第三次集训· Java 在 ACM 中的应用

编辑：段昊宇

(1) 在一般比赛中，Java 程序会有额外的时间和空间，而实际上经过实验，在执行计算密集任务的时候 Java 并不比 C/C++慢多少，只是 JVM 虚拟机启动较慢而已。

(2) Java 简单而功能强大，有些东西用 Java 实现起来更为方便，比如高精度。

(3) 用 Java 不易犯细微的错误，比如 C/C++中的指针，“if (n = m) ... ” 等

(4) 目前来看 Eclipse 已成基本配置，写 Java 程序反而比 C/C++更方便调试。在具体竞赛时也算多一种选择。

(5) 学会 Java 对以后工作有好处。现在国外很多地方会 Java 的人比会 C/C++的人多。

1. 基本输入输出：

(1) JDK 1.5.0 新增的 Scanner 类为输入提供了良好的基础，简直就是为 ACM-ICPC 而设的。

一般用法为：

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main (String[] args) {
        Scanner in = new Scanner (System.in);
        /* .....代码段 */
    }
}
```

(2)

Scanner 类方法	对应 C 操作	对应 C++操作
int n = cin.nextInt();	scanf("%d", &n);	cin >> n;
String s = cin.next();	scanf("%s", s);	cin >> s;
double t = cin.nextDouble();	scanf("%lf", &t);	cin >> t;
String s = cin.nextLine();	gets(s);	cin.getline(...);

(3) 输出一般可以直接用 System.out.print() 和 System.out.println()，前者不输出换行，而后者输出。

比如：`System.out.println(n);` // n 为 int 型

同一行输出多个整数可以用

```
System.out.println(new Integer(n).toString() + " " + new Integer(m).toString());
```

也可重新定义：

```
PrintWriter cout = new PrintWriter(new BufferedOutputStream(System.out));
        cout.println(n);
```

(4) 对于输出浮点数保留几位小数的问题，可以使用 DecimalFormat 类，

```
import java.text.*;
```

```
DecimalFormat f = new DecimalFormat("#.00#");
```

```
DecimalFormat g = new DecimalFormat("0.000");

double a = 123.45678, b = 0.12;

System.out.println(f.format(a));

System.out.println(f.format(b));

System.out.println(g.format(b));
```

这里 0 指一位数字，#指除 0 以外的数字。

2. 大数字

BigInteger 和 BigDecimal 是在 java.math 包中已有的类，前者表示整数，后者表示浮点数

用法： 不能直接用符号如+、-来使用大数字，例如：

```
(import java.math.*)  // 需要引入 java.math 包
BigInteger a = BigInteger.valueOf(100);
BigInteger b = BigInteger.valueOf(50);
BigInteger c = a.add(b)   // c = a + b;
```

主要有以下方法可以使用：

```
BigInteger add(BigInteger other)
BigInteger subtract(BigInteger other)
BigInteger multiply(BigInteger other)
BigInteger divide(BigInteger other)
BigInteger mod(BigInteger other)
int compareTo(BigInteger other)
static BigInteger valueOf(long x)
```

输出大数字时直接使用 System.out.println(a) 即可。

3.Date()类 和 Calender()类

java.util.GregorianCalendar

Calendar 类的功能要比 Date 类强大很多，而且在实现方式上也比 Date 类要复杂一些

1、Calendar 类对象的创建

Calendar 类是一个抽象类 ,在实际使用时实现特定的子类的对象。由于 Calendar 类是抽象类 ,且 Calendar 类的构造方法是 protected 的，所以无法使用 Calendar 类的构造方法来创建对象，API 中提供了 getInstance 方法用来创建对象。

a、创建一个代表系统当前日期的 Calendar 对象

Calendar c = Calendar.getInstance();//默认是当前日期

b、创建一个指定日期的 Calendar 对象

使用 Calendar 类代表特定的时间，需要首先创建一个 Calendar 的对象，然后再设定该对象中的年月日参数来完成。

//创建一个代表 2013 年 5 月 5 日的 Calendar 对象

```
Calendar c1 = Calendar.getInstance();
c1.set(2013, 5 - 1, 5);
```

Calendar 类对象字段类型

Calendar 类中用一下这些常量表示不同的意义，jdk 内的很多类其实都是采用的这种思想
Calendar.YEAR——年份
Calendar.MONTH——月份
Calendar.DATE——日期
Calendar.DAY_OF_MONTH——日期，和上面的字段意义完全相同
Calendar.HOUR——12 小时制的小时
Calendar.HOUR_OF_DAY——24 小时制的小时
Calendar.MINUTE——分钟
Calendar.SECOND——秒
Calendar.DAY_OF_WEEK——星期几

3、Calendar 类对象信息的设置与获得

a、Calendar 类对象信息的设置

●Set 设置

如：Calendar c1 = Calendar.getInstance();

调用：**public final void set(int year,int month,int date)**

c1.set(2013, 5 - 1, 5);//把 Calendar 对象 c1 的年月日分别设这为：2013、5、5

利用字段类型设置

如果只设定某个字段，例如日期的值，则可以使用如下 set 方法：

调用：public void set(int field,int value)

//把 c1 对象代表的日期设置为 10 号，其它所有的数值会被重新计算

```
c1.set(Calendar.DATE,10);
```

//把 c1 对象代表的年份设置为 2012 年，其他的所有数值会被重新计算

```
c1.set(Calendar.YEAR,2012);
```

其他字段属性 set 的意义以此类推

```
Calendar c1 = Calendar.getInstance();
```

//把 c1 对象的日期加上 10，也就是 c1 所表的日期的 10 天后的日期，其它所有的数值会被重新计算

```
c1.add(Calendar.DATE, 10);
```

//把 c1 对象的日期加上 10，也就是 c1 所表的日期的 10 天前的日期，其它所有的数值会被重新计算

```
c1.add(Calendar.DATE, -10);
```

其他字段属性的 add 的意义以此类推

b、Calendar 类对象信息的获得

```
Calendar c1 = Calendar.getInstance();// 获得年份
int year = c1.get(Calendar.YEAR);// 获得月份
int month = c1.get(Calendar.MONTH) + 1;// 获得日期
int date = c1.get(Calendar.DATE);// 获得小时
int hour = c1.get(Calendar.HOUR_OF_DAY);// 获得分钟
int minute = c1.get(Calendar.MINUTE);// 获得秒
int second = c1.get(Calendar.SECOND);// 获得星期几（注意（这个与 Date 类是不同的）：1 代表星期日、2 代表星期 1、3 代表星期二，以此类推）
int day = c1.get(Calendar.DAY_OF_WEEK);
```

4．其他注意的事项

(1) Java 是面向对象的语言，思考方法需要变换一下，里面的函数统称为方法，不要搞错。

(2) Java 里的数组有些变动，多维数组的内部其实都是指针，所以 Java 不支持 fill 多维数组。

数组定义后必须初始化，如 int[] a = new int[100];

(3) 布尔类型为 boolean，只有 true 和 false 二值，在 if (...) / while (...) 等语句的条件中必须为 boolean 类型。

在 C/C++中的 if (n % 2) ... 在 Java 中无法编译通过。

(4) 下面在 java.util 包里 Arrays 类的几个方法可替代 C/C++里的 memset、qsort/sort 和 bsearch:

```
Arrays.fill()
Arrays.sort()
Arrays.binarySearch()
```

对应练习：

课上例题对应

POJ1001、HDU1042、POJ3751

POJ 高精度计算题目

1131、1205、1220、1405、1503、1604 1894、2084、2305、2325、2389、2413、3101、3199