n 个丑数

项目：day1103\_丑数

类：day1103.Test1

**package** day1103;

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    System.***out***.print("求第几个丑数：");

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

**long** r = *f*(n);

    System.***out***.println(r);

}

//低效算法

**private** **static** **long** g(**int** n) {

**if**(n == 1) {

**return** 1;

    }

**int** count = 1;

**for**(**long** i=2; ;i++){

        //i是否是丑数

**long** j = i;

        //当j能被2整除，除2

**while**(j%2==0) j /= 2;

**while**(j%3==0) j /= 3;

**while**(j%5==0) j /= 5;

        //结果j是不是1

**if**(j==1) {

           count++;

**if**(count == n) {

**return** i;

           }

        }

    }

}

**private** **static** **long** f(**int** n) {

    /\*

     \* 10 12 16

     \* ------------------2

     \* 9 12 15 18 24

     \* ------------------3

     \* 10 15 20 25 30 40

     \* ------------------5

     \*

     \* r = 1 2 3 4 5 6 8

     \*

     \* \*)先创建三个集合

     \* \*)初始分别放入2,3,5

     \* \*)从头部移除最小值

     \* \*)最小值分别乘2,3,5放入三个集合

     \*/

**if**(n == 1) {

**return** 1;

    }

    LinkedList<Long> list2 = **new** LinkedList<>();

    LinkedList<Long> list3 = **new** LinkedList<>();

    LinkedList<Long> list5 = **new** LinkedList<>();

    list2.add(2L);

    list3.add(3L);

    list5.add(5L);

**long** r = 0;

    //从第2个开始，求到第n个

**for** (**int** i = 2; i <= n; i++) {

        //从头部移除最小值

**long** a = list2.getFirst();

**long** b = list3.getFirst();

**long** c = list5.getFirst();

        r = Math.*min*(a, Math.*min*(b, c));

**if**(r == a) list2.removeFirst();

**if**(r == b) list3.removeFirst();

**if**(r == c) list5.removeFirst();

        //\*2,3,5放入集合

        list2.add(r \* 2);

        list3.add(r \* 3);

        list5.add(r \* 5);

    }

**return** r;

}

}

**4   HashMap**

哈希表、散列表

存放键值对数据

|  |  |
| --- | --- |
| key | value |
| 9527 | 唐伯虎 |
| 9528 | 华夫人 |
| 9529 | 祝枝山 |
| 9530 | 旺财 |
| 9531 | 小强 |
| 9532 | 如花 |

用键，可以快速定位数据，来提取键对应的值

键：

  不重复

  无序

方法：

**put(key, value)**

放入键值对数据，重复的键，新值覆盖旧值

**get(key)**

获得键对应的值，键不存在，得到null

**remove(key)**

移除键值对数据，返回被移除的值

**size()**

键值对的数量

**练习4  HashMap测试**

项目：day1104\_HashMap

类：day1104.Test1

**package** day1104;

**import** java.util.HashMap;

**public** **class** Test1 {

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

    HashMap<Integer, String> map = **new** HashMap<>();

    map.put(99527, "唐伯虎");

    map.put(99528, "华夫人");

    map.put(99529, "祝枝山");

    map.put(99530, "旺财");

    map.put(99531, "小强");

    map.put(99532, "石榴姐");

    map.put(99533, "秋香");

    map.put(99533, "如花");//新值替换旧值

    map.put(99534, **null**);

    map.put(**null**, "---");

    System.***out***.println(map.size());

    System.***out***.println(map);

    System.***out***.println(map.get(9527));

    System.***out***.println(map.get(9999));

    System.***out***.println(map.remove(9531));

    System.***out***.println(map);

}

}

**练习5  字符串的字符统计**

"abacdedac"

    i

key  value

a     2

b     1

c     1

Test2

**package** day1104;

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** Test2 {

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

    System.***out***.println("输入字符串：");

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

    //统计方法

    HashMap<Character, Integer> map = *tongJi*(s);

    System.***out***.println(map);

}

**private** **static** HashMap<Character, Integer> tongJi(String s) {

    /\*

     \* \*)先新建 HashMap 对象 map

     \* \*)i循环遍历字符串s

     \*      \*)从s取出i位置的一个字符 c

     \*      \*)从 map 取字符c对应的计数值 Integer count

     \*      \*)count是null

     \*           \*)放入c -- 1

     \*      \*)否则

     \*           \*)放入c -- count+1

     \*

     \* \*)返回 map

     \*/

    HashMap<Character, Integer> map = **new** HashMap<>();

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

**char** c = s.charAt(i);

        Integer count = map.get(c);

**if**(count == **null**) {

           map.put(c, 1);

        } **else** {

           map.put(c, count+1);

        }

    }

**return** map;

}

}

**5   异常**

封装错误信息的对象

**5.1     错误信息：**

  类型

  提示消息

  行号

**5.2     异常的继承结构**

Throwable

   |- Error 系统级错误

   |- Exception 可修复的错误

      |- 其他Exception

      |- RuntimeException

         |- NullPointerException

         |- ArrayIndexOutOfBoundsException

         |- NumberFormatException

         |- ArithmeticException

         |- ClassNotFoundException

         |- ...

**5.3     异常捕获**

try {

} catch(AException e) {

} catch(BException e) {

} catch(父类型Exception e) {

} finally {

不管出不出错，都会执行

}

try {

} catch(AException e) {

}

try {

} finally {

}

**练习6  异常**

项目：day1105\_异常

类：day1105.Test1

**package** day1105;

**import** java.util.Scanner;

**public** **class** Test1 {

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

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

**try** {

*f*();

**break**;

        } **catch** (ArrayIndexOutOfBoundsException e) {

           System.***out***.println("输入两个！");

           e.printStackTrace();

        } **catch** (ArithmeticException e) {

           System.***out***.println("不能除0！");

        } **catch** (Exception e) {

           System.***out***.println("出错，请重试");

        } **finally** {

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

        }

    }

}

**private** **static** **void** f() {

    /\*

     \* "5235,45" --> ["5235", "45"]

     \*                  0       1

     \*

     \* 5235 / 45

     \*/

    System.***out***.println("输入逗号隔开的两个整数");

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

    String[] a = s.split(",");

**int** n1 = Integer.*parseInt*(a[0]);

**int** n2 = Integer.*parseInt*(a[1]);

**int** r = n1 / n2;

    System.***out***.println(r);

    /\*

     \* 3453,33       345     abc,def    345,0

     \*/

}

}

**6   作业**

  重写

  day1102\_LinkedList

  day1103\_丑数

  day1104\_HashMap

  day1001\_猜游戏

添加异常捕获，重新输入