Contents

1 Notes

1.1 Math

```
/* 三角函數 */
sin(double a)
cos(double a)
tan(double a)
Math.PI

/*四捨五入*/

// 無條件向上取整數
Math.ceil(2.1) = 3.0
// 無條件向下取整數
Math.floor(2.1) = 2.0
// 取最接近整數 如果一樣近就往偶數的那個取
Math.rint(2.5) = 2.0
Math.rint(-2.5) = -2.0

/* 取亂數 */
Math.random()
```

1.2 Array

```
#define inf 10000

// Arrays.fill 類似 C++ 的 memset( a, false, sizeof(a) );
Arrays.fill( prime,false );

// Array 自動排序
Arrays.sort(arr);

// System.arraycopy(來源, 起始索引, 目的, 起始索引, 複製長度)
System.arraycopy(arr1, 0, arr2, 0, arr1.length);
```

1.3 ArrayList

```
public static void removeDuplicate(ArrayList<Integer> list){
        for(int i = 0; i < list.size(); i++){
            // 第一次出現此元素
            int first = list.indexOf(list.get(i));
            // 最後一次出現此元素
            int last = list.lastIndexOf(list.get(i));

            if(first == last){
                list.remove(i);
                i--;
            }
        }
    }
```

1.4 ArrayDeque

```
/ 宣告
Deque deque = new LinkedList<>();
Deque deque = new ArrayDeque<>();

// 在尾部加入元素
add(element)
addLast(element)
// 在頭部加入元素
addFirst(element)

//     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
offer(element)
//     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
offerFirst(element)
//     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
offerLast(element):

// 返回 Deque 的所有元素
iterator()

// 以相反的排序方式返回 Deque 的所有元素
descendingIterator()

// 在頭部加入元素
push(element)
// 刪除在頭部元素，同時返回被刪除元素
pop(element)
// 刪除在頭部元素
removeFirst()
// 刪除在尾部元素
removeLast()

// 刪除在頭部元素，或者返回 Null 值假如是空的 Aarray
poll()
// 刪除在頭部元素，或者返回 Null 值假如是空的 Aarray
pollFirst()
// 刪除在尾部元素，或者返回 Null 值假如是空的 Aarray
pollLast():

// 搜索及返回在頭部元素，或者返回 Null 值假如是空的 Aarray
peek()
peekFirst()
// 搜索及返回在尾部元素，或者返回 Null 值假如是空的 Aarray
peekLast()
```

1.5 BigDecimal

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

1.6 BigInteger

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

```
    BigInteger ans = n.nextProbablePrime();
    System.out.println(ans);
    n = ans;
}
```

1.7 pascal triangle

```
import java.math.BigInteger;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        BigInteger[] triangle = new BigInteger[n+5];
        Arrays.fill(triangle, BigInteger.ONE);

        for( int i = 0; i <= n; i++ ){
            for( int j = i; j >0; j-- ){
                triangle[j] =
                    triangle[j].add(triangle[j-1]);
            }
            for( int j = n-i; j > 0; j-- ){
                System.out.print(" ");
            }
            for( int j = 0; j <= i; j++ ){
                System.out.printf(" %d", triangle[j]);
                if( i % 2 == 0 && i != 0 && j == i ){
                    break;
                }
            }
            if( i != 0 ){
                System.out.print(" 1");
            }
            System.out.println();
        }
    }
}
```

1.8 Point2D

```
import java.awt.geom.Point2D;

public class Main {

    public static void main(String[] args) {
        double x1 = 2.0, y1 = 6.0;
        Point2D pt = new Point2D.Double(x1, y1);

        System.out.println(pt.distance(0,0));
    }
}

// Output:
// 6.324555320336759
```

1.9 Split StringTokenizer

```
/*
當字串中帶有 `+ * | \` 等符號時要寫 `\\`
    轉義，因為他們在正則表達示中有相應的不同意義
最後放 0 的原因 -> limit is 0; array contains all
    substrings
Positive Lookahead or Lookbehind 觀念
“((?=@)|(?<=@))” -> 才會把 運算子 還有 運算元
    都單獨切開
*/
String[] array =
    str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*|/|\\)|\
/*
預設 ans 賦值 可控制小數點位數 ex 0.000 ->
    小數點後三位
*/
BigDecimal ans = new BigDecimal("0.000");

/*
取運算子的部分用 StringTokenizer 更為方便
*/
StringTokenizer st = new
    StringTokenizer(str,"0123456789.");
boolean flag = true;
while(st.hasMoreTokens()){
    System.out.print( flag ? st.nextToken() : " " +
        st.nextToken());
    flag = false;
}
```

1.10 Str Contain

```
boolean flag = true;
for( String check : array){

    // string.contains 一次查詢多個關鍵字的寫法
        要另外存一個數組 token
    // 且在查詢時要以此 for ( String check : array )
        去做掃描
    if( !token.contains(check) ){

        // 將字串型態的數字轉成實數並相加
        BigDecimal b = new BigDecimal(check);

        // (boolean 判斷式) ? (true的輸出) : (false
            的輸出)
        System.out.print( flag ? check : " " + check);
        flag = false;

        // BigDecimal 內建 .add 可以直接相加
        ans = ans.add(b);
    }
}

/*
找 BC 是 ABCD 的子字串
可以用 String.contains 找
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        String str = input.nextLine();

        // 已確認 : 切割後的空白不會存進 array
        String[] array = str.split(" ");

        // 原來可以用 str.contains 去找子字串
            我還以為只能找字元..
        if(array[0].contains(array[1])){
            System.out.println(array[1] + " is a
                substring of " + array[0]);
        }
        else{
            System.out.println(array[1] + " is not a
                substring of " + array[0]);
        }

    }
```

```
}
```

1.11 Stringbuilder

```
// 先把 i 轉成 String ( 因為這樣比較方便 reverse )
String str = String.valueOf(i);

// 使用 StringBuilder 宣告 rev -> 因為 StringBuilder 才有 .reverse() 可以用
StringBuilder rev = new StringBuilder();

// 這裡 append (附加) str 的值給 rev
rev.append(str);

// reverse
rev.reverse();

/* 置換 */

// replace
import java.lang.*;

public class StringBuilderDemo {

  public static void main(String[] args) {

    StringBuilder str = new StringBuilder("Java Util Package");
    System.out.println("string = " + str);

    // 從 index 5 到 9 換掉
    str.replace(5, 9, "Lang");

    System.out.println("After replacing: " + str);
  }
}

// string = Java Util Package
// After replacing: Java Lang Package
```

1.12 this

```
/**
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
*/

public class CashCard {

    private String number;
    private int balance;
    private int bonus;

    public CashCard(String number, int balance, int bonus) {
        this.number = number;
        // 參數number指定給這個物件的number

        this.balance = balance;
        // 參數balance指定給這個物件的balance

        this.bonus = bonus;
        // 參數bonus指定給這個物件的bonus
    }
    ...
}
```

1.13 Rational

```
static class Rational {

    private int Molecular, Denominator;

    // 設立 consturctor 接收兩個整數參數為 分子/分母的初值
    public Rational( int m, int d){
        this.Molecular = m;
        this.Denominator = d;
    }

    // 分別創立 Molecular、Denominator 的 get 和 set member methods
    public int getMolecular() {
        return this.Molecular;
    }
    public int getDenominator() {
        return this.Denominator;
    }
    // set member methods -> 為了讓 private 的參數可以使用的 member function
    public int setMolecular(){
        return this.Molecular;
    }
    public int setDenominator(){
        return this.Denominator;
    }

    // 整數的四則運算
    public Rational add( Rational rt2 ) {

        int d = this.Denominator * rt2.Denominator;
        int m = (( this.Molecular * ( d / this.Denominator )) + ( rt2.Molecular * ( d / rt2.Denominator )));

        BigInteger gcdn = BigInteger.valueOf(m).gcd( BigInteger.valueOf(d) );
        m =  Integer.parseInt( String.valueOf( BigInteger.valueOf(m).divide(gcdn) ) );
        d =  Integer.parseInt( String.valueOf( BigInteger.valueOf(d).divide(gcdn) ) );

        if( m == 0 || d == 0 ){
            return new Rational(0,0);
        }
        else if( m < 0 && d < 0 ){
            return new Rational( -m, -d );
        }
        return new Rational( m, d );
    }

                    ...

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

                ...

    if( oper.equals("+") ){
        // 要設立一個 Ration div 去接回傳回來的 Rational class 型態
        // 之後取 Molecular 和 Denominator 都是用他去接
        Rational div = rt1.add(rt2);
        if( div.Molecular == 0 ){
            System.out.println("0");
        }
        else if( div.Molecular % div.Denominator == 0 ){
            System.out.println(div.Molecular / div.Denominator);
        }
        else{
            System.out.println("(" + div.Molecular + "/ " + div.Denominator + ")");
        }
    }
```

```
                    ...
}
```

1.14 Exception

```
// 宣告 Exception 類別 (除了變數名稱外 格式是固定的
static class IllegalTriangleException extends
    Exception {
}
static class Triangle{
    private static double side1, side2, side3;

    // 在頭宣告 throws 內部依據要求不符者 throw
    public Triangle(double side1, double side2,
        double side3) throws IllegalTriangleException
        {
        if(side1 + side2 > side3 && side2 + side3 >
            side1 && side1 + side3 > side2){
            this.side1 = side1;
            this.side2 = side2;
            this.side3 = side3;
        }
        else{
            throw new IllegalTriangleException();
        }
    }
    ...
}

// 一定要用 try 和 catch 接 必要時還會有 finally
try{
    Triangle tri = new Triangle(side1, side2, side3);
    System.out.printf("%.3f %.3f\n",
        Triangle.getPerimeter(), Triangle.getArea());
}
// catch <Exception class 名稱>
// 這裡的 e 是自訂物件名稱
catch(IllegalTriangleException e){
    System.out.println("The sum of any two sides is
        greater than the other side");
    System.out.printf("Side1: %.1f\nSide2:
        %.1f\nSide3: %.1f\n", side1, side2, side3);
    break;
}
```

1.15 AbstractClass

```
// 父類別
abstract class CShape{
   protected String color;

   public void setColor(String str){
      color = str;
   }
   public abstract void show();
}

// 子類別
class CRectangle extends CShape{
   protected int width,height;

   public CRectangle(int w,int h){
      width = w;
      height = h;
   }
   // 定義繼承父類別的抽象函數 show()
   public void show(){
      System.out.print("color=" + color + ",  ");
      System.out.println("area=" + width*height);
   }
}

// main
public class abstract_01{

   public static void main(String args[]){

      //透過子類別建立物件
      CRectangle rect = new CRectangle(5,10);

      // 利用子類別的物件去呼叫父類別的 member
          function setColor()
      rect.setColor("Yellow");
      rect.show();
   }
}
```

1.16 Interface

```
/* 實作 */

interface CShape{
   final double PI = 3.14;
   public abstract void show();
}

class CCircle implementation CShape{

   double radius;
   public CCircle(double r){
     radius = r;
   }

   //改寫介面中抽象函數
   public void show(){
     System.out.println("面積＝" + PI*radius*radius);
   }
}

public static void main(String args[]){

      CCircle cir = new CCircle(2.0);
      cir.show();
}

---

/* 多重繼承*/

interface CShape{
   final double PI = 3.14;
   public abstract void show();
}
interface CColor{
  abstract void setColor();
}

// CCircle 同時繼承了 CShape 和 CColor
class CCircle implements CShape,CColor{
   double radius;
   String color;

   public CCircle(double r){
      radius = r;
   }

   public void setColor(String str){
      color = str;
      System.out.println("color=" + color);
   }
   public void show(){
      System.out.println("area=" + PI*radius*radius);
   }
}

public class interface02
{
```

```
    public static void main(String args[])
    {
      CCircle moon = new CCircle(5.0);
      moon.setColor("yellow");
      moon.show();
    }
}

---

/* interface extends */

// 父介面
interface CShape{
   final double PI = 3.14;
   abstract void setColor(String str);
}

// 子介面 繼承了 CShape
interface CShape2D extends CShape{
   abstract void area();
}

// 實作
class CCircle implements CShape2D{
   double radius;
   String color;

   public CCircle(double r){
      radius = r;
   }
   public void setColor(String str){
      color = str;
      System.out.println("color=" + color);
   }
   public void show(){
      System.out.println("area=" + PI*radius*radius);
   }
}

---

/* Comparable Interface */

class Student implements Comparable<Student>{
    String name;
    int age;

    Student(String name, int age){
        this.name = name;
        this.age = age;
    }

    public int compareTo(Student st){
        if(age == st.age){
            return 0;
        }
        else if(age > st.age){
            return 1;
        }
        else{
            return -1;
        }
    }
}

// main
public class MainSort{
    public static void main(String args[]){

        ArrayList<Student> al = new
            ArrayList<Student>();
        al.add(new Student("Vijay",23));
        al.add(new Student("Ajay",27));
        al.add(new Student("Jai",21));

        Collections.sort(al);
        for(Student st: al){
            System.out.println(st.name + " " +
                st.age);
        }
    }
}
```