

Contents

1 Notes

1.1 Math	1
1.2 Array	1
1.3 ArrayList	1
1.4 ArrayDeque	1
1.5 BigDecimal	1
1.6 BigInteger	1
1.7 pascal triangle	2
1.8 Point2D	2
1.9 Split StringTokenizer	2
1.10 Str Contain	2
1.11 Stringbuilder	3
1.12 this	3
1.13 Rational	3
1.14 Exception	4
1.15 AbstractClass	4
1.16 Interface	4

1 Notes

1.1 Math

```

1  /* 三角函数 */
2  sin(double a)
3  cos(double a)
4  tan(double a)
5  Math.PI
6
7  /*四捨五入*/
8
9  // 無條件向上取整數
10 Math.ceil(2.1) = 3.0
11 // 無條件向下取整數
12 Math.floor(2.1) = 2.0
13 // 取最接近整數 如果一樣近就往偶數的那個取
14 Math rint(2.5) = 2.0
15 Math rint(-2.5) = -2.0
16
17 /* 取亂數 */
18 Math.random()

```

1.2 Array

```

1 #define inf 10000
2
3 // Arrays.fill 類似 C++ 的 memset( a, false,
4 // sizeof(a) );
5 Arrays.fill( prime, false );
6
7 // Array 自動排序
8 Arrays.sort(arr);
9
10 // System.arraycopy(來源, 起始索引, 目的, 起始索引,
11 // 複製長度)
12 System.arraycopy(arr1, 0, arr2, 0, arr1.length);

```

1.3 ArrayList

```

1 public static void removeDuplicate(ArrayList<Integer>
2     list){
3     for(int i = 0; i < list.size(); i++){
4         // 第一次出現此元素
5         int first = list.indexOf(list.get(i));
6         // 最後一次出現此元素
7         int last = list.lastIndexOf(list.get(i));
8
9         if(first == last){
10             list.remove(i);
11             i--;
12         }
13     }
14 }

```

```

11     }
12 }
13 }

```

1.4 ArrayDeque

```

1  / 宣告
2  Deque deque = new LinkedList<>();
3  Deque deque = new ArrayDeque<>();
4
5  // 在尾部加入元素
6  add(element)
7  addLast(element)
8  // 在頭部加入元素
9  addFirst(element)
10
11 //
12 // 在尾部加入元素，同時也返回一個 boolean 值來表示是否成功加入元
13 offer(element)
14 //
15 // 在頭部加入元素，同時也返回一個 boolean 值來表示是否成功加入元
16 offerFirst(element)
17 //
18 // 在尾部加入元素，同時也返回一個 boolean 值來表示是否成功加入元
19 offerLast(element):
20
21 // 返回 Deque 的所有元素
22 iterator()
23
24 // 以相反的排序方式返回 Deque 的所有元素
25 descendingIterator()
26
27 // 在頭部加入元素
28 push(element)
29 // 刪除在頭部元素，同時返回被刪除元素
30 pop(element)
31 // 刪除在頭部元素
32 removeFirst()
33 // 刪除在尾部元素
34 removeLast()
35
36 // 刪除在頭部元素，或者返回 Null 值假如是空的 Array
37 poll()
38 // 刪除在頭部元素，或者返回 Null 值假如是空的 Array
39 pollFirst()
40 // 刪除在尾部元素，或者返回 Null 值假如是空的 Array
41 pollLast():
42
43 // 搜索及返回在頭部元素，或者返回 Null 值假如是空的
44 Array
45 peek()
46 peekFirst()
47 // 搜索及返回在尾部元素，或者返回 Null 值假如是空的
48 Array
49 peekLast()

```

1.5 BigDecimal

```

1 // 利用 BigDecimal 去取小數點並四捨五入
2 BigDecimal ans = new BigDecimal(a);
3 System.out.println( ans.setScale(3,
4     RoundingMode.HALF_UP));

```

1.6 BigInteger

```

1 // BigInteger 有內建搜尋 nextProbablePrime
2 BigInteger n = new
3     BigInteger(String.valueOf(1000000));
4 for(int i = 0; i < 10; i++){

```

```

4   BigInteger ans = n.nextProbablePrime();
5   System.out.println(ans);
6   n = ans;
7 }

```

1.7 pascal triangle

```

1 import java.math.BigInteger;
2 import java.util.Arrays;
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9         int n = input.nextInt();
10
11         BigInteger[] triangle = new BigInteger[n+5];
12         Arrays.fill(triangle, BigInteger.ONE);
13
14         for( int i = 0; i <= n; i++ ){
15             for( int j = i; j > 0; j-- ){
16                 triangle[j] =
17                     triangle[j].add(triangle[j-1]);
18             }
19             for( int j = n-i; j > 0; j-- ){
20                 System.out.print(" ");
21             }
22             for( int j = 0; j <= i; j++ ){
23                 System.out.printf(" %d", triangle[j]);
24                 if( i % 2 == 0 && i != 0 && j == i ){
25                     break;
26                 }
27             }
28             if( i != 0 ){
29                 System.out.print(" 1");
30             }
31             System.out.println();
32         }
33 }

```

1.8 Point2D

```

1 import java.awt.geom.Point2D;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double x1 = 2.0, y1 = 6.0;
7         Point2D pt = new Point2D.Double(x1, y1);
8
9         System.out.println(pt.distance(0,0));
10    }
11 }
12
13 // Output:
14 // 6.324555320336759

```

1.9 Split StringTokenizer

```

1 /*
2 當字串中帶有 `+ * / \` 等符號時要寫 `\\`
3 轉義，因為他們在正則表達式中有相應的不同意義
4 最後放 0 的原因 -> limit is 0; array contains all
5 substrings
6 Positive Lookahead or Lookbehind 觀念
7 "((?=@)|(?<=0))" -> 才會把 運算子 還有 運算元
8 都單獨切開
9 */

```

```

7 String[] array =
8     str.split("((?=\\+|\\-|\\/|\\\\)/\\\\(|%)/(<?=\\+|\\-|\\/|\\\\)/\\\\)");
9
10 /*
11 預設 ans 賦值 可控制小數點位數 ex 0.000 ->
12 小數點後三位
13 */
14 BigDecimal ans = new BigDecimal("0.000");
15
16 /*
17 取運算子的部分用 StringTokenizer 更為方便
18 */
19 StringTokenizer st = new
20     StringTokenizer(str, "0123456789.");
21 boolean flag = true;
22 while(st.hasMoreTokens()){
23     System.out.print( flag ? st.nextToken() : " " +
24         st.nextToken());
25     flag = false;
26 }

```

1.10 Str Contain

```

1 boolean flag = true;
2 for( String check : array){
3
4     // string.contains 一次查詢多個關鍵字的寫法
5     // 要另外存一個數組 token
6     // 且在查詢時要以此 for ( String check : array )
7     // 去做掃描
8     if( !token.contains(check) ){
9
10         // 將字串型態的數字轉成實數並相加
11         BigDecimal b = new BigDecimal(check);
12
13         // (boolean 判斷式) ? (true的輸出) : (false
14         // 的輸出)
15         System.out.print( flag ? check : " " + check);
16         flag = false;
17
18         // BigDecimal 內建 .add 可以直接相加
19         ans = ans.add(b);
20     }
21 }
22
23 /*
24 找 BC 是 ABCD 的子字串
25 可以用 String.contains 找
26 */
27
28 import java.util.Scanner;
29
30 public class Main {
31
32     public static void main(String[] args) {
33         Scanner input = new Scanner(System.in);
34
35         String str = input.nextLine();
36
37         // 已確認：切割後的空白不會存進 array
38         String[] array = str.split(" ");
39
40         // 原來可以用 str.contains 去找子字串
41         // 我還以為只能找字元..
42         if(array[0].contains(array[1])){
43             System.out.println(array[1] + " is a
44                 substring of " + array[0]);
45         }
46         else{
47             System.out.println(array[1] + " is not a
48                 substring of " + array[0]);
49         }
50     }
51 }

```

46 }

1.11 StringBuilder

```

1 // 先把 i 轉成 String ( 因為這樣比較方便 reverse )
2 String str = String.valueOf(i);
3
4 // 使用 StringBuilder 宣告 rev -> 因為 StringBuilder
  才有 .reverse() 可以用
5 StringBuilder rev = new StringBuilder();
6
7 // 這裡 append (附加) str 的值給 rev
8 rev.append(str);
9
10 // reverse
11 rev.reverse();
12
13 /* 置換 */
14
15 // replace
16 import java.lang.*;
17
18 public class StringBuilderDemo {
19
20     public static void main(String[] args) {
21
22         StringBuilder str = new StringBuilder("Java Util
23             Package");
24         System.out.println("string = " + str);
25
26         // 從 index 5 到 9 換掉
27         str.replace(5, 9, "Lang");
28
29         System.out.println("After replacing: " + str);
30     }
31
32     // string = Java Util Package
33     // After replacing: Java Lang Package

```

1.12 this

```

1 /**
2  你看到在建構式參數與物件資料成員同名時，可用 this 加以區別
3  */
4
5 public class CashCard {
6
7     private String number;
8     private int balance;
9     private int bonus;
10
11     public CashCard(String number, int balance, int
12         bonus) {
13         this.number = number;
14         // 參數 number 指定給這個物件的 number
15
16         this.balance = balance;
17         // 參數 balance 指定給這個物件的 balance
18
19         this.bonus = bonus;
20         // 參數 bonus 指定給這個物件的 bonus
21     }
22     ...
23 }

```

1.13 Rational

```

1 static class Rational {
2

```

```

private int Molecular, Denominator;
4
5 // 設立 constructor 接收兩個整數參數為 分子/分母
  的初值
6 public Rational( int m, int d){
7     this.Molecular = m;
8     this.Denominator = d;
9 }
10
11 // 分別創立 Molecular、Denominator 的 get 和 set
   member methods
12 public int getMolecular() {
13     return this.Molecular;
14 }
15 public int getDenominator() {
16     return this.Denominator;
17 }
18 // set member methods -> 為了讓 private
   的參數可以使用的 member function
19 public int setMolecular(){
20     return this.Molecular;
21 }
22 public int setDenominator(){
23     return this.Denominator;
24 }
25
26 // 整數的四則運算
27 public Rational add( Rational rt2 ) {
28
29     int d = this.Denominator * rt2.Denominator;
30     int m = (( this.Molecular * ( d /
31         this.Denominator )) + ( rt2.Molecular * (
32         d / rt2.Denominator )));
33
34     BigInteger gcdn = BigInteger.valueOf(m).gcd(
35         BigInteger.valueOf(d) );
36     m = Integer.parseInt( String.valueOf(
37         BigInteger.valueOf(m).divide(gcdn) ) );
38     d = Integer.parseInt( String.valueOf(
39         BigInteger.valueOf(d).divide(gcdn) ) );
40
41     if( m == 0 || d == 0 ){
42         return new Rational(0,0);
43     }
44     else if( m < 0 && d < 0 ){
45         return new Rational( -m, -d );
46     }
47     return new Rational( m, d );
48 }
49
50 ...
51
52 }
53 public static void main(String[] args) {
54
55     ...
56
57     if( oper.equals("+") ){
58         // 要設立一個 Ration div 去接回傳回來的
59         Rational class 型態
60         // 之後取 Molecular 和 Denominator
61         都是用他去接
62         Rational div = rt1.add(rt2);
63         if( div.Molecular == 0 ){
64             System.out.println("0");
65         }
66         else if( div.Molecular % div.Denominator == 0
67             ){
68             System.out.println(div.Molecular /
69                 div.Denominator);
70         }
71         else{
72             System.out.println("(" + div.Molecular +
73                 "/" + div.Denominator + ")");
74         }
75     }
76 }

```

```

66
67     ...
68 }

```

1.14 Exception

```

1 // 宣告 Exception 類別 (除了變數名稱外 格式是固定的)
2 static class IllegalTriangleException extends
   Exception {
3 }
4 static class Triangle{
5     private static double side1, side2, side3;
6
7     // 在頭宣告 throws 內部依據要求不符者 throw
8     public Triangle(double side1, double side2,
9         double side3) throws IllegalTriangleException
10    {
11        if(side1 + side2 > side3 && side2 + side3 >
12            side1 && side1 + side3 > side2){
13            this.side1 = side1;
14            this.side2 = side2;
15            this.side3 = side3;
16        }
17        else{
18            throw new IllegalTriangleException();
19        }
20    }
21 // 一定要用 try 和 catch 接 必要時還會有 finally
22 try{
23     Triangle tri = new Triangle(side1, side2, side3);
24     System.out.printf("%.3f %.3f\n",
25         Triangle.getPerimeter(), Triangle.getArea());
26 }
27 // catch <Exception class 名稱>
28 // 這裡的 e 是自訂物件名稱
29 catch(IllegalTriangleException e){
30     System.out.println("The sum of any two sides is
31         greater than the other side");
32     System.out.printf("Side1: %.1f\nSide2:
33         %.1f\nSide3: %.1f\n", side1, side2, side3);
34     break;
35 }

```

1.15 AbstractClass

```

1 // 父類別
2 abstract class CShape{
3     protected String color;
4
5     public void setColor(String str){
6         color = str;
7     }
8     public abstract void show();
9 }
10
11 // 子類別
12 class CRectangle extends CShape{
13     protected int width,height;
14
15     public CRectangle(int w,int h){
16         width = w;
17         height = h;
18     }
19     // 定義繼承父類別的抽象函數 show()
20     public void show(){
21         System.out.print("color=" + color + ", ");
22         System.out.println("area=" + width*height);
23     }
24 }
25

```

```

26 // main
27 public class abstract_01{
28
29     public static void main(String args[]){
30
31         //透過子類別建立物件
32         CRectangle rect = new CRectangle(5,10);
33
34         // 利用子類別的物件去呼叫父類別的 member
35         // function setColor()
36         rect.setColor("Yellow");
37         rect.show();
38     }
39 }

```

1.16 Interface

```

1 /* 實作 */
2
3 interface CShape{
4     final double PI = 3.14;
5     public abstract void show();
6 }
7
8 class CCircle implementation CShape{
9
10     double radius;
11     public CCircle(double r){
12         radius = r;
13     }
14
15     //改寫介面中抽象函數
16     public void show(){
17         System.out.println("面積=" + PI*radius*radius);
18     }
19 }
20
21 public static void main(String args[]){
22
23     CCircle cir = new CCircle(2.0);
24     cir.show();
25 }
26
27 ---
28
29 /* 多重繼承*/
30
31 interface CShape{
32     final double PI = 3.14;
33     public abstract void show();
34 }
35
36 interface CColor{
37     abstract void setColor();
38 }
39
40 // CCircle 同時繼承了 CShape 和 CColor
41 class CCircle implements CShape,CColor{
42     double radius;
43     String color;
44
45     public CCircle(double r){
46         radius = r;
47     }
48
49     public void setColor(String str){
50         color = str;
51         System.out.println("color=" + color);
52     }
53     public void show(){
54         System.out.println("area=" + PI*radius*radius);
55     }
56 }
57
58 public class interface02
59 {

```

```

59     public static void main(String args[])
60     {
61         CCircle moon = new CCircle(5.0);
62         moon.setColor("yellow");
63         moon.show();
64     }
65 }
66 ---
67
68 /* interface extends */
69
70 // 父介面
71 interface CShape{
72     final double PI = 3.14;
73     abstract void setColor(String str);
74 }
75
76 // 子介面 繼承了 CShape
77 interface CShape2D extends CShape{
78     abstract void area();
79 }
80
81 // 實作
82 class CCircle implements CShape2D{
83     double radius;
84     String color;
85
86     public CCircle(double r){
87         radius = r;
88     }
89     public void setColor(String str){
90         color = str;
91         System.out.println("color=" + color);
92     }
93     public void show(){
94         System.out.println("area=" + PI*radius*radius);
95     }
96 }
97
98 ---
99
100 /* Comparable Interface */
101
102 class Student implements Comparable<Student>{
103     String name;
104     int age;
105
106     Student(String name, int age){
107         this.name = name;
108         this.age = age;
109     }
110
111     public int compareTo(Student st){
112         if(age == st.age){
113             return 0;
114         }
115         else if(age > st.age){
116             return 1;
117         }
118         else{
119             return -1;
120         }
121     }
122 }
123
124 // main
125 public class MainSort{
126     public static void main(String args[]){
127
128         ArrayList<Student> al = new
129             ArrayList<Student>();
130         al.add(new Student("Vijay",23));
131         al.add(new Student("Ajay",27));
132         al.add(new Student("Jai",21));
133
134         Collections.sort(al);
135
136         for(Student st: al){
137             System.out.println(st.name + " " +
138                 st.age);
139         }
140     }
141 }

```