Java 经典问题算法大全

/*【程序 1】

题目：古典问题：有一对兔子，从出生后第 3 个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少?

1.程序分析： 兔子的规律为数列 1,1,2,3,5,8,13,21....

*/

```
package cn.com.flywater.FiftyAlgorthm;
public class FirstRabbit {
public static final int MONTH = 15;
public static void main(String[] args) {
   long f1 = 1L, f2 = 1L;
   long f;
   for(int i=3; i<MONTH; i++) {
    f = f2;
    f2 = f1 + f2;
    f1 = f;
    System.out.print("第" + i +"个月的兔子对数: ");
    System.out.println(" " + f2);
   }
}
}
```

/*【程序 2】

* 作者 若水飞天

题目：判断 101-200 之间有多少个素数，并输出所有素数。

1.程序分析：判断素数的方法：用一个数分别去除 2 到 sqrt(这个数)，如果能被整除，则表明此数不是素数，反之是素数。 */

```
package cn.com.flywater.FiftyAlgorthm;
public class SecondPrimeNumber {
public static int count = 0;
 public static void main(String[] args) {
     for(int i=101;i<200;i++){
         boolean b = true;//默认此数就素数
         for(int ,j=2;j<=Math.sqrt(i);j++){
             if(i%j==0){
                 b = false; //此数不是素数
                 break;
             }
         }
         if(b){
             count++;
             System.out.print(i+" ");
         }
     }
     System.out.println("\n 素数的个数: "+count);
    }
}
```

/*【程序 3】

作者 若水飞天
题目：打印出所有的"水仙花数(narcissus number)"，所谓"水仙花数"是指一个三位数，
其各位数字立方和等于该数本身。例如：153 是一个"水仙花数"，因为 153=1 的三次方＋5 的三次方＋3 的三次方。
1.程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。 */

```
package cn.com.flywater.FiftyAlgorthm;
public class ThirdNarcissusNum {
static int b, bb, bbb;
public static void main(String[] args) {

   for(int num=101; num<1000; num++) {
    ThirdNarcissusNum tnn = new ThirdNarcissusNum();
    tnn.f(num);
   }
}
public void f(int m) {
   bbb = m / 100;
   bb = (m % 100) / 10;
   b = (m % 100) % 10;
   if((bbb * bbb * bbb + bb * bb * bb + b * b * b) == m) {
    System.out.println(m);
   }
}
}
```

/*【程序 4】
作者 若水飞天
题目：将一个正整数分解质因数。例如：输入 90,打印出 90=2*3*3*5。
程序分析：对 n 进行分解质因数，应先找到一个最小的质数 k，然后按下述步骤完成:
(1)如果这个质数恰等于 n，则说明分解质因数的过程已经结束，打印出即可。
(2)如果 n>k，但 n 能被 k 整除，则应打印出 k 的值，并用 n 除以 k 的商,作为新的正整数你 n,重复执行第一步。
(3)如果 n 不能被 k 整除，则用 k+1 作为 k 的值,重复执行第一步。 */

```
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
public class FourthPrimeFactor {
static int n, k = 2;
public static void main(String[] args) {
   Scanner s = new Scanner(System.in);
   n = s.nextInt();
   System.out.print(n + "=" );
   FourthPrimeFactor fpf = new FourthPrimeFactor();
   fpf.f(n);
}
public void f(int n) {
   while(k <= n) {
    if(k == n) {
     System.out.println(n);
     break;
    } else if(n > k && n % k == 0) {
```

```
      System.out.print(k + "*");
      n = n / k;
      f(n);
      break;
     } else if(n > k && n % k != 0) {
      k++;
      f(n);
      break;
     }
    }
}

}
```

/*【程序 5】
作者 若水飞天
题目：利用条件运算符的嵌套来完成此题：学习成绩>=90 分的同学用 A 表示，60-89 分之间的用 B 表示，60 分以下的用 C 表示。
1.程序分析：(a>b)?a:b 这是条件运算符的基本例子。 */

```
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
public class FifthCondition {
//public static final int S1 = 90;
//public static final int S2 = 60;
static int grade;
public static void main(String[] args) {
   Scanner str = new Scanner(System.in);
   int s = str.nextInt();
   FifthCondition fc = new FifthCondition();
   grade = fc.compare(s);
   if(grade == 1) {
    System.out.print('A');
   } else if(grade == 2) {
    System.out.print('B');
   } else {
    System.out.println('C');
   }
}
public int compare(int s) {
   return s > 90 ? 1
     : s > 60 ? 2
     :3;
}
}
```

/*【程序 6】
作者 若水飞天
题目：输入两个正整数 m 和 n，求其最大公约数和最小公倍数。
1.程序分析：利用辗除法。 */

```
/*
```

* 在循环中，只要除数不等于0，用较大数除以较小的数，将小的一个数作为下一轮循环的大数，取得的余数作为下一轮循环的较小的数，如此循环直到较小的数的值为0，返回
* 较大的数，此数即为最小公约数，最小公倍数为两数之积除以最小公倍数。
* */

```
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class SixthCommonDiviser {
public static void main(String[] args) {
   int a, b;
   Scanner s1 = new Scanner(System.in);
   Scanner s2 = new Scanner(System.in);
   a = s1.nextInt();
   b = s2.nextInt();
   SixthCommonDiviser scd = new SixthCommonDiviser();
   int m = scd.division(a, b);
   int n = a * b / m;
   System.out.println("最大公约数: " + m);
   System.out.println("最小公倍数: " + n);
}

public int division(int x, int y) {
   int t;
   if(x < y) {
    t = x;
    x = y;
    y = t;
   }

   while(y != 0) {
    if(x == y) return 1;
    else {
     int k = x % y;
     x = y;
     y = k;
    }
   }
   return x;
}
}
```

/*【程序7】
作者 若水飞天
题目：输入一行字符，分别统计出其中英文字母、空格、数字和其它字符的个数。
1.程序分析：利用while语句,条件为输入的字符不为 '\n '. */

```
package cn.com.flywater.FiftyAlgorthm;
import java.util.*;
public class SeventhCharacterStatistics {
```

```
static int digital = 0;
static int character = 0;
static int other = 0;
static int blank = 0;
public static void main(String[] args) {
    char[] ch = null;
    Scanner sc = new Scanner(System.in);
    String s = sc.nextLine();
    ch = s.toCharArray();


    for(int i=0; i<ch.length; i++) {
     if(ch[i] >= '0' && ch[i] <= '9') {
      digital ++;
     } else if((ch[i] >= 'a' && ch[i] <= 'z') || ch[i] > 'A' && ch[i] <= 'Z') {
      character ++;
     } else if(ch[i] == ' ') {
      blank ++;
     } else {
      other ++;
     }

    }

    System.out.println("数字个数: " + digital);
    System.out.println("英文字母个数: " + character);
    System.out.println("空格个数: " + blank);
    System.out.println("其他字符个数:" + other );
}

}
```

/*【程序 8】
作者 若水飞天
题目：求 s=a+aa+aaa+aaaa+aa...a 的值，其中 a 是一个数字。例如 2+22+222+2222+22222(此时共有 5 个数相加)，几个数相加有键盘控制。
*/

```
/*
* 算法:  定义一个变量 b,  赋初值为 0; 定义一变量 sum,  赋初值为 0,
* 进入循环后，将 a + b 的值赋给 b，将 sum + b 的值赋给 sum;
* 同时，将 a 增加十倍,  ++ i;  继续循环;
* 循环结束后，输出 sum 的值。
*/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class EightPlus {
static long a = 2, b = 0;
public static void main(String[] args) {
```

```
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    int i = 0;
    long sum = 0;
    while(i < n) {
     b = b + a;
     sum = sum + b;
     a = a * 10;
     ++ i;
    }
    System.out.println("input number: " + n);
    System.out.println(sum);
}
}
```

/*【程序 9】
题目：一个数如果恰好等于它的因子之和，这个数就称为 "完数 "。例如 6=1＋2＋3.编程 找出 1000 以内的所有完数。
*/

```
package cn.com.flywater.FiftyAlgorthm;
public class NinthWanshu {

public static void main(String[] args) {

    System.out.println("1 到 1000 的完数有：  ");
    for(int i=1; i<1000; i++) {
     int t = 0;
     for(int j=1; j<= i/2; j++) {
      if(i % j == 0) {
       t = t + j;
      }
     }
     if(t == i) {
      System.out.print(i + " ");
     }
    }
}
}
```

/*【程序 10】
作者 若水飞天
题目：一球从 100 米高度自由落下，每次落地后反跳回原高度的一半；再落下，
求它在 第 10 次落地时，共经过多少米？第 10 次反弹多高？
*/

```
package cn.com.flywater.FiftyAlgorthm;
public class TenthTreeFall {
static double height = 100;
static double distance = 100;
```

```
public static void main(String[] args) {
    for(int i=1; i<10; i++) {
     distance = distance + height;
     height = height / 2;
    }

    System.out.println("路程: " + distance);
    System.out.println("高度: " + height / 2);
}
}
```

/*【程序 11】
* 作者 若水飞天
题目：有 1、2、3、4 个数字，能组成多少个互不相同且无重复数字的三位数？都是多少？
1.程序分析：可填在百位、十位、个位的数字都是 1、2、3、4。组成所有的排列后再去 掉不满足条件的排列。
*/
/*算法：3 个 for 循环加一个 if 语句;
*
*/

```
package cn.com.flywater.FiftyAlgorthm;
public class EleventhNumberRange {
public static void main(String[] args) {
    int count = 0;
    for(int x=1; x<5; x++) {
     for(int y=1; y<5; y++) {
      for(int z=1; z<5; z++) {
       if(x != y && y != z && x != z) {
        count ++;
        System.out.print(x*100 + y*10 + z + "   ");
        if(count % 4 == 0) {
         System.out.println();
        }
       }
      }
     }
    }
    System.out.println("共有" + count + "个三位数");
}
}
```

/*【程序 12】
* 作者 若水飞天
题目：企业发放的奖金根据利润提成。利润(I)低于或等于 10 万元时，奖金可提 10%;
利润高于 10 万元，低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分，
可可提成 7.5%；20 万到 40 万之间时，高于 20 万元的部分，
可提成 5%；40 万到 60 万之间时高于 40 万元的部分，可提成 3%;
60 万到 100 万之间时，高于 60 万元的部分，可提成 1.5%，高于 100 万元时，超过 100 万元的部分按 1%提成，
从键盘输入当月利润 I，求应发放奖金总数？

1.程序分析：请利用数轴来分界，定位。注意定义时需把奖金定义成长整型。
*/
/*注意： 要精确到小数点后多少位，用 DecimalFormat df = new DecimalFormat("#0.0000");
*
*/
```
package cn.com.flywater.FiftyAlgorthm;
import java.text.DecimalFormat;
import java.util.*;
public class TwelfthProfitAward {
static double profit = 0;
static double award = 0;
public static void main(String[] args) {
   Scanner s = new Scanner(System.in);
   profit = s.nextInt();
   System.out.println("输入的利润是" + profit + "万");
   if(profit > 0 && profit <= 10) {
    award = profit * 0.1;
   } else if(profit > 10 && profit <= 20) {
    award = 10 * 0.1 + (profit - 10) * 0.075;
   } else if(profit > 20 && profit <= 40) {
    award = 10 * 0.1 + 10 * 0.075 + (profit - 20) * 0.05;
   } else if(profit > 40 && profit <= 60) {
    award = 10 * 0.1 + 10 * 0.075 + 20 * 0.05 + (profit - 40) * 0.03;
   } else if(profit > 60 && profit <= 100) {
    award = 20 * 0.175 + 20 * 0.05 + 20 * 0.03 + (profit - 60) * 0.015;
   } else if(profit > 100) {
    award = 20 * 0.175 + 40 * 0.08 + 40 * 0.015 + (profit - 100) * 0.01;
   }
   DecimalFormat df = new DecimalFormat("#0.00000");

   System.out.println("应该提取的奖金是 " + df.format(award) + "万");
}
}
```

/*【程序13】
* 作者 若水飞天
题目：一个整数，它加上100后是一个完全平方数，再加上168又是一个完全平方数，请问该数是多少?
1.程序分析：在10万以内判断，先将该数加上100后再开方，再将该数加上268后再开方，
如果开方后的结果满足如下条件，即是结果。请看具体分析：
*/
```
package cn.com.flywater.FiftyAlgorthm;
public class ThirteenthTwiceSqrt {
public static void main(String[] args) {
   for(long l=1L; l<100000; l++) {
    if(Math.sqrt((long)(l+100)) % 1 == 0) {
     if(Math.sqrt((long)(l+268)) % 1 == 0) {
      System.out.println(l + "加100是一个完全平方数，再加168又是一个完全平方数");
     }
```

```
    }
   }
}
}

*【程序14】
* 作者 若水飞天
题目：输入某年某月某日，判断这一天是这一年的第几天?
1.程序分析：以3月5日为例，应该先把前两个月的加起来，
然后再加上5天即本年的第几天，特殊情况，闰年且输入月份大于3时需考虑多加一天。
*/
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
import .java.io.*;
public class FourteenthYearMonthDay {
public static void main(String[] args) {
   int year, month, day;
   int days = 0;
   int d = 0;

   FourteenthYearMonthDay fymd = new FourteenthYearMonthDay();

   System.out.print("Input the year:");
   year =fymd.input();
   System.out.print("Input the month:");
   month = fymd.input();
   System.out.print("Input The Day:");
   day = fymd.input();



   if (year < 0 ‖ month < 0 ‖ month > 12 ‖ day < 0 ‖ day > 31) {
    System.out.println("Input error, please run this program again!");
    System.exit(0);
   }
   for (int i=1; i <month; i++) {
    switch (i) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
     days = 31;
     //d += days;
     break;
    case 4:
    case 6:
```

```
     case 9:
     case 11:
      days = 30;
      //d += days;
      break;
     case 2:
      if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
       days = 29;
      } else {
       days = 28;
      }
      //d += days;
      break;
     }

     d += days;

    }
    System.out.println(year + ":" + month + ":" + day + "是今年的第" + (d+day) + "天。");
}
public int input() {
    int value = 0;
    Scanner s = new Scanner(System.in);
    value = s.nextInt();
    return value;
}
}

/*【程序15】
* 作者    若水飞天
题目：输入三个整数x,y,z，请把这三个数由小到大输出。
1.程序分析：我们想办法把最小的数放到x上，先将x与y进行比较，如果x> y则将x与y的值进行交换，然后再用x与z进行比较，如果x> z则将x与z的值进行交换，这样能使x最小。
*/
package cn.com.flywater.FiftyAlgorthm;
import java.util.*;
public class FifteenthNumberCompare {
public static void main(String[] args) {
    FifteenthNumberCompare fnc = new FifteenthNumberCompare();
    int a, b, c;

    System.out.println("Input 3 numbers:");
    a = fnc.input();
    b = fnc.input();
    c = fnc.input();
//
//    fnc.compare(a, b);//方法调用不能通过改变形参的值来改变实参的值
//    fnc.compare(b, c);// 这种做法是错的
```

```
//    fnc.compare(a, c);
    //System.out.println("result:" + a +" " + b + " " + c);// 没有改变

    if(a > b) {
     int t = a;
     a = b;
     b = t;
    }

    if(a > c) {
     int t = a;
     a = c;
     c = t;
    }

    if(b > c) {
     int t = b;
     b = c;
     c = t;
    }
    System.out.println( a + " " + b + " " + c);
}
public int input() {
    int value = 0;
    Scanner s = new Scanner(System.in);
    value = s.nextInt();
    return value;
}
public void compare(int x, int y) {//此方法没用
    if(x > y) {
     int t = x;
     x = y;
     y = t;
    }
}
}

/*【程序16】
* 作者 若水飞天
*题目：输出9*9口诀。
*1.程序分析：分行与列考虑，共9行9列，i控制行，j控制列。
**/
package cn.com.flywater.FiftyAlgorthm;
public class SixteenthMultiplicationTable {
public static void main(String[] args) {
    for(int i=1; i<10; i++) {
     for(int j=1; j<=i; j++) {
      System.out.print(j + "*" + i + "=" + j*i + " " );
```

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

//【程序17】
//作者 若水飞天
//题目：猴子吃桃问题：猴子第一天摘下若干个桃子，当即吃了一半，还不瘾，
//又多吃了一个 第二天早上又将剩下的桃子吃掉一半，又多吃了一个。
//以后每天早上都吃了前一天剩下 的一半零一个。到第10天早上想再吃时，
//见只剩下一个桃子了。求第一天共摘了多少。
//1.程序分析：采取逆向思维的方法，从后往前推断。

package cn.com.flywater.FiftyAlgorthm;
public class SeventeenthMonkeyPeach {
public static void main(String[] args) {
   int lastdayNum = 1;
   for(int i=2; i<=10; i++) {
    lastdayNum = (lastdayNum+1) * 2;
   }
   System.out.println("猴子第一天摘了 " + lastdayNum + " 个桃子");
}
}
/*【程序18】
* 作者 若水飞天
题目：两个乒乓球队进行比赛，各出三人。甲队为a,b,c三人，乙队为x,y,z三人。
已抽签决定比赛名单。有人向队员打听比赛的名单。a说他不和x比，c说他不和x,z比，请编程序找出三队赛手的名单。
*/
/*
* 这个程序写得很不好，是知道结果后拼凑起来的，还不如直接写输出语句加上结果来的好。
*/
package cn.com.flywater.FiftyAlgorthm;
public class EighteenthPingpong {
static char[] m = { 'a', 'b', 'c' };
static char[] n = { 'x', 'y', 'z' };
public static void main(String[] args) {
   for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < n.length; j++) {
     if (m[i] == 'a' && n[j] == 'x') {
      continue;
     } else if (m[i] == 'a' && n[j] == 'y') {
      continue;
     } else if ((m[i] == 'c' && n[j] == 'x')
       || (m[i] == 'c' && n[j] == 'z')) {
      continue;
     } else if ((m[i] == 'b' && n[j] == 'z')
```

```
        ‖ (m[i] == 'b' && n[j] == 'y')) {
       continue;
      } else
       System.out.println(m[i] + " vs " + n[j]);
     }
    }
}
}
```

题目：打印出如下图案（菱形）

```
    *
   ***
  *****
*******
  *****
   ***
    *
```

1.程序分析：先把图形分成两部分来看待，前四行一个规律，后三行一个规律，利用双重 for 循环，第一层控制行，第二层控制列。

```
*/
/*上半部分循环变量的控制方法是
* for(int i=0; i<(HEIGHT+1) / 2; i++) {
    for(int j=1; j<WIDTH/2-i; j++) {
    for(int k=1; k<(i+1)*2; k++) {

     下半部分循环变量的控制方法是
for(int i=1; i<=HEIGHT/2; i++) {
    for(int j=1; j<=i; j++) {
*     for(int k=1; k<=WIDTH-2*i-1; k++) {
*/
package cn.com.flywater.FiftyAlgorthm;
public class NineteenthPrintRhombic {
static final int HEIGHT = 7;
static final int WIDTH = 8;
public static void main(String[] args) {
    for(int i=0; i<(HEIGHT+1) / 2; i++) {
     for(int j=1; j<WIDTH/2-i; j++) {
      System.out.print(" ");
     }
     for(int k=1; k<(i+1)*2; k++) {
      System.out.print("*");
     }
     System.out.println();
    }

    for(int i=1; i<=HEIGHT/2; i++) {
     for(int j=1; j<=i; j++) {
      System.out.print(" ");
```

```
     }
     for(int k=1; k<=WIDTH-2*i-1; k++) {
      System.out.print("*");
     }
     System.out.println();
    }
}
}
```

上半部分第二重循环应改为：　　for(int j=0; j<WIDTH/2-i; j++)

上半部分第三重循环应改为：　　for(int k=1; k<=WIDTH-2*i; k++)

```
/*【程序 20】
* 作者 若水飞天
```

题目：有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13...求出这个数列的前 20 项之和。

1.程序分析：请抓住分子与分母的变化规律。

```
*/

package cn.com.flywater.FiftyAlgorthm;
import java.text.DecimalFormat;
public class TwentiethFractionSum {
public static void main(String[] args) {
   int x = 2, y = 1, t;
   double sum = 0;

   DecimalFormat df = new DecimalFormat("#0.0000");

   for(int i=1; i<=20; i++) {
    sum += (double)x / y;
    t = y;
    y = x;
    x = y + t;
    System.out.println("第 " + i + " 次相加，和是 " + df.format(sum));
   }
}

/*【程序 21】
* 作者 若水飞天
```

题目：求 1+2!+3!+...+20!的和

1.程序分析：此程序只是把累加变成了累乘。

```
*/
package cn.com.flywater.FiftyAlgorthm;
public class Twenty_firstFactorialSum {
static long sum = 0;
static long fac = 0;
public static void main(String[] args) {
   long sum = 0;
   long fac = 1;
   for(int i=1; i<=10; i++) {
    fac = fac * i;
```

```
     sum += fac;
    }
    System.out.println(sum);
}
}
/*【程序 22】
* 作者 若水飞天
题目：利用递归方法求 5!。
1.程序分析：递归公式：fn=fn_1*4!
*/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Twenty_secondFactorialRecursion {
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    Twenty_secondFactorialRecursion tfr = new Twenty_secondFactorialRecursion();
    System.out.println(tfr.recursion(n));
}
public long recursion(int n) {
    long value = 0 ;
    if(n ==1 || n == 0) {
     value = 1;
    } else if(n > 1) {
     value = n * recursion(n-1);
    }
    return value;
}
}
/*【程序 23】
* 作者 ： 若水飞天
题目：有 5 个人坐在一起，问第五个人多少岁？他说比第 4 个人大 2 岁。
问第 4 个人岁数，他说比第 3 个人大 2 岁。问第三个人，又说比第 2 人大两岁。
问第 2 个人，说比第一个人大两岁。最后问第一个人，他说是 10 岁。请问第五个人多大?
1.程序分析：利用递归的方法，递归分为回推和递推两个阶段。
要想知道第五个人岁数，需知道第四人的岁数，依次类推，推到第一人（10 岁），再往回推。
**/
package cn.com.flywater.FiftyAlgorthm;
public class Twenty_thirdPeopleAge {
public static void main(String[] args) {
    int age = 10;

    for(int i=2; i<=5; i++) {
     age += 2;
    }
    System.out.println(age);
}
}
```

```
/*【程序24】
* 作者 若水飞天
题目：给一个不多于5位的正整数，
要求：一、求它是几位数，二、逆序打印出各位数字。
说明： 这个算法实现虽然实现了这个功能，但不健壮，当输入字符是，会出现异常。
*/
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
public class Twenty_fourthNumber {

public static void main(String[] args) {

   Twenty_fourthNumber tn = new Twenty_fourthNumber();
   Scanner s = new Scanner(System.in);
   long a = s.nextLong();
   if(a < 0 || a > 100000) {
    System.out.println("Error Input, please run this program Again");
    System.exit(0);
   }

   if(a >=0 && a <=9) {
    System.out.println( a + "是一位数");
    System.out.println("按逆序输出是" + '\n' + a);
   } else if(a >= 10 && a <= 99) {
    System.out.println(a + "是二位数");
    System.out.println("按逆序输出是" );
    tn.converse(a);
   } else if(a >= 100 && a <= 999) {
    System.out.println(a + "是三位数");
    System.out.println("按逆序输出是" );
    tn.converse(a);
   } else if(a >= 1000 && a <= 9999) {
    System.out.println(a + "是四位数");
    System.out.println("按逆序输出是" );
    tn.converse(a);
   } else if(a >= 10000 && a <= 99999) {
    System.out.println(a + "是五位数");
    System.out.println("按逆序输出是" );
    tn.converse(a);
   }
}
public void converse(long l) {
   String s = Long.toString(l);
   char[] ch = s.toCharArray();
   for(int i=ch.length-1; i>=0; i--) {
    System.out.print(ch[i]);

   }
```

```
}
}
```
这个算法实在太土了，也许只有我若水飞天才会这样写，
下面写一个优雅一点的
```
import java.util.Scanner;
public class Twenty_fourthNumber {
public static void main(String[] args) {

   Twenty_fourthNumber tn = new Twenty_fourthNumber();
   Scanner s = new Scanner(System.in);
   long a = s.nextLong();
      String s = Long.toString(l);
   char[] ch = s.toCharArray();
System.out.println(a + "是" + ch.length + "位数");
   for(int i=ch.length-l; i>=0; i--) {
    System.out.print(ch[i]);
}
}
}
/*【程序 25】
* 作者 若水飞天
题目：一个 5 位数，判断它是不是回文数。即 12321 是回文数，个位与万位相同，十位与千位相同。
**/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Twenty_fifthPalindrom {
static int[] a = new int[5];
static int[] b = new int[5];
public static void main(String[] args) {

   boolean is =false;
   Scanner s = new Scanner(System.in);
   long l = s.nextLong();

   if (l > 99999 || l < 10000) {
    System.out.println("Input error, please input again!");
    l = s.nextLong();
   }

   for (int i = 4; i >= 0; i--) {
    a[i] = (int) (l / (long) Math.pow(10, i));
    l =(l % ( long) Math.pow(10, i));
   }
   System.out.println();
   for(int i=0,j=0; i<5; i++, j++) {
     b[j] = a[i];
   }
```

```
    for(int i=0,j=4; i<5; i++, j--) {
     if(a[i] != b[j]) {
      is = false;
      break;
     } else {
      is = true;
     }
    }
    if(is == false) {
     System.out.println("is not a Palindrom!");
    } else if(is == true) {
     System.out.println("is a Palindrom!");
    }
}
}
```

这个更好，不限位数

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("请输入一个正整数: ");
    long a = s.nextLong();
    String ss = Long.toString(a);
     char[] ch = ss.toCharArray();
     boolean is =true;
    int j=ch.length;
    for(int i=0; i<j/2; i++) {
    if(ch[i]!=ch[j-i-1]){is=false;}
    }
    if(is==true){System.out.println("这是一个回文数");}
      else {System.out.println("这不是一个回文数");}
     }
```

```
/*【程序 26】
* 作者    若水飞天
题目：请输入星期几的第一个字母来判断一下是星期几，
如果第一个字母一样，则继续 判断第二个字母。
1.程序分析：用情况语句比较好，如果第一个字母一样，
则判断用情况语句或 if 语句判断第二个字母。
此程序虽然实现了基本功能，但必须严格按照题目的要求输入，否则程序无法执行
**/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Twenty_sixthWeek {
Scanner s = new Scanner(System.in);
public static void main(String[] args) {
    Twenty_sixthWeek tw = new Twenty_sixthWeek();
    char ch = tw.getChar();
    switch(ch) {
```

```
     case 'M':
      System.out.println("Monday");
      break;
     case 'W':
      System.out.println("Wednesday");
      break;
     case 'F':
      System.out.println("Friday");
      break;
     case 'T': {
      System.out.println("please input the second letter!");
      char ch2 = tw.getChar();
      if(ch2 == 'U') {System.out.println("Tuesday"); }
      else if(ch2 == 'H') {System.out.println("Thursday"); }

     };
      break;
     case 'S': {
      System.out.println("please input the scecond letter!");
      char ch2 = tw.getChar();
      if(ch2 == 'U') {System.out.println("Sunday"); }
      else if(ch2 == 'A') {System.out.println("Saturday"); }

     };
      break;
    }
}
public char getChar() {
    String str = s.nextLine();
    char ch = str.charAt(0);
    if(ch<'A' || ch>'Z') {
     System.out.println("Input error, please input a capital letter");
     getChar();
    }
    return ch;
}
}
/*【程序 27】
* 作者 若水飞天
题目：求 100 之内的素数
1.程序分析：判断素数的方法：用一个数分别去除 2 到 sqrt(这个数)，
如果能被整除， 则表明此数不是素数，反之是素数。
**/

package cn.com.flywater.FiftyAlgorthm;
public class Twenty_seventhPrimeNumber {
public static void main(String[] args) {
    boolean b =false;
```

```
    int count = 0;
    for(int i=2; i<100; i+=1) {
     for(int j=2; j<=Math.sqrt(i); j++) {
      if(i % j == 0) {
       b = false;
       break;
      } else{
       b = true;
      }
     }

     if(b == true) {
      count ++;
      System.out.print(i + " ");
     }
    }
    System.out.println('\n' + "The number of PrimeNumber is :" + count);
}
}

/*【程序28】
* 作者 若水飞天
题目：对10个数进行排序
1.程序分析：可以利用选择法，即从后9个比较过程中，
选择一个最小的与第一个元素交换， 下次类推，
即用第二个元素与后8个进行比较，并进行交换。
**/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Twehty_eighthNumberSort {

public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int[] a = new int[10];
    for(int i=0; i<10; i++) {
     a[i] = s.nextInt();
    }
    for(int i=0; i<10; i++) {
     for(int j=i+1; j<10; j++) {
      if(a[i] > a[j]) {
       int t = a[i];
       a[i] = a[j];
       a[j] = t;
      }
     }
    }

    for(int i=0; i<10; i++) {
```

```
     System.out.print(a[i] + " ");
    }

}

}
```

/*【程序 29】
* 作者　　若水飞天
* 按程序分析，好像只是求主对角线的和
题目：求一个 3*3 矩阵对角线元素之和
1.程序分析：利用双重 for 循环控制输入二维数组，再将 a[i][i]累加后输出。
**/

```
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Twenty_ninthCrossSum {

public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int[][] a = new int[3][3];

    for(int i=0; i<3; i++) {
     for(int j=0; j<3; j++) {
      a[i][j] = s.nextInt();
     }
    }

    System.out.println("输入的 3 * 3 矩阵是:");
    for(int i=0; i<3; i++) {
     for(int j=0; j<3; j++) {
      System.out.print(a[i][j] + " ");
     }
     System.out.println();
    }

    int sum = 0;
    for(int i=0; i<3; i++) {
     for(int j=0; j<3; j++) {
      if(i == j) {
       sum += a[i][j];
      }
     }
    }
    System.out.println("对角线和是 " + sum);
}
}
```

/*【程序 30】
* 作者 若水飞天
题目：有一个已经排好序的数组。现输入一个数，要求按原来的规律将它插入数组中。

1. 程序分析：首先判断此数是否大于最后一个数，
然后再考虑插入中间的数的情况，插入后此元素之后的数，依次后移一个位置。
**/

```
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class ThirtiethInsert {
public static void main(String[] args) {

    int[] a = new int[]{1, 2, 3, 4, 5, 6, 7};
    int[] b = new int[a.length+1];
    int t1 =0, t2 = 0;
    int i =0;
    Scanner s= new Scanner(System.in);
    int num = s.nextInt();

    /*
    * 定义两个数组a，b，一个a的长度比另一个b大1， a看做是
    * 已经排好序的。
    * 接下来的过程是
    * 1: 如果num 比最后一个数大，把num赋值给数组b的最后一个数
    *     再按顺序把a 的每个元素赋给b
    * 2: 否则（num 不比a 的最后一个数大），
    *      如果a 的元素比num 小，则将这些元素按顺序赋给b
    *      将num 赋给比num大的b数组的元素，
    *      跳出第一个for循环。
    * 3: 定义一个循环控制变量，从num传给数组后num的下标值加一开始；
    *     直到b的结尾，将剩下的a 的值赋给b,赋值的过程是b[j] = a[i-1];
    *
    */

    if(num >= a[a.length-1]) {
     b[b.length-1] = num;
     for(i=0; i<a.length; i++) {
      b[i] = a[i];
     }
    } else {
     for(i=0; i<a.length; i++) {
      if(num >= a[i]) {
       b[i] = a[i];
      } else {
       b[i] = num;
       break;
      }
     }
     for(int j=i+1; j<b.length; j++) {
      b[j] = a[j-1];
     }
    }
```

```
    for (i = 0; i < b.length; i++) {
     System.out.print(b[i] + " ");
    }
}

}
/*【程序 21】
* 作者  若水飞天
题目：求 1+2!+3!+...+20!的和
1.程序分析：此程序只是把累加变成了累乘。
*/
package cn.com.flywater.FiftyAlgorthm;
public class Twenty_firstFactorialSum {
static long sum = 0;
static long fac = 0;
public static void main(String[] args) {
    long sum = 0;
    long fac = 1;
    for(int i=1; i<=10; i++) {
     fac = fac * i;
     sum += fac;
    }
    System.out.println(sum);
}
}
/*【程序 32】
* 作者    若水飞天
题目：取一个整数 a 从右端开始的 4～7 位。
程序分析：可以这样考虑：
(1)先使 a 右移 4 位。
(2)设置一个低 4 位全为 1,其余全为 0 的数。可用~(~0 < <4)
(3)将上面二者进行&运算。
**/
/*这个题我不会做，如有高手路过，还望指点
*
*/
package cn.com.flywater.FiftyAlgorthm;
public class Thirty_secondFS {
public static void main(String[] args) {

}
}
```

我没有用提示的方法，采用了字串截取。

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    boolean is =true;
```

```
    System.out.print("请输入一个 7 位以上的正整数：");
    long a = s.nextLong();
    String ss = Long.toString(a);
    char[] ch = ss.toCharArray();
    int j=ch.length;
    if (j<7){System.out.println("输入错误！");}
    else {
    System.out.println("截取从右端开始的 4～7 位是："+ch[j-7]+ch[j-6]+ch[j-5]+ch[j-4]);
    }
    }
```

【程序 33】

```
* 作者    若水飞天
```

题目：打印出杨辉三角形（要求打印出 10 行如下图）

1.程序分析：

```
         1
        1 1
       1 2 1
      1 3 3 1
     1 4 6 4 1
    1 5 10 10 5 1
*/
/*
* 网上千篇一律是这种写法，我也没有什么创新，
* a[i][j]=a[i-1][j]+a[i-1][j-1] 就是这个程序的核心
* 定义的是二维数组，为了使输出的结果看起来漂亮一点
* 可以用 for（int k=0; k<2*(10-i)-1; k++）控制输出的空格
* 这个循环是在控制行数的循环里面，控制列数的循环外面。
* 记得在输出菱形时为了控制下半部分的输出，在下拼命的写出
* for(int k=1; k<=WIDTH-2*i-1; k++) 才算了事。
*/
package cn.com.flywater.FiftyAlgorthm;
public class Thirty_thirdYangTriangle {
public static void main(String[] args) {

    int[][] a = new int[10][10];
    for(int i=0; i<10; i++) {
     a[i][i] = 1;
     a[i][0] = 1;
    }
    for(int i=2; i<10; i++) {
     for(int j=1; j<i; j++) {
      a[i][j] = a[i-1][j-1] + a[i-1][j];
     }
    }

    for(int i=0; i<10; i++) {
     for(int k=0; k<2*(10-i)-1; k++) {
      System.out.print(" ");
```

```
     }
     for(int j=0; j<=i; j++) {
      System.out.print(a[i][j] + "    ");
     }
     System.out.println();
    }
}
}
/*【程序 34】
* 作者      若水飞天
题目：输入 3 个数 a,b,c，按大小顺序输出。
1.程序分析：利用指针方法。
*/
/*
* 可惜，Java 好像没有指针
*/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Thirty_forthCompare {
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int a = s.nextInt();
    int b = s.nextInt();
    int c = s.nextInt();

    if(a < b) {
     int t = a;
     a = b;
     b = t;
    }

    if(a < c) {
     int t = a;
     a = c;
     c = t;
    }

    if(b < c) {
     int t = b;
     b = c;
     c = t;
    }

    System.out.println("从大到小的顺序输出:");
    System.out.println(a + " " + b + " " + c);
}

}
```

```
/*【程序35】
* 作者 若水飞天
题目：输入数组，最大的与第一个元素交换，最小的与最后一个元素交换，输出数组。
**/
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
public class Thirty_fifthSwop {
static final int N = 8;
public static void main(String[] args) {

   int[] a = new int [N];
   Scanner s = new Scanner(System.in);
   int index1 = 0, index2 = 0;

   System.out.println("please input numbers");
   for(int i=0; i<N; i++) {
    a[i] = s.nextInt();
    System.out.print(a[i] + " ");
   }

   int max =a[0], min = a[0];
   for(int i=0; i<a.length; i++) {
    if(a[i] > max) {
     max = a[i];
     index1 = i;
    }
    if(a[i] < min) {
     min = a[i];
     index2 = i;
    }
   }

   if(index1 != 0) {
    int temp = a[0];
    a[0] = a[index1];
    a[index1] = temp;
   }

   if(index2 != a.length-1) {
    int temp = a[a.length-1];
    a[a.length-1] = a[index2];
    a[index2] = temp;
   }
   System.out.println("after swop:");
   for(int i=0; i<a.length; i++) {
    System.out.print(a[i] + " ");
   }
}
```

```
}
/*【程序36】
* 作者      若水飞天
题目：有n个整数，使其前面各数顺序向后移m个位置，最后m个数变成最前面的m个数
**/
/*
* 这个题不知道有什么好办法，比较直接方法的是把这个数组分成两个数组，
* 再将两个数组合起来，但如果不控制好数组的下标，就会带来很多麻烦。
*/
package cn.com.flywater.FiftyAlgorthm;
import java.util.Scanner;
public class Thirty_sixthBackShift {
public static final int N =10;
public static void main(String[] args) {
    int[] a = new int[N];
    Scanner s = new Scanner(System.in);
    System.out.println("please input array a, ten numbers:");
    for(int i=0; i<a.length; i++) {
     a[i] = s.nextInt();
    }
    System.out.println("please input m , one number:");
    int m = s.nextInt();

    int[] b = new int[m];
    int[] c = new int[N-m];
    for(int i=0; i<m; i++) {
     b[i] = a[i];
    }

    for(int i=m,j=0; i<N; i++,j++) {
     c[j] = a[i];
    }

    for(int i=0; i<N-m; i++) {
     a[i] = c[i];
    }

    for(int i=m,j=0; i<N; i++,j++) {
     a[i] = b[j];
    }
    for(int i=0; i<a.length; i++) {
     System.out.print(a[i] + " ");
    }
}

}
/*【程序37】
* 作者 若水飞天
```

题目：有 n 个人围成一圈，顺序排号。从第一个人开始报数（从 1 到 3 报数），
凡报到 3 的人退出圈子，问最后留下的是原来第几号的那位。
**/
/*
* 这个程序是完全抄别人的
*/
```
package cn.com.flywater.FiftyAlgorthm;
import .java.util.Scanner;
public class Thirty_sevenCount3Quit {
public static void main(String[] args) {
   Scanner s = new Scanner(System.in);
   int n = s.nextInt();

   boolean[] arr = new boolean[n];
   for(int i=0; i<arr.length; i++) {
    arr[i] = true;//下标为 TRUE 时说明还在圈里
   }

   int leftCount = n;
   int countNum = 0;
   int index = 0;

   while(leftCount > 1) {
    if(arr[index] == true) {//当在圈里时
     countNum ++; //报数递加
     if(countNum == 3) {//报道 3 时
      countNum =0;//从零开始继续报数
      arr[index] = false;//此人退出圈子
      leftCount --;//剩余人数减一
     }
    }
    index ++;//每报一次数，下标加一

    if(index == n) {//是循环数数，当下标大于 n 时，说明已经数了一圈，
     index = 0;//将下标设为零重新开始。
    }
   }

   for(int i=0; i<n; i++) {
    if(arr[i] == true) {
     System.out.println(i);
    }
   }
}
}
```
/*【程序 38】
* 作者 若水飞天
题目：写一个函数，求一个字符串的长度，在 main 函数中输入字符串，并输出其长度。

```
*/
package cn.com.flywater.FiftyAlgorthm;
public class Thirty_eighthStringLength {
public static void main(String[] args) {
   String s = "jdfifdfhfhuififffdfggee";
   System.out.print("字符串的长度是: ");
   System.out.println(s.length());
}
}
```

*【程序 39】
* 作者　　若水飞天
题目：编写一个函数，输入 n 为偶数时，调用函数求 1/2+1/4+…+1/n,
当输入 n 为奇数时，调用函数 1/1+1/3+…+1/n(利用指针函数)
**/

```
package cn.com.flywater.FiftyAlgorthm;
import java.text.DecimalFormat;
import java.util.*;
public class Thirty_ninthFactionSum {
public static void main(String[] args) {
   Scanner s = new Scanner(System.in);
   int n = s.nextInt();
   DecimalFormat df = new DecimalFormat("#0.00000");

   System.out.println( n +" **** result " + df.format(sum(n)));

}
public static double sum(int n) {
   double result = 0;
   if(n % 2 == 0) {
    for(int i=2; i<=n; i+=2) {
     result += (double)1 / n;
    }
   } else {
    for(int i=1; i<=n; i+=2) {
     result += (double)1 / i ;
    }
   }
   return result;
}
}
```

/*【程序 40】
* 作者　　若水飞天
题目：字符串排序。
**/

```
package cn.com.flywater.FiftyAlgorthm;
public class FortiethStringSort {

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

```
    String temp = null;
    String[] s = new String[5];
    s[0] = "china".toLowerCase();
    s[1] = "apple".toLowerCase();
    s[2] = "MONEY".toLowerCase();
    s[3] = "BOOk".toLowerCase();
    s[4] = "yeah".toLowerCase();
    /*
    for(int i=0; i<s.length; i++) {
     for(int j=i+1; j<s.length; j++) {
      if(s[i].compareToIgnoreCase(s[j]) > 0) {
       temp = s[i];
       s[i] = s[j];
       s[j] = temp;
      }
     }
    }*/

    for(int i=0; i<s.length; i++) {
     for(int j=i+1; j<s.length; j++) {
      if(compare(s[i], s[j]) == false) {
       temp = s[i];
       s[i] = s[j];
       s[j] = temp;
      }
     }
    }

    for(int i=0; i<s.length; i++) {
     System.out.println(s[i]);
    }
}
static boolean compare(String s1, String s2) {
    boolean result = true;
    for(int i=0; i<s1.length() && i<s2.length(); i++) {
     if(s1.charAt(i) > s2.charAt(i)) {
      result = false;
      break;
     } else if(s1.charAt(i) <s2.charAt(i)) {
      result = true;
      break;
     } else {
      if(s1.length() < s2.length()) {
       result = true;
      } else {
       result = false;
      }
```

```
    }
   }
   return result;
}
}
```

LinkedList 类里面较重要的方法就是"addBefore(){}"和"private void remove(DNode <T> curr){}"
很多方法都与它俩有关系!!
下面的代码是个双向循环链表,在 LinkedList 里抄的...

```
package LinkedList;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.NoSuchElementException;
public class MyLinkedList<T> {
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    private DNode<T> header;
    private int listSize;
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    public MyLinkedList() {
        header = new DNode<T>();
        listSize = 0;
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    private static class DNode<T> {
        T nodeValue;
        DNode<T> prev;
        DNode<T> next;
        public DNode() { // for header
            nodeValue = null;
            prev = this; // left
            next = this; // right
        }
        public DNode(T item) {
            nodeValue = item;
            prev = this;
            next = this;
        }
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    public boolean isEmpty() {
        return (header.prev == header || header.next == header);
    }

    public int size() {
        return listSize;
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    private DNode<T> addBefore(DNode<T> curr, T item) {
```

```
        DNode<T> newNode, prevNode;
        newNode = new DNode<T>(item);

        prevNode = curr.prev;

        newNode.prev = prevNode;
        newNode.next = curr;

        prevNode.next = newNode;
        curr.prev = newNode;
        return newNode;
    }
    public boolean add(T item) {
        addBefore(header, item);
        listSize++;
        return true;
    }

    public void addFirst(T item) {
        addBefore(header.next, item);
        listSize++;
    }

    public void addLast(T item) {
        addBefore(header, item);
        listSize++;
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    private void remove(DNode<T> curr) {
        if(curr.next == curr) return;

        DNode<T> prevNode = curr.prev, nextNode = curr.next;

        prevNode.next = nextNode;
        nextNode.prev= prevNode;
    }

    public boolean remove(Object o) {
        for(DNode<T> p = header.next; p != header; p = p.next) {
            if(o.equals(p.nodeValue)) {
                remove(p);
                listSize--;
                return true;
            }
        }
        return false;
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
public void printList() {
    for(DNode<T> p = header.next; p != header; p = p.next)
        System.out.println(p.nodeValue);
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
private class MyIterator implements Iterator<T> {

    public DNode<T> nextNode = header.next;
    public DNode<T> lastReturned = header;

    public boolean hasNext() {
        return nextNode != header;
    }

    public T next() {
        if(nextNode == header)
            throw new NoSuchElementException("");

        lastReturned = nextNode;
        nextNode = nextNode.next;
        return lastReturned.nodeValue;
    }
    public void remove() {
        if(lastReturned == header)
            throw new IllegalStateException("");

        MyLinkedList.this.remove(lastReturned);
        lastReturned = header;
        listSize--;
    }
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
private class MyListIterator extends MyIterator implements ListIterator<T> {

    private int nextIndex;

    MyListIterator(int index) {
        if(index < 0 || index > listSize)
            throw new IndexOutOfBoundsException("");

        //如果index小于listSize/2，就从表头开始查找定位，否则就从表尾开始查找定位
        if(index < (listSize >> 1)) {
            nextNode = header.next;
            for(nextIndex = 0; nextIndex < index; nextIndex++)
                nextNode = nextNode.next;
        }else {
            nextNode = header;
            for(nextIndex = listSize; nextIndex > index; nextIndex--)
```

```
            nextNode = nextNode.prev;
        }
    }
    public boolean hasPrevious() {
        return nextIndex != 0;
        //return nextNode.prev != header;
    }

    public T previous() {
        if (nextIndex == 0)
            throw new NoSuchElementException("no");

        lastReturned = nextNode = nextNode.prev;
        nextIndex--;
        return lastReturned.nodeValue;
    }

    public void remove() {
        if(lastReturned == header)
            throw new IllegalStateException("");

        MyLinkedList.this.remove(lastReturned);
        nextIndex--;
        listSize--;

        if(lastReturned == nextNode)
            nextNode = nextNode.next;
        lastReturned = header;
    }

    public void add(T item) {
        MyLinkedList.this.addBefore(nextNode, item);
        nextIndex++;
        listSize++;
        lastReturned = header;
    }

    public void set(T item) {
         if (lastReturned == header)
             throw new IllegalStateException();

         lastReturned.nodeValue = item;
    }

    public int nextIndex() {
        return nextIndex;
    }
    public int previousIndex() {
```

```
            return nextIndex - 1;
        }
    }

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    public Iterator<T> iterator() {
        return new MyIterator();
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    public ListIterator<T> listIterator(int index) {
        return new MyListIterator(index);
    }
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    public static void main(String[] args) {
        MyLinkedList<String> t = new MyLinkedList<String>();
        t.add("A");
        t.add("B");
        t.add("C");
        t.add("D");
        //t.remove("B");
        //t.addFirst("AA");
        //t.addLast("BB");
        //t.printList();


        ListIterator<String> it = t.listIterator(t.size());

        while(it.hasPrevious()) {
            System.out.println(it.previous()); // D C B A
        }
    }
}// MyLinkedList end~
```

ArrayList 底层数组实现的,当实例化一个 ArrayList 是也相当实例化了一个数组
所以对元素的随即访问较快,而增加删除操作慢
LinkedList 底层实现是一个双向链表,没一个结点都包含了前一个元素的引用和后一个元素的引用和结点值
所以对元素的随即访问很慢,而增删较快
java 实现链表和 c 实现一样。 就是指针变成了引用。
【参考资料】JAVA 的链表(2009-05-11 01:35:49)标签：java 链表　 分类：学习资料
又是个不错的地方：http://blog.sina.com.cn/s/articlelist_1282789430_0_1.html

链表是一种重要的数据结构，在程序设计中占有很重要的地位。C 语言和 C＋＋语言中是用指针来实现链表结构的，由于 Java 语言不提供指针，所以有人认为在 Java 语言中不能实现链表，其实不然，Java 语言比 C 和 C＋＋更容易实现链表结构。Java 语言中的对象引用实际上是一个指针（本文中的指针均为概念上的意义，而非语言提供的数据类型），所以我们可以编写这样的类来实现链表中的结点。

```
    class Node
    {
    Object data;
```

```
Node next;//指向下一个结点
}
```

将数据域定义成 Object 类是因为 Object 类是广义超类，任何类对象都可以给其赋值，增加了代码的通用性。为了使链表可以被访问还需要定义一个表头，表头必须包含指向第一个结点的指针和指向当前结点的指针。为了便于在链表尾部增加结点，还可以增加一指向链表尾部的指针，另外还可以用一个域来表示链表的大小，当调用者想得到链表的大小时，不必遍历整个链表。下图是这种链表的示意图:

链表的数据结构

我们可以用类 List 来实现链表结构，用变量 Head、Tail、Length、Pointer 来实现表头。存储当前结点的指针时有一定的技巧，Pointer 并非存储指向当前结点的指针，而是存储指向它的前趋结点的指针,当其值为 null 时表示当前结点是第一个结点。那么为什么要这样做呢? 这是因为当删除当前结点后仍需保证剩下的结点构成链表，如果 Pointer 指向当前结点，则会给操作带来很大困难。那么如何得到当前结点呢，我们定义了一个方法 cursor(),返回值是指向当前结点的指针。类 List 还定义了一些方法来实现对链表的基本操作，通过运用这些基本操作我们可以对链表进行各种操作。例如 reset()方法使第一个结点成为当前结点。insert(Object d)方法在当前结点前插入一个结点，并使其成为当前结点。remove()方法删除当前结点同时返回其内容，并使其后继结点成为当前结点，如果删除的是最后一个结点，则第一个结点变为当前结点。

链表类 List 的源代码如下:

```
import java.io.*;
public class List
{

private Node Head=null;
private Node Tail=null;
private Node Pointer=null;
private int Length=0;
public void deleteAll()

{
Head=null;
Tail=null;
Pointer=null;
Length=0;
}
public void reset()

{
Pointer=null;
}
public boolean isEmpty()

{
return(Length==0);
}
public boolean isEnd()

{
if(Length==0)
  throw new java.lang.NullPointerException();
else if(Length==1)
```

```
  return true;
else
  return(cursor()==Tail);
}
public Object nextNode()

{
if(Length==1)
  throw new java.util.NoSuchElementException();
else if(Length==0)
  throw new java.lang.NullPointerException();
else
{
  Node temp=cursor();
  Pointer=temp;
  if(temp!=Tail)
    return(temp.next.data);
  else
    throw new java.util.NoSuchElementException();
}
}
public Object currentNode()

{
Node temp=cursor();
return temp.data;
}

public void insert(Object d)

{
Node e=new Node(d);
if(Length==0)
{
  Tail=e;
  Head=e;
}
else
{
  Node temp=cursor();
  e.next=temp;
  if(Pointer==null)
    Head=e;
  else
    Pointer.next=e;
}
Length＋＋;
}
```

```
public int size()

{
return (Length);
}
public Object remove()

{
Object temp;
if(Length==0)
  throw new java.util.NoSuchElementException();
else if(Length==1)
{
  temp=Head.data;
  deleteAll();
}
else
{
  Node cur=cursor();
  temp=cur.data;
  if(cur==Head)
    Head=cur.next;
  else if(cur==Tail)
  {
    Pointer.next=null;
    Tail=Pointer;
    reset();
  }
  else
    Pointer.next=cur.next;
    Length－－;
}
return temp;
}
private Node cursor()

{
if(Head==null)
  throw new java.lang.NullPointerException();
else if(Pointer==null)
  return Head;
else
  return Pointer.next;
}
public static void main(String[] args)

{
List a=new List ();
```

```
    for(int i=1;i<=10;i＋＋)
      a.insert(new Integer(i));
      System.out.println(a.currentNode());
      while(!a.isEnd())
        System.out.println(a.nextNode());
        a.reset();
        while(!a.isEnd())
        {
          a.remove();
        }
        a.remove();
        a.reset();
        if(a.isEmpty())
          System.out.println("There is no Node in List \n");
          System.in.println("You can press return to quit\n");
        try
        {
          System.in.read();
          //确保用户看清程序运行结果
        }
        catch(IOException e)
        {}
      }
    }
    class Node

    {
      Object data;
      Node next;
      Node(Object d)
      {
        data=d;
        next=null;
      }
    }
```

Java 算法程序题：

该公司笔试题就 1 个，要求在 10 分钟内作完。

题目如下：用 1、2、2、3、4、5 这六个数字，用 java 写一个 main 函数，打印出所有不同的排列，如：512234、412345 等，要求："4"不能在第三位，"3"与"5"不能相连。

Java 算法基本思路：

1 把问题归结为图结构的遍历问题。实际上 6 个数字就是六个结点，把六个结点连接成无向连通图，对于每一个结点求这个图形的遍历路径，所有结点的遍历路径就是最后对这 6 个数字的排列组合结果集。

2 显然这个结果集还未达到题目的要求。从以下几个方面考虑：

1. 3，5不能相连：实际要求这个连通图的结点3，5之间不能连通，可在构造图结构时就满足改条件，然后再遍历图。

2. 不能有重复：考虑到有两个2，明显会存在重复结果，可以把结果集放在TreeSet中过滤重复结果

3. 4不能在第三位：仍旧在结果集中去除满足此条件的结果。

采用二维数组定义图结构，最后的代码是：

```
1. import java.util.Iterator;
2. import java.util.TreeSet;
3.
4. public class TestQuestion {
5.
6. private String[] b = new String[]{"1", "2", "2", "3", "4", "5"};
7. private int n = b.length;
8. private boolean[] visited = new boolean[n];
9. visited =false;
10. private int[][] a = new int[n][n];
11. private String result = "";
12. private TreeSet TreeSet = new TreeSet();
13.
14. public static void main(String[] args) {
15. new TestQuestion().start();
16. }
17.
18. private void start() {
19. for (int i = 0; i < n; i++) {
20. for (int j = 0; j < n; j++) {
21. if (i == j) {
22. a[i][j] = 0;
23. } else {
24.     a[i][j] = 1;
25. }
26. }
27. }a[3][5] = 0;
28. a[5][3] = 0;
29. for (int i = 0; i < n; i++) {
30.     this.depthFirstSearch(i);
31. }
32. Iterator it = set.iterator();
33. while (it.hasNext()) {
34. String string = (String) it.next();
35.
36. if (string.indexOf("4") != 2) {
37. System.out.println(string);
```

```
38. }
39. }
40. }
41.
42. private void depthFirstSearch(int startIndex) {
43. visited[startIndex] = true;
44. resultresult = result + b[startIndex];
45. if (result.length() == n) {
46. TreeSet .add(result);
47. }
48. for(int j = 0; j < n; j++) {
49. if (a[startIndex][j] == 1 && visited[j] == false) {
50. depthFirstSearch(j);
51. } else {
52. continue;
53. }
54. }
55.     resultresult = result.substring(0, result.length() -1);
56.     visited[startIndex] = false;
57. }
58. }
59.
```

整数划分问题

正整数 n 的划分数 p（n）=q（n，n）

代码:

```
public static int q(int n,int m){
  if((n==1)‖(n<1)) return 0;
  if (n<m return q(n,n));
  if (n==m) return q(n,m-1)+1;
  return q(n,m-1)+q(n-m,m);
}
```

每个加数不能重复比如（4+5+5+6=20 5 和 5 重复不可以。1+19 和 19+1 也不可以 2+18 和 18+2 等等）