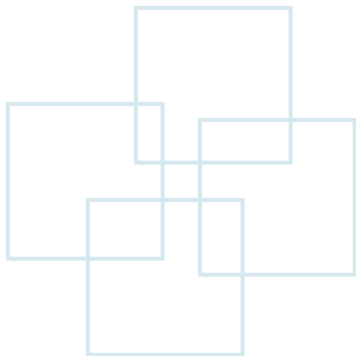


National University of Kaohsiung

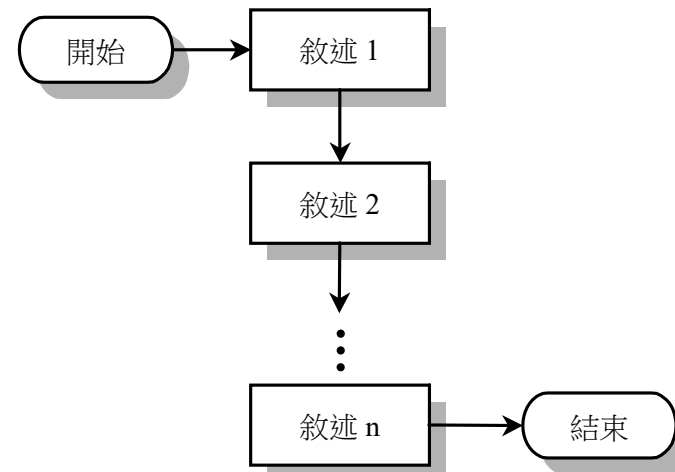
Java Programming Flow of Control

*Based on slides from the authors
Revised by Ya-Ju Yu 2017*

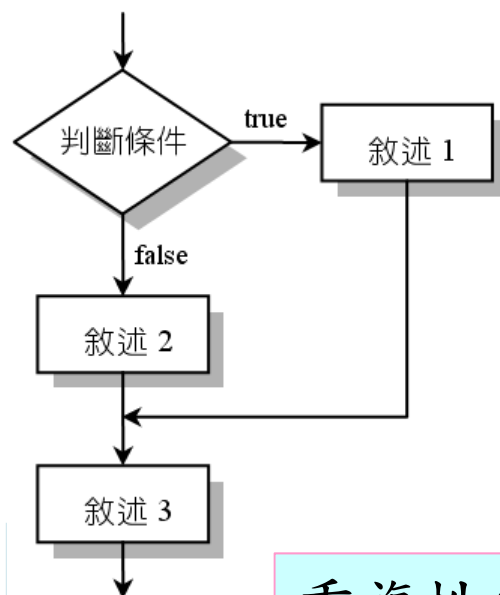


程式的結構

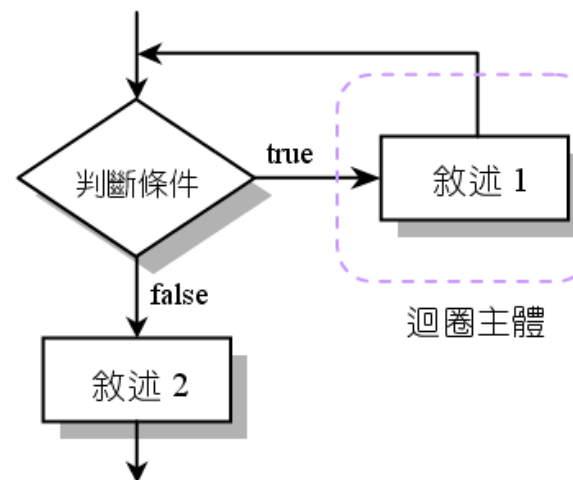
□循序性結構 (sequence structure)



□選擇性結構 (selection structure)



□重複性結構 (iteration structure)



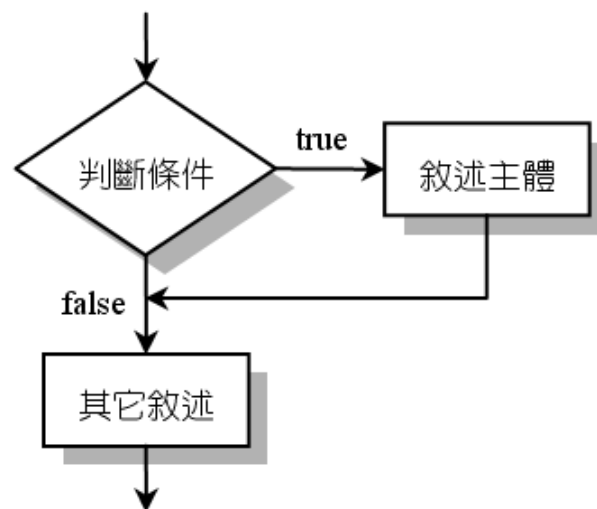
重複性結構有 for
、while 及 do
while 三種迴圈

if 敘述

□ 根據判斷的結果來執行不同的敘述

if 敘述的格式

```
if (判斷條件)  
{  
    敘述主體;  
}
```



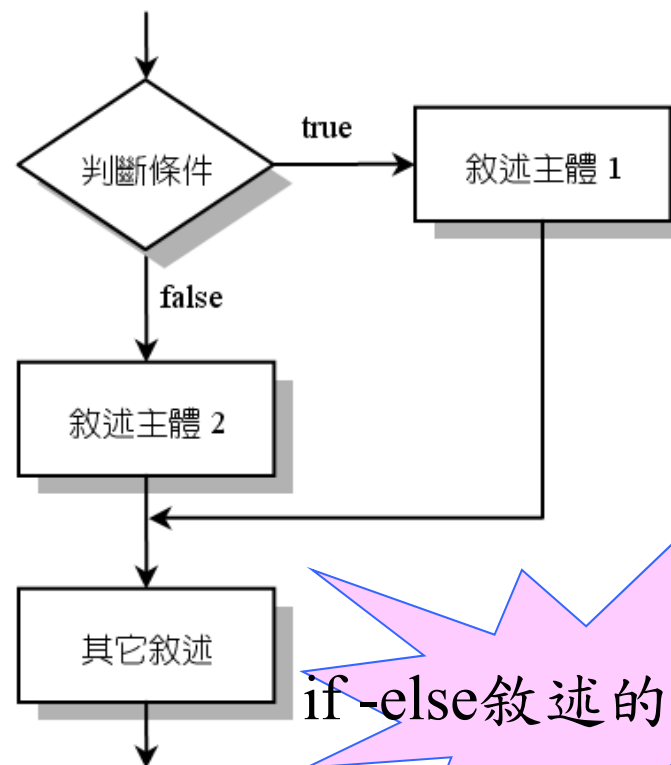
if 敘述的流程圖

if-else 敘述 (1/2)

□ if-else 敘述的格式與流程圖如下：

if-else 敘述的格式

```
if (判斷條件)
{
    敘述主體1;
}
else
{
    敘述主體2;
}
```



if-else 敘述的流程圖



if-else 敘述 (2/2)

▣ 下面的範例可用來判斷變數a是奇數或是偶數

```
01 // app5_1, if-else 敘述
02 public class app5_1
03 {
04     public static void main(String args[])
05     {
06         int a=15;
07
08         if (a%2==0)           // 如果可被 2 整除
09             System.out.println(a+" is an even number"); // 印出 a 為偶數
10         else
11             System.out.println(a+" is an odd number"); // 印出 a 為奇數
12     }
13 }
```

/* app5_1 OUTPUT-----

15 is an odd number

-----*/



National University of Kachsiung

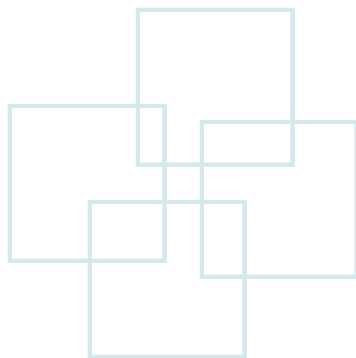
If Statement

❑ Do not use a string inequalities such as

– $\text{min} < \text{result} < \text{max}$

❑ You should use

– $(\text{min} < \text{result}) \ \&\& \ (\text{result} < \text{max})$





Pitfall: Using == with Strings

- ❑ The equality comparison operator (==) can correctly test two values of a *primitive* type
- ❑ However, when applied to two *objects* such as objects of the **String** class, == tests to see if they are stored in the same memory location, not whether or not they have the same value
- ❑ In order to test two strings to see if they have equal values, use the method **equals**, or **equalsIgnoreCase**

```
string1.equals(string2)
```

```
string1.equalsIgnoreCase(string2)
```



巢狀 if 敘述

□ if 敘述中又包含其它 if 敘述時，稱為巢狀 if 敘述 (nested if)

若判斷條件1成立，則執行這個部份

巢狀if敘述的格式

if (判斷條件1)

{

if (判斷條件2)

{

敘述主體;

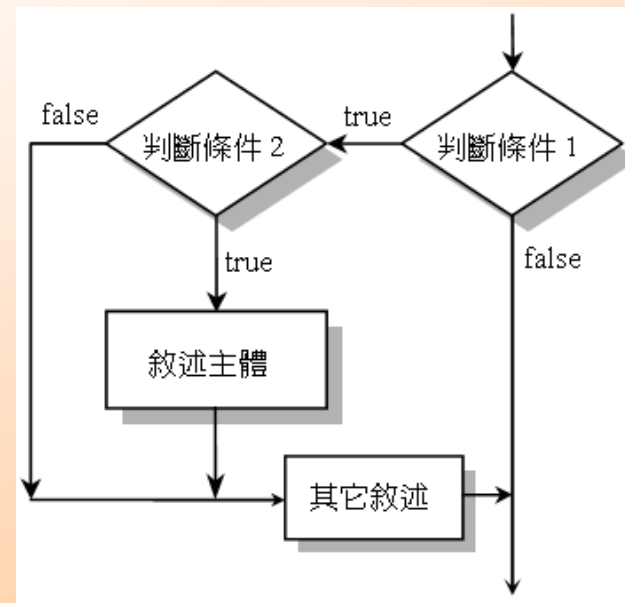
}

...

其它敘述;

}

若判斷條件2成立，則執行這個部份





Nested if Statements

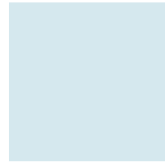
```
1  ...
2      if (score >= 90)
3          System.out.println("A");
4      else {
5          if (score >= 80)
6              System.out.println("B");
7          else {
8              if (score >= 70)
9                  System.out.println("C");
10             else {
11                 if (score >= 60)
12                     System.out.println("D");
13                 else
14                     System.out.println("F");
15             }
16         }
17     }
18  ...
```



Nested if Statements

- ❑ An if-elseif-else statement is a preferred format for multiple alternatives
- ❑ The performance may degrade due to the order of conditions.

```
1  ...
2      if (score >= 90)
3          System.out.println("A");
4      else if (score >= 90)
5          System.out.println("B");
6      else if (score >= 80)
7          System.out.println("C");
8      else if (score >= 70)
9          System.out.println("D");
10     else
11         System.out.println("F");
12  ...
```



Nested if Statements

- ❑ Terrible code snippet if the curly braces are ignored

```
1  ...
2      int i = 1, j = 2, k = 3;
3      if (i > j)
4          if (i > k)
5              System.out.println("Max = " + i);
6      else if (j > k)
7          System.out.println("Max = " + j);
8      else
9          System.out.println("Max = " + k);
10 ...
```



National University of Kaohsiung

條件運算子 (1/2)

□ 條件運算子的說明：

| 條件運算子 | 意義 |
|-------|--------------------------|
| ?: | 根據條件的成立與否，來決定結果為?或:後的運算式 |

□?: 的格式：

?:的敘述格式

變數 = 判斷條件 ? 運算式1 : 運算式2;



條件運算子 (2/2)

▣ 下面的程式可找出二數之間較大的數：

```
01 // app5_2, 條件運算子?:的使用
02 public class app5_2
03 {
04     public static void main(String args[])
05     {
06         int a=8,b=3,max;
07
08         max=(a>b)?a:b; // a>b 時,max=a, 否則 max=b
09
10         System.out.println("a="+a+", b="+b);
11         System.out.println(max+"是較大的數");
12     }
13 }
```

/ app5_2 OUTPUT----*
a=8, b=3
8 是較大的數
-----*/

for迴圈 (1/2)

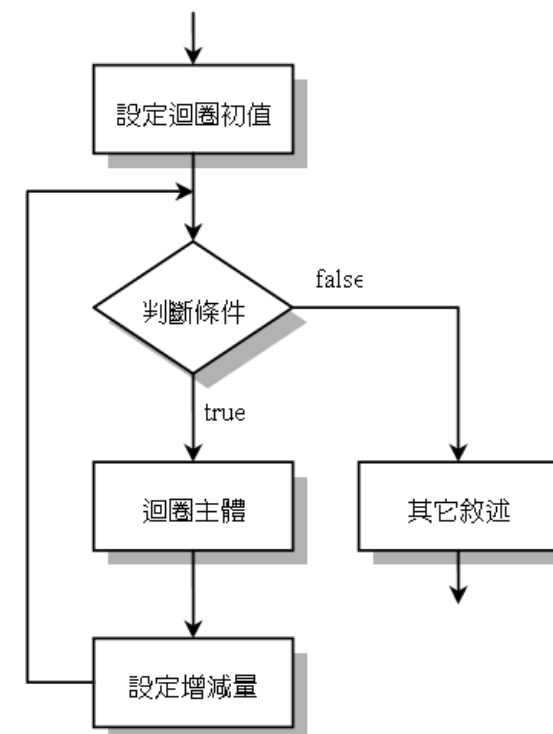
□ for迴圈的格式及執行流程：

for迴圈敘述格式

```
for (設定迴圈初值; 判斷條件; 設定增減量)
{
    迴圈主體;
}
```

這兒不可以加分號

這兒不可以加分號



1. 第一次進入 for 迴圈時，設定迴圈控制變數的起始值。
2. 根據判斷條件的內容，檢查是否要繼續執行迴圈，當條件判斷值為真（true），繼續執行迴圈主體；條件判斷值為假（false），則跳出迴圈執行其它敘述。
3. 執行完迴圈主體內的敘述後，迴圈控制變數會根據增減量的設定，更改迴圈控制變數的值，再回到步驟 2 重新判斷是否繼續執行迴圈。



for迴圈 (2/2)

▣ 下面的程式利用for迴圈計算 $1+2+\dots+10$ ：

```
01  // app5_3, for 迴圈
02  public class app5_3
03  {
04      public static void main(String args[])
05      {
06          int i, sum=0;
07
08          for(i=1; i<=10; i++)
09              sum+=i;    // 計算 sum=sum+i
10          System.out.println("1+2+...+10="+sum); // 印出結果
11      }
12  }
```

/* app5_3 OUTPUT---

1+2+...+10=55

-----*/



for迴圈裡的區域變數

□ 迴圈裡宣告的變數是區域變數（local variable），跳出迴圈，這個變數便不能再使用

□ for迴圈裡的區域變數使用範例：

```
01 // app5 4, 區域變數
02 public class app5 4
03 {
04     public static void main(String args[])
05     {
06         int sum=0;
07
08         for(int i=1;i<=5;i++)    // 在迴圈內宣告變數 i
09         {
10             sum=sum+i;
11             System.out.println("i="+i+", sum="+sum);
12         }
13     }
14 }
```

/* app5_4 OUTPUT---

```
i=1, sum=1
i=2, sum=3
i=3, sum=6
i=4, sum=10
i=5, sum=15
```

-----*/

} 變數 i 的有效範圍



National University of Kaohsiung

for迴圈裡的迴圈初值之設定

□正確的「設定迴圈初值」方式

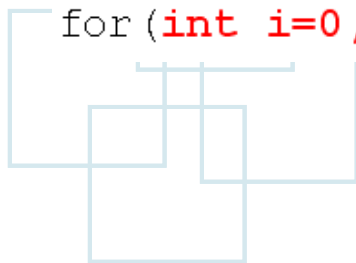
```
for (int i=0, j=0; 判斷條件; 設定增減量)    // 正確的「設定迴圈初值」方式
{ 迴圈主體 }
```

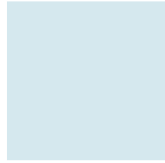
□錯誤的「設定迴圈初值」方式

```
for (int i=0, int j=0; 判斷條件; 設定增減量)    // 錯誤，關鍵字 int 只能出現一次
```

```
for (i=0, int j=0; 判斷條件; 設定增減量)    // 錯誤，int 要寫在第 1 個宣告變數之前
```

```
for (int i=0, short j=0; 判斷條件; 設定增減量)    // 錯誤，i 和 j 的型態必須相同
```



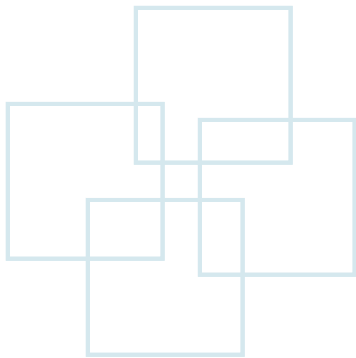


無窮迴圈

- 當迴圈控制變數使用不當，就有可能導致無窮迴圈
- $i < 0$ 永遠不會成立，將造成無窮迴圈

```
for (int i=1; i<0; )  
{  
    // 迴圈主體;  
}
```

不設定增減量將導致無窮迴圈





National University of Kaohsiung

Short-Circuit and Complete Evaluation

- Java can take a shortcut when the evaluation of the first part of a Boolean expression produces a result that evaluation of the second part cannot change
- This is called *short-circuit evaluation* or *lazy evaluation*
 - For example, when evaluating two Boolean subexpressions joined by `&&`, if the first subexpression evaluates to `false`, then the entire expression will evaluate to `false`, no matter the value of the second subexpression
 - In like manner, when evaluating two Boolean subexpressions joined by `||`, if the first subexpression evaluates to `true`, then the entire expression will evaluate to `true`





National University of Kaohsiung

Short-Circuit and Complete Evaluation

□ There are times when using short-circuit evaluation can prevent a *runtime error*

- In the following example, if the number of **kids** is equal to zero, then the second subexpression will not be evaluated, thus preventing a *divide by zero error*
- Note that reversing the order of the subexpressions will not prevent this

```
if ((kids !=0) && ((toys/kids) >=2)) . . .
```

□ Sometimes it is preferable to always evaluate both expressions, i.e., request complete evaluation

- In this case, use the **&** and **|** operators instead of **&&** and **||**

while 迴圈 (1/2)

while 迴圈敘述格式

設定迴圈初值；

while (判斷條件)

{

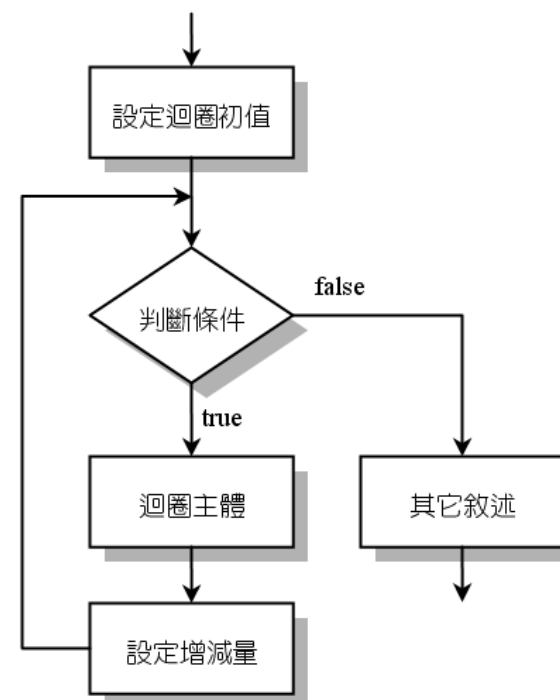
迴圈主體；

設定增減量；

}

這兒不可以加分號

這兒不可以加分號



1. 第一次進入 **while** 迴圈前，就必須先設定迴圈控制變數的起始值。
2. 根據判斷條件的內容，檢查是否要繼續執行迴圈，如果條件判斷值為 **true**，則繼續執行迴圈主體；如果條件判斷值為 **false**，則跳出迴圈執行後續的敘述。
3. 執行完迴圈主體內的敘述後，重新設定（增加或減少）迴圈控制變數的值，由於 **while** 迴圈不會主動更改迴圈控制變數的內容，所以在 **while** 迴圈中，設定迴圈控制變數的工作要由我們自己來做，再回到步驟 2 重新判斷是否繼續執行迴圈。

while 迴圈 (2/2)

□ 利用while迴圈找尋最大的n值

```
01 // app5_5, while 迴圈
02 public class app5_5
03 {
04     public static void main(String args[])
05     {
06         int n=0,sum=0;
07         while(sum<20)
08         {
09             System.out.println("n="+n+", sum="+sum);
10             n++;           // 將 n 值加 1
11             sum+=n;        // 累加計算
12         }
13     }
14 }
```

在程式設計的慣例上，會在確定迴圈次數時選擇for迴圈，而在不確定迴圈次數時選擇while迴圈，這樣的作法能讓語意更清楚的表達

```
/* app5_5 OUTPUT---
n=0, sum=0
n=1, sum=1
n=2, sum=3
n=3, sum=6
n=4, sum=10
n=5, sum=15
-----*/
```



do while 迴圈 (1/2)

□ do while 是用於迴圈執行的次數未知時

□ do while 至少會執行1次迴圈主體

do while 迴圈敘述格式

設定迴圈初值;

do

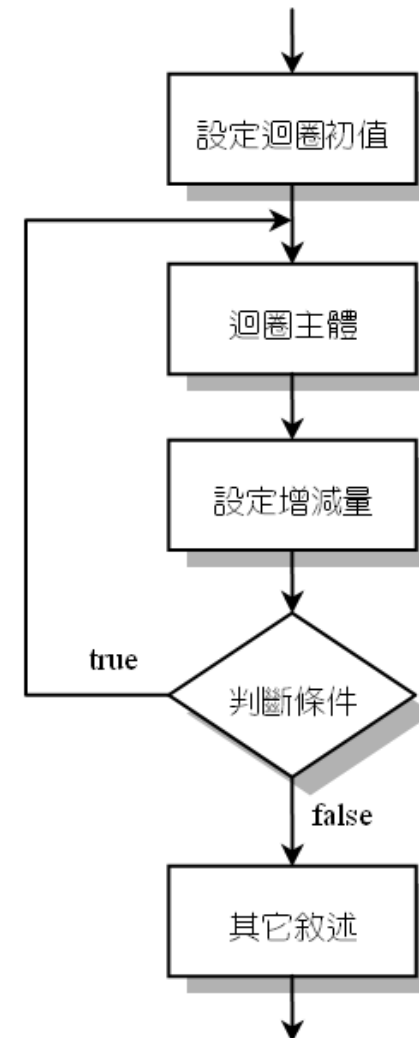
{

迴圈主體;

設定增減量;

} **while** (判斷條件);

要加分號





do while 迴圈 (2/2)

```
01 // app5_6, do while 迴圈
02 import java.util.Scanner;
03 public class app5_6
04 {
05     public static void main(String args[])
06     {
07         Scanner scn=new Scanner(System.in);
08         int n,i=1,sum=0;
09
10         do{
11             System.out.print("請輸入累加的最大值: ");
12             n=scn.nextInt();
13         }while(n<1); // 輸入 n, n 要大於等於 1, 否則會一直重複輸入
14
15         do
16             sum+=i++; // 計算 sum=sum+i, 然後 i 值再加 1
17         while(i<=n);
18
19         System.out.println("1+2+...+"+n+"="+sum); // 印出結果
20     }
21 }
```

/* app5_6 OUTPUT-----

請輸入累加的最大值: **-8**

請輸入累加的最大值: **10**

1+2+...+10=55

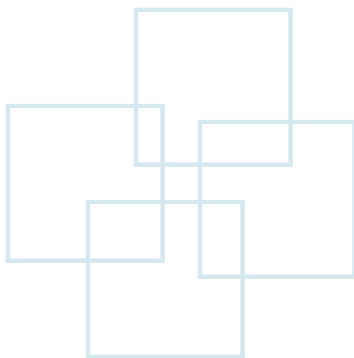
-----*/



While Loop

- ❑ When you use while loop, if you make a mistake and write your program so that the Boolean expression is always true, the loop will run forever.
 - Called an **infinite loop**

```
int number = 1;
while (number != 12)
{
    number = number + 2;
}
```



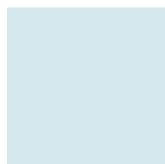


巢狀迴圈 (nested loops)

- 迴圈敘述中又有其它迴圈敘述時，稱為巢狀迴圈
- 以列印部份的九九乘法表為例，練習巢狀迴圈：

```
01 // app5_7, 巢狀 for 迴圈求 9*9 乘法表
02 public class app5_7
03 {
04     public static void main(String args[])
05     {
06         int i,j;
07
08         for (i=1;i<=3;i++)          // 外層迴圈
09         {
10             for (j=1;j<=3;j++)      // 內層迴圈
11                 System.out.print(i+"*"+j+"="+ (i*j)+"\t");
12             System.out.println();
13         }
14     }
15 }
```

```
/* app5_7 OUTPUT-----
1*1=1   1*2=2   1*3=3
2*1=2   2*2=4   2*3=6
3*1=3   3*2=6   3*3=9
-----*/
```



break敘述 (1/2)

□ break敘述格式：

break 敘述格式--for迴圈

for (初值設定; 判斷條件; 設定增減量)

{

敘述1;

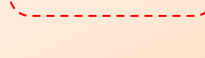
敘述2;

...

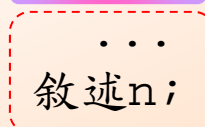
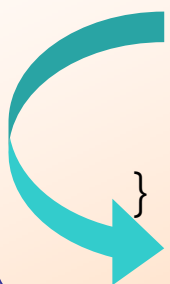


...

敘述n;



}



若執行break敘述，則此區塊內的敘述不會被執行



break敘述 (2/2)

□ 在for迴圈中使用break敘述的範例：

```
01  // app5_8, break 的使用
02  public class app5_8
03  {
04      public static void main(String args[])
05      {
06          int i;
07
08          for (i=1;i<=10;i++)
09          {
10              if(i%3==0)                // 判斷 i%3 是否為 0
11                  break;
12              System.out.println("i="+i);    // 印出 i 的值
13          }
14          System.out.println("when loop interrupted,i="+i);
15      }
16  }
```

/* app5_8 OUTPUT-----
i=1
i=2
when loop interrupted,i=3
-----*/



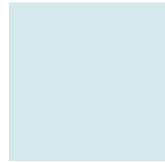
continue敘述 (1/2)

□ continue敘述會強迫程式跳到迴圈的起頭

□ continue敘述的格式：

continue 敘述格式--for迴圈





continue敘述 (2/2)

□ 使用continue敘述的範例：

```
01 // app5_9, continue 的使用
02 public class app5_9
03 {
04     public static void main(String args[])
05     {
06         int i;
07
08         for (i=1;i<=10;i++)
09         {
10             if(i%3==0)
11                 continue;
12             System.out.println("i="+i);
13         }
14         System.out.println("when loop interrupted,i="+i);
15     }
16 }
```

/* app5_9 OUTPUT-----

i=1
i=2
i=4
i=5
i=7
i=8
i=10
when loop interrupted,i=11
-----*/

// 判斷 i%3 是否為 0

// 印出 i 的值



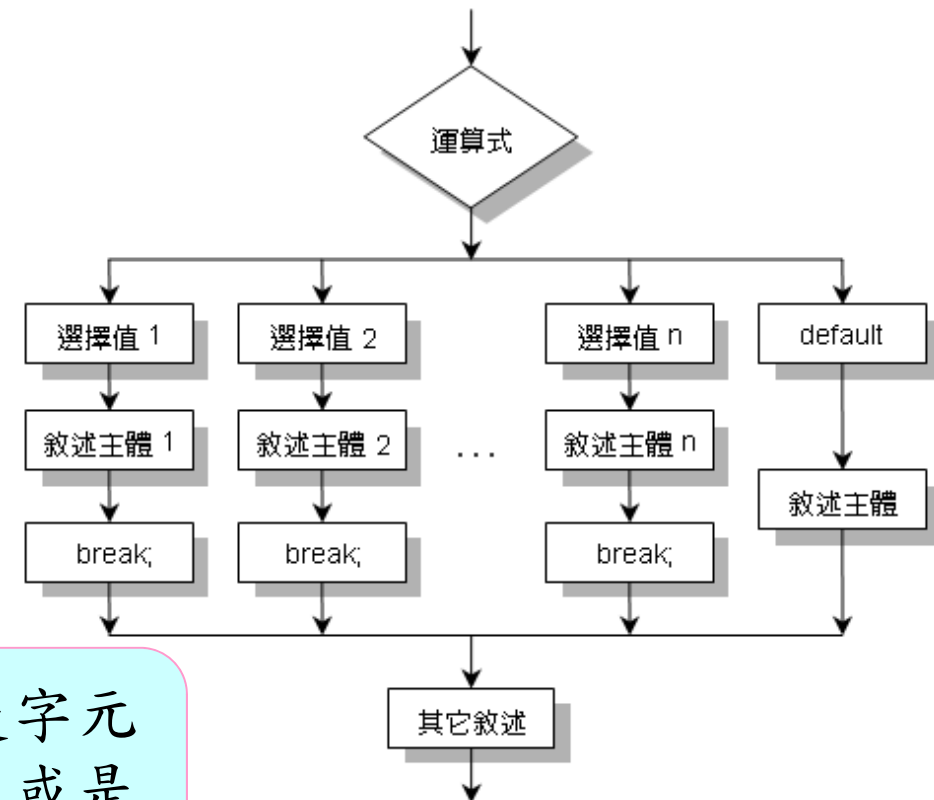
switch敘述 (1/2)

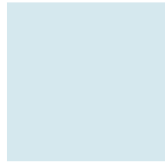
□ switch敘述可將多選一的情況簡化，格式如下：

switch 敘述格式

```
switch(運算式)
{
    case 選擇值1:
        敘述主體1;
        break;
    case 選擇值2:
        敘述主體2;
        break;
    ...
    case 選擇值n:
        敘述主體n;
        break;
    default:
        敘述主體;
}
```

可以是字元、字串或是整數





switch敘述 (2/2)

```
01 // app5_10, switch敘述
02 public class app5_10
03 {
04     public static void main(String args[])
05     {
06         int a=50,b=20;
07         char oper='*';
08
09         switch (oper)
10         {
11             case '+':          // 印出 a+b
12                 System.out.println(a+"+"+b+"="+ (a+b));
13                 break;
14             case '-':          // 印出 a-b
15                 System.out.println(a+"-"+b+"="+ (a-b));
16                 break;
17             case '*':          // 印出 a*b
18                 System.out.println(a+"*"+b+"="+ (a*b));
19                 break;
20             case '/':          // 印出 a/b
21                 System.out.println(a+"/"+b+"="+ ((float) a/b));
22                 break;
23             default:           // 印出字串
24                 System.out.println("Unknown expression!!");
25         }
26     }
27 }
```

```
/* app5_10 OUTPUT---
50*20=1000
-----*/
```

如果沒有在case敘述結尾處加上break，則會一直執行到switch敘述的尾端，才會離開switch敘述，如此將造成執行結果的錯誤



National University of Kachsiung

Exercises

□ Write a program to display the following results

*

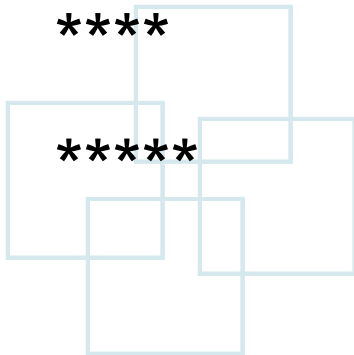
*

**

**

*

*





National University of Kaohsiung

Random Number Generation

```
import java.util.Random;
```

```
Random randomGenerator = new Random();
```

```
int r = randomGenerator.nextInt();
```

```
// all possible integers
```

```
int r = randomGenerator.nextInt(n);
```

```
// a random integer in the range from 0 to n-1
```

```
int r = randomGenerator.nextInt(3) + 4;
```

```
// 4, 5, or 6
```

```
int r = randomGenerator.nextDouble();
```

```
// To generate a random double
```



Math Class

- ❑ The Math class contains the methods needed to perform basic mathematical functions.
- ❑ The **Math** class provides common methods like: max, min, round, ceil, floor, abs, pow, exp, sqrt, log, log10, sin, cos, asin, acos, and random.
- ❑ Full document can be found <https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html>
- ❑ **import java.lang.Math;**



National University of Kachsiung

Example

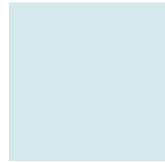
□ Password generator

□ Write a program which generates ten characters as a password. There may be lower-case letters, upper-case letters, and digital characters in the character sequence.





```
1  ...
2  public static void main (String[] args) {
3      for (int i = 1; i <= 10; ++i){
4          int type = (int) (Math.random() * 3);
5          switch (type) {
6              case 0:
7                  System.out.printf("%c",
8                      getRandomUpperCaseLetter());
9                  break;
10             case 1:
11                 System.out.printf("%c",
12                     getRandomLowerCaseLetter());
13                 break;
14             case 2:
15                 System.out.printf("%c",
16                     getRandomDigitalCharacter());
17                 break;
18             }
19         }
20     }
21 }
22
23 random()
24 Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
```



Example

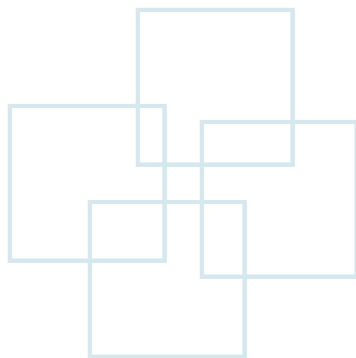
```
1  ...
2      static char getRandomUpperCaseLetter() {
3          return (char) (Math.random() * ('Z' - 'A' + 1) + 'A');
4      }
5
6      static char getRandomLowerCaseLetter() {
7          return (char) (Math.random() * ('z' - 'a' + 1) + 'a');
8      }
9
10     static char getRandomDigitalCharacter() {
11         return (char) (Math.random() * ('9' - '0' + 1) + '0');
12     }
13  ...
```



National University of Kachsiung

Exercises

- ❑ Write a program which sums two random integers and lets the user repeatedly enter a new answer until it is correct.



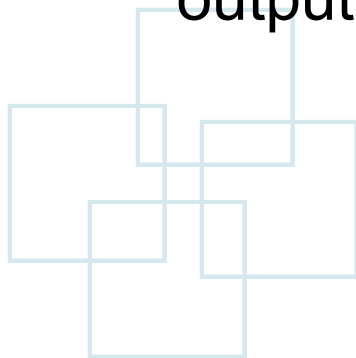


Debugging

❑ Insert `system.out.println()` in your code where you would like to observe.

❑ Assertion Checks

- If the Boolean evaluates to true, nothing happens, but if the Boolean evaluates to false, the problem ends and outputs an error message.



```
int n = 1;  
assert (n == 0);
```




National University of Kaohsiung

Debugging-Assertion Checks

▣如要在Eclipse中使用assertion機制，請作以下設定：

設定一：(編譯設定)

Windows->Preference->Java->Compiler-> 頁面。

將..JDK Compliance level->Compiler compliance level調成1.4以上。

設定二：(執行設定)

Run->Run Congration->(x)=Arguments頁面，在VM arguments加入-**ea**參數，按下

Run button便可看到啟動assertion後的執行結果。



National University of Kachsiung

Debugging Example (1 of 9)

- ❑ The following code is supposed to present a menu and get user input until either 'a' or 'b' is entered.

```
String s = "";
char c = ' ';
Scanner keyboard = new Scanner(System.in);

do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s.toLowerCase();
    c = s.substring(0,1);
}
while ((c != 'a') || (c != 'b'));
```



Debugging Example (2 of 9)

Result: Syntax error:

```
c = s.substring(0,1);      : incompatible types  
found:   java.lang.String  
required: char
```

- ❑ Using the “random change” debugging technique we might try to change the data type of `c` to `String`, to make the types match
- ❑ This results in more errors since the rest of the code treats `c` like a `char`



Debugging Example (3 of 9)

- ❑ First problem: substring returns a String, use charAt to get the first character:

```
String s = "";
char c = ' ';
Scanner keyboard = new Scanner(System.in);

do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s.toLowerCase();
    c = s.charAt(0);
}
while ((c != 'a') || (c != 'b'));
```

Now the program compiles, but it is stuck in an infinite loop. Employ tracing:



National University of Kaohsiung

Debugging Example (4 of 9)

```
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    System.out.println("String s = " + s);
    s.toLowerCase();
    System.out.println("Lowercase s = " + s);
    c = s.charAt(0);
    System.out.println("c = " + c);
}
while ((c != 'a') || (c != 'b'));
```

Sample output:

```
Enter 'A' for option A or 'B' for option B.
A
String s = A
Lowercase s = A
c = A
Enter 'A' for option A or 'B' for option B.
```

From tracing we can see that the string is never changed to lowercase. Reassign the lowercase string back to s.



National University of Kachsiung

Debugging Example (5 of 9)

- ❑ The following code is supposed to present a menu and get user input until either 'a' or 'b' is entered.

```
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s = s.toLowerCase();
    c = s.charAt(0);
}
while ((c != 'a') || (c != 'b'));
```

However, it's still stuck in an infinite loop. What to try next?





Debugging Example (6 of 9)

❑ Could try the following “patch”

```
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s = s.toLowerCase();
    c = s.charAt(0);
    if ( c == 'a')
        break;
    if (c == 'b')
        break;
}
while ((c != 'a') || (c != 'b'));
```

This works, but it is ugly! Considered a coding atrocity, it doesn't fix the underlying problem. The boolean condition after the while loop has also become meaningless. Try more tracing:



Debugging Example (7 of 9)

```
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s = s.toLowerCase();
    c = s.charAt(0);
    System.out.println("c != 'a' is " + (c != 'a'));
    System.out.println("c != 'b' is " + (c != 'b'));
    System.out.println("(c != 'a') || (c != 'b') is "
        + ((c != 'a') || (c != 'b')));
}
while ((c != 'a') || (c != 'b'));
```

Sample output:

Enter 'A' for option A or 'B' for option B.

A

c != 'a' is false

c != 'b' is true

(c != 'a') || (c != 'b') is true

From the trace we can see that the loop's boolean expression is true because c cannot be not equal to 'a' and not equal to 'b' at the same time.



National University of Kaohsiung

Debugging Example (8 of 9)

❑ Fix: We use `&&` instead of `||`

```
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s = s.toLowerCase();
    c = s.charAt(0);
}
while ((c != 'a') && (c != 'b'));
```





Debugging Example (9 of 9)

- ❑ Even better: Declare a boolean variable to control the do-while loop. This makes it clear when the loop exits if we pick a meaningful variable name.

```
boolean invalidKey;
do
{
    System.out.println("Enter 'A' for option A or 'B' for option B.");
    s = keyboard.next();
    s = s.toLowerCase();
    c = s.charAt(0);
    if (c == 'a')
        invalidKey = false;
    else if (c == 'b')
        invalidKey = false;
    else
        invalidKey = true;
}
while (invalidKey);
```