目录



# Day05.Java

# 1 回顾

- 数据类型
  - 基本类型（8种）
  - 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
    true 00000001
    false 00000000
  - 运算规则（5条）
    - 3/2, 1
    - byte,short,char
      byte a = 3;
      a << 24
    - Integer.MAX_VALUE+1, 最小值
      01111111111111111111111111111111 +1
      10000000000000000000000000000000
    - 2-1.9, 0.10000000000009
      4.35*100, 434.9999999999999 4
    - Infinity 3.14/0
      NaN Math.sqrt(-3)

- 运算符
  - +-*/

- %
- == != > >= < <=
- && || !
- & | ^ ~ >> >>> <<

  异或：对同一个值异或两次，得到原值

  110 <<1

  1100

  左移1位，相当于*2，右移一位，相当于/2
- ++

  --

  print(a++)

  print(++a)
- =
- += -= /= ^= >>=

```
byte a = 3;
a = (byte) (a+a);
a += a;
a += 3.14;
```

- ? :

```
1 ? 2 : 3
```

- ()
- 优先级

- 流程控制
  - if-else if-else
  - switch-case-default
    - byte,short,char,int
    - enum
    - jdk1.7 String
  - for
  - while 先判断，后执行
  - do-while 先执行，后判断，至少执行一次
- 数组
  - 创建
    - int[] a = new int[5];
    - int[] a = {3,5,7,34,2};
    - a = new int[]{2,6,8,3};
  - 长度属性 a.length
  - 遍历

```
for(int i=0;i<a.length;i++) {
   a[i]
}
```

- 变量
  - 局部变量
    - 方法，或局部代码块
    - 作用域，定义的带括号范围
    - 手动初始化
  - 成员变量
    - 类中
    - 访问控制符
    - 自动初始化，有默认值
      - 数字 0, 0.0
      - 布尔值 false

- 引用类型 null

- 方法
  - void 无返回值
  - return; 返回，方法到此结束，回到调用位置继续执行
  - return xxxxx;
    把数据返回到调用位置

## 求π值

```
π/4 = 1/1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 ...
```

项目：day0501_求π值
类：day0501.Test1

```
package day0501;

import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.print("次数： ");
      int n = new Scanner(System.in).nextInt();

      double pi = f(n);
      System.out.println(pi);
   }

   private static double f(int n) {
      double sum = 0;
      for(int a=1,b=1,i=1; i<=n; a*=-1,b+=2,i++) {
         sum += a/(double)b;
      }
      return sum*4;
   }
}
```

## 质数

2,3,5,7,11,13,17,19,23....
判断n是否是质数
从2到n开方+1范围，寻找能把n整除的值
找到，n不是质数
找不到，n是质数

项目：day0502_质数
类：day0502.Test1

```
package day0502;
```

```
import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.print("输入整数: ");
      int n = new Scanner(System.in).nextInt();

      if(isPrime(n)) {
         System.out.println("是质数");
      } else {
         System.out.println("不是质数");
      }
   }

   private static boolean isPrime(int n) {
      if(n==2) {
         return true;
      }
      if(n<2) {
         return false;
      }
      //n开方+1
      double max = Math.sqrt(n) + 1;
      //在2到max范围，找能把n整除的值
      for(int i=2; i<max; i++) {
         if(n % i == 0) {
            //n不是质数
            return false;
         }
      }
      return true;//n是质数
   }
}
```

## Test3

```
package day0502;

import java.util.Scanner;

public class Test3 {
   public static void main(String[] args) {
      System.out.print(
            "输入整数n，求n内质数的数量: ");
      int n = new Scanner(System.in).nextInt();

      int count = f(n);
```

```
        System.out.println(count);
    }

    private static int f(int n) {
        if(n==2) {
            return 1;
        }
        if(n<2) {
            return 0;
        }

        //计数变量，有一个已知的质数
        int count = 1;
        //从3到n寻找质数
        outer:
        for(int i=3;i<=n;i++) {
            //判断i是不是质数
            //i开方+1
            double max = Math.sqrt(i)+1;
            //从2到max范围，找能把i整除的值
            for(int j=2;j<max;j++) {
                if(i % j == 0) {//i不是质数
                    continue outer;
                }
            }
            count++; //i是质数
        }
        return count;
    }
}
```

## 冒泡排序

项目：day0503_冒泡排序
类：day0503.Test1

```
package day0503;

import java.util.Arrays;
import java.util.Random;

public class Test1 {
    public static void main(String[] args) {
        //随机的乱序数组
        int[] a = suiJi();
        System.out.println(Arrays.toString(a));
        System.out.println("----------------");

        sort(a); //对数组a排序
```

```
        System.out.println("----------------");
        System.out.println(Arrays.toString(a));
    }

    private static int[] suiJi() {
        //随机产生一个整数值，存到n
        //范围 5+ [0, 6)
        int n = 5 + new Random().nextInt(6);
        //新建int[]数组，存到变量a
        //长度 n
        int[] a = new int[n];
        //遍历数组，填入 100 内随机值
        for(int i=0;i<a.length;i++) {
            a[i] = new Random().nextInt(100);
        }
        //返回数组 a
        return a;
    }

    private static void sort(int[] a) {
        /*
         * 冒泡排序
         *                                    j
         * [8, 25, 59, 94, 88, 59, 63, 75, 74]
         *                i
         * j循环，把较小值向前交换，
         * 最终，把最小值，交换到 i 位置
         *
         *
         *        j循环开始之前 flag = flase 没有交换
         *
         *
         *
         * [2, 39, 51, 90, 92, 93]
         *              i
         *
         *           if(flag是false) {
         *
         *           }
         *
         *
         */
        //i循环从头到尾遍历
            //j循环从a.length-1 到 j>i  递减
                //如果j位置值，比 j-1位置值小
                    //j和j-1位置的值交换
        for(int i=0;i<a.length;i++) {
            boolean flag = false;
            //j循环较小值向前交换，最小值交换到i位置
            for(int j=a.length-1; j>i; j--) {
                if(a[j]<a[j-1]) {
                    int t = a[j];
```

```
                a[j] = a[j-1];
                a[j-1] = t;
                flag = true;
            }
        }
        //flag是false，没有交换数据，
        //数据位置都是正确的
        if(!flag) {
            break;
        }
        System.out.println(Arrays.toString(a));
    }


}
}
```

## 二分法查找、折半查找

在有序数组中，查找目标值所在的下标位置

项目：day0504_二分法查找
类：day0504.Test1

```
package day0504;

import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        int[] a = suiJi();
        //对a数组排序
        //优化的快速排序算法
        Arrays.sort(a);
        System.out.println(Arrays.toString(a));
        System.out.println("查找的目标值：");
        int t = new Scanner(System.in).nextInt();

        //二分法查找
        //如果目标值不存在，返回负数特殊值
        int index = binanrySearch(a, t);
        System.out.println(index);
    }

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

```
        * 1.随机整数 n,  5+[0,6)
        * 2.新建int[]数组，长度n
        * 3.遍历填入100内随机值
        * 4.返回数组
        */
       int n = 5+ new Random().nextInt(6);
       int[] a = new int[n];
       for(int i=0;i<a.length;i++) {
          a[i] = new Random().nextInt(100);
       }
       return a;
    }

    private static int binanrySearch(int[] a, int t) {
       //定义下标变量
       //lo=0
       //hi=a.length-1
       //mid;

       //当lo<=hi
            //计算中间位置，赋给mid
            //如果mid位置的值比目标值t大
                  //hi定位到mid-1
            //否则如果小
                  //lo定位mid+1
            //否则
                  //返回 mid 下标值

       //循环结束，没有数据，
       //返回负数无意义值 -1
       int lo = 0;
       int hi = a.length - 1;
       int mid;
       while(lo<=hi) {
          mid = (lo+hi)/2;
          if(a[mid] > t) {
             hi = mid-1;
          } else if(a[mid] < t) {
             lo = mid+1;
          } else {
             return mid;
          }
       }

       return -1;


    }
}
```

# 2 数组工具类 Arrays

java.util.Arrays

- Arrays.toString(数组)
  把数组中的值，连接成字符串
  "[值1，值2，值3]"

- Arrays.copyOf(数组，长度)
  把数组，复制成指定长度的新数组
  更长，增长位置
  更短，截取

- Arrays.sort(数组)
  数组排序
  基本类型数组，优化的快速排序
  引用类型数组，优化的合并排序

- Arrays.binarySearch(数组，目标值)
  在有序数组中，查找目标值的位置下标
  找不到，返回负数无意义值

- Arrays.fill(数组，值)
  填满指定的值

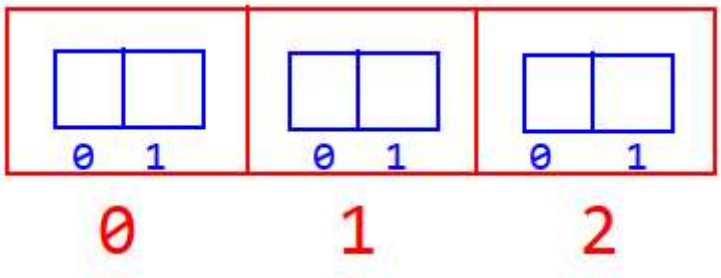
# 3 数组复制

- Arrays.copyOf()
  会创建新数组

- System.arraycopy(
  原数组，
  原数组的起始位置，
  目标数组，
  目标数组的起始位置，
  复制的数据数量)
  不会新建数组，目标数组必须实现存在，
  使用操作系统底层方法，直接赋值内存块

# 4 二维数组

- 存放数组的数组




- 创建二维数组
  - int[][] a = new int[3][2];
    - 共创建了4个数组
    - 外围数组长度 3
    - 内部的三个数组，长度是 2
    - 外围数组中，保存的是内部数组的地址
    - 内部数组中保存的是默认值 0

  - int[][] a = new int[3][];
    - 只新建一个外围数组
    - 内部保存默认值 null
    - 可以之后，再新建数组，存入外围数组

```
a[0] = new int[5];
a[1] = new int[]{2,5,1};
a[2] = new int[]{6,3,5,7};
```

  - int[][] a = {

```
        {5,3,7,5,2},
        {7,5,1},
        {3,6,2,8}
};
```

- 二维数组的遍历

```
for(int i=0;i<a.length;i++) {
   for(int j=0;j<a[i].length;j++) {
      a[i][j]
   }
}
```

## 二维数组

项目：day0505_二维数组
类：day0505.Test1

```
package day0505;

public class Test1 {
   public static void main(String[] args) {
      char[][] a = {
            {'渭','城','朝','雨','浥','轻','尘'},
            {'客','舍','青','青','柳','色','新'},
            {'劝','君','更','尽','一','杯','酒'},
```

```
            {'西','出','阳','关','无','故','人'}
        };

        /*
         * 从右向左，竖排显示
         *
         *      a.length-1 2 1 0  j
         *   i
         *   0   西劝客渭
         *   1   出君舍城
         *   2   阳更青朝
         *   ...
         *
         * a[0].length
         */
        for(int i=0;i<a[0].length;i++) {
           for(int j=a.length-1; j>=0; j--) {
              System.out.print(a[j][i]);
           }
           System.out.println();
        }
    }
}
```

# 5 方法递归

- 在方法中，调用自身
- 一步一步的简化问题，把问题简化成最简问题，再倒推求出结果
- 一般不会同时做多次递归调用，否则，运算量会急剧增加，考虑用循环代替

```
void f() {
   f();
}
```

```
f(5)
f(4)
f(3)
f(2)
```

```
f(5)
  5*f(4)
       4*f(3)
            3*f(2)
                 2*f(1)
                     1*f(0)
```

## 递归求阶乘

项目：day0506_递归求阶乘
类：day0506.Test1

```
package day0506;

import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.print("输入整数求阶乘:");
      int n = new Scanner(System.in).nextInt();
      long r = f(n);
      System.out.println(r);
   }

   private static long f(int n) {
      //最简问题
      if(n == 0) {
         return 1;
      }
      return n*f(n-1);
   }
}
```

## 斐波那契数

1 1 2 3 5 8 13 21 34 55 89 144...

求第n个斐波那契数

项目：day0507_斐波那契数
类：day0507.Test1

```
package day0507;

import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.print("求第几个斐波那契数：");
      int n = new Scanner(System.in).nextInt();
      long r = g(n);
      System.out.println(r);
   }
```

```
    private static long g(int n) {
        /*
         * 1 1 2 3 5 8
         *         a b
         *
         * b = a+b
         * a = b-a
         */
        long a = 1;
        long b = 1;
        //从第3个求到第n个
        for(int i=3; i<=n; i++) {
            b = a+b;
            a = b-a;
        }
        return b;
    }

    //反例
    private static long f(int n) {
        //最简问题
        if(n==1 || n==2) {
            return 1;
        }
        return f(n-1) + f(n-2);
    }
}
```

## 汉诺塔

项目：day0508_汉诺塔
类：day0508.Test1

```
package day0508;

import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.print("玩几层汉诺塔：");
        int n = new Scanner(System.in).nextInt();

        f(n, "A", "B", "C");
    }

    private static void f(
            int n,
            String z1,
            String z2,
```

```
            String z3) {
        if(n==1) {
            System.out.println(z1+"->"+z3);
            return;
        }

        //n-1层，从z1->z3->z2
        f(n-1, z1, z3, z2);
        //底部的一层，从z1->z2->z3
        f(1, z1, z2, z3);
        //z2上的n-1层，从z2->z1->z3
        f(n-1, z2, z1, z3);
    }
}
```

# 6 作业

- 重写
  - day0502_质数，n内质数的数量
  - day0503_冒泡排序
  - day0504_二分法查找
- 有序数组并归

```
[1,1,3,4,8,12]
                j
[3,4,7,12,12,45,99,102,931]
                k
```

合并成一个有序数组

```
[1,1,3,3,4,4,7,8,12,12,12          ]
                                  i
```

  - j和k位置值，较小值放入i位置，对应下标递增
  - 特殊情况
    - j越界,k后面的值，全部放入i位置
    - k越界,j后面的值，全部放入i位置