Java

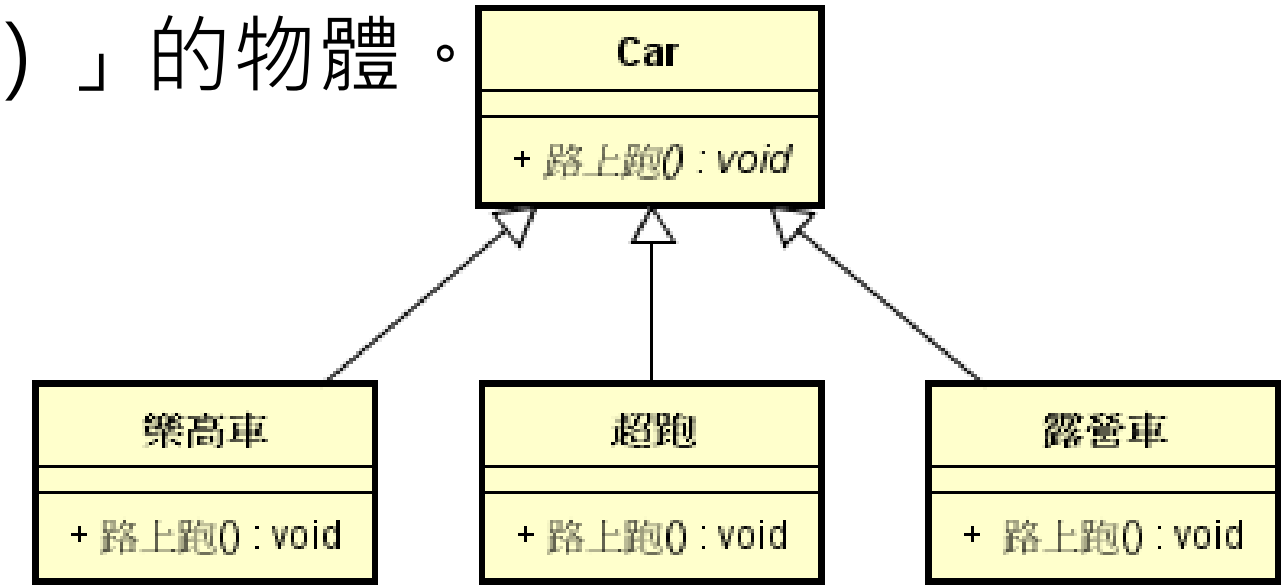
學生一：范真瑋
學生二：陳奕元
學生三：林建業

Java簡介

物件導向、跨平台性、自動垃圾回收

物件導向

- 程式碼和資料的實際集合體叫做「物件」。
- 一個物件可以想像成繫結了很多「行為（程式碼）」和「狀態（資料）」的物體。



跨平台性

- 「一次編譯，到處執行」。
- 必須安裝Java執行環境（Java Runtime Environment，JRE）

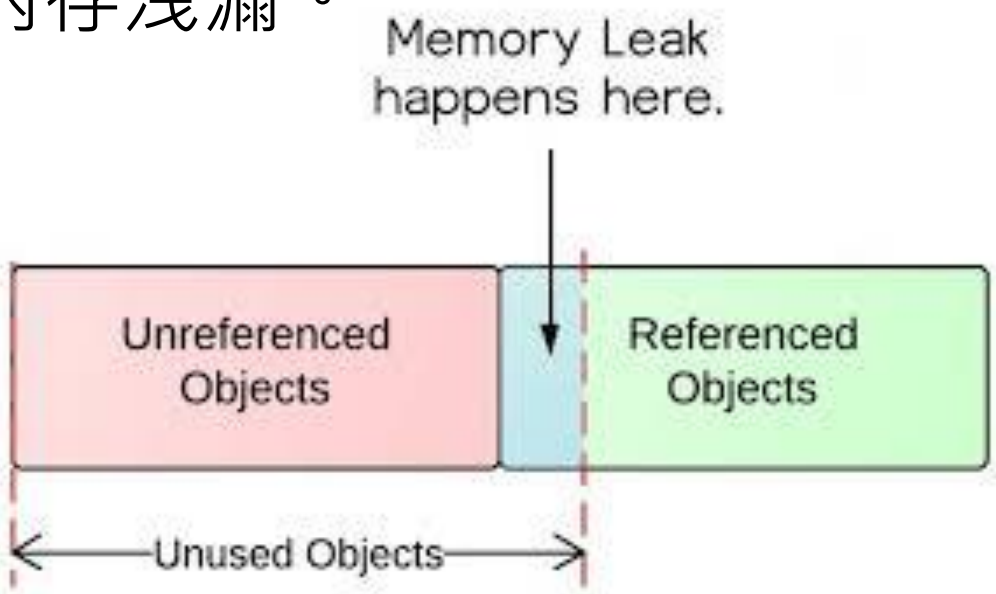
  → JRE內部有一個Java虛擬機器（Java Virtual Machine，JVM）

以及一些標準的類別庫（Class Library）。

自動垃圾回收
(Garbage Collection)

- C++語言大多不提供垃圾收集機制，若程式員在stack分配記憶體卻又沒刪除，易產生內存洩漏。
- 當物件沒有任何參照時，

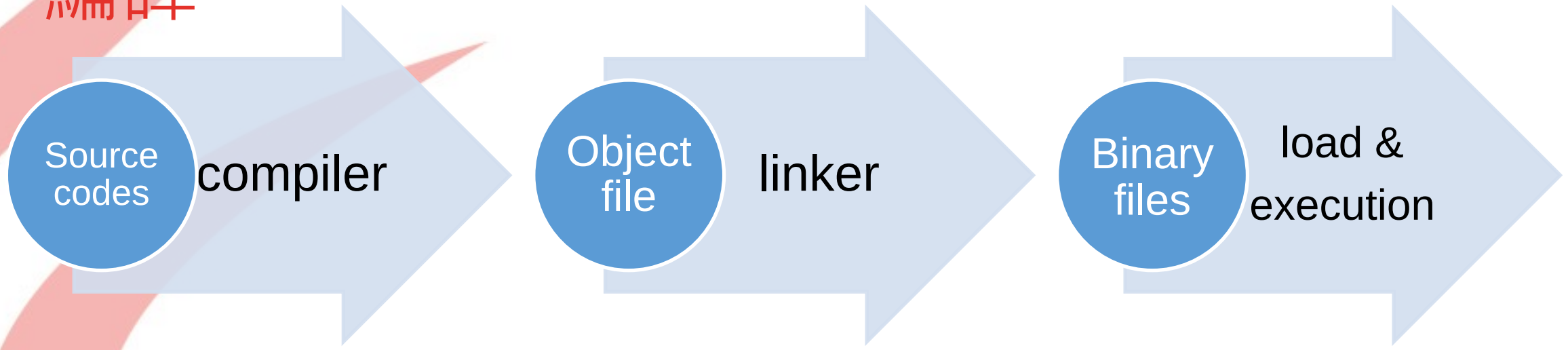
會自動刪除這個物件佔用的空間。



編譯與直譯

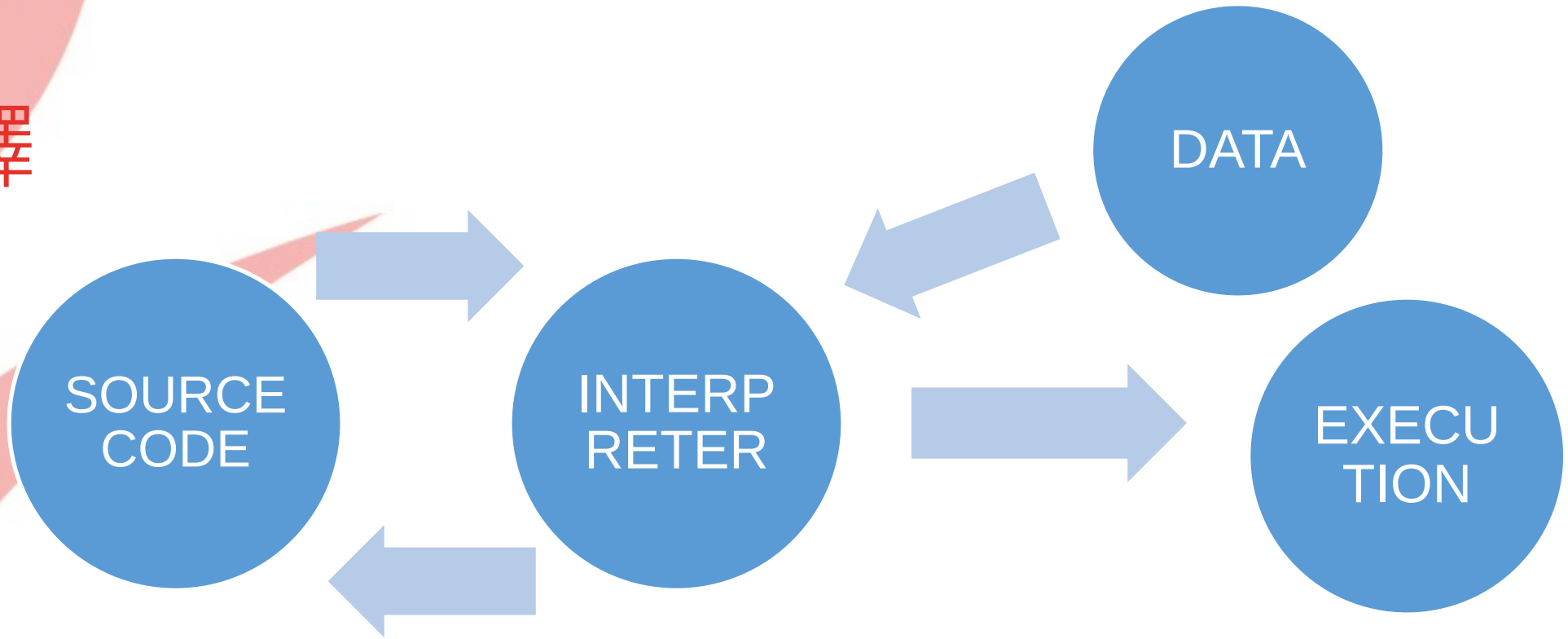
高階語言的執行方式有編譯式與直譯式兩種組譯

編譯



- 優點：執行期的執行速度較快
- 缺點：要先編譯才能執行程式、開發長度和除錯時間較長
- Ex： C, C++, FORTRAN 等。

直譯



- 優點：初學者易使用
- 缺點：程式完成時，執行較慢
- Ex：Python, Ruby, Perl等。

C++和Java的比較

不同的設計目標

不同的設計目標

- C++和Java語言它們有著不同的設計目標
- C++ 被設計成主要用在系統性應用程式設計上
- Java 最開始是被設計用來支援網路計算。
- 它依賴一個虛擬機來保證安全和可移植性。

C++	Java
幾乎和C語言相容	不對之前的語言向前相容
程序式程式設計、 物件導向程式設計	必須使用物件導向的 程式設計方式
一次編寫多處編譯	一次編寫多處執行
被編譯成機器碼	被編譯成Java虛擬機的 位元組碼
能直接使用底層系統介面	在虛擬機器中執行

- 解決此類問題的方法之一，是定義一種虛擬機器(Virtual Machine, VM)，讓程式語言編譯時不要翻成實體機器的指令，而是翻成VM的目的碼。

- VM一般是以軟體來模擬的，只要新的平台有VM，則原始程式不用編譯，就可以執行舊機器上已有的VM目的碼。

JVM

翻譯員

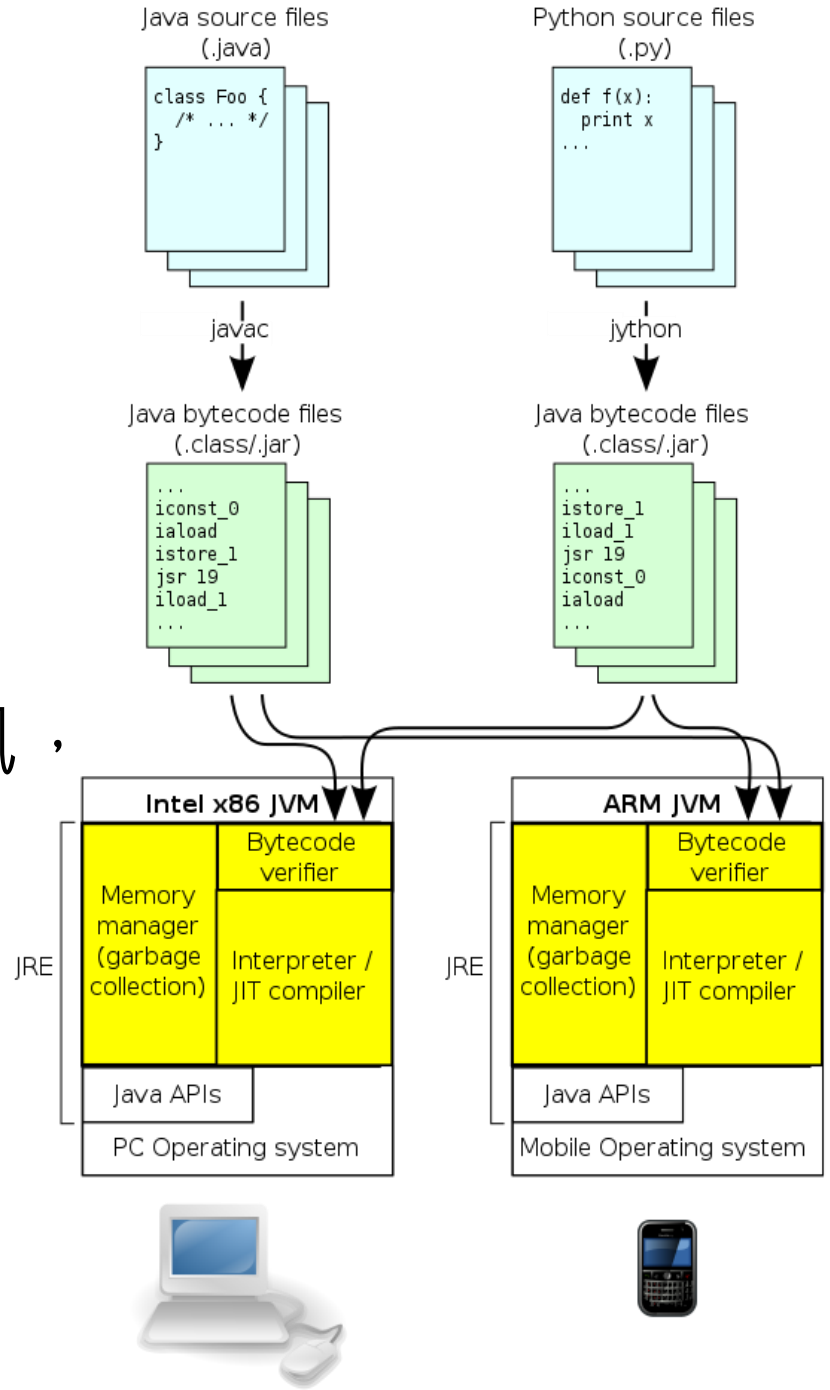
- Java虛擬機（JVM），

一種能夠執行Java bytecode的虛擬機器，以堆疊(stack)結構機器來進行實做。

- JVM遮蔽了與具體作業系統平台相關的資訊，

使得Java程式只需生成在Java虛擬機器上執行的位元組碼，

就可以在多種平台上不加修改地執行。



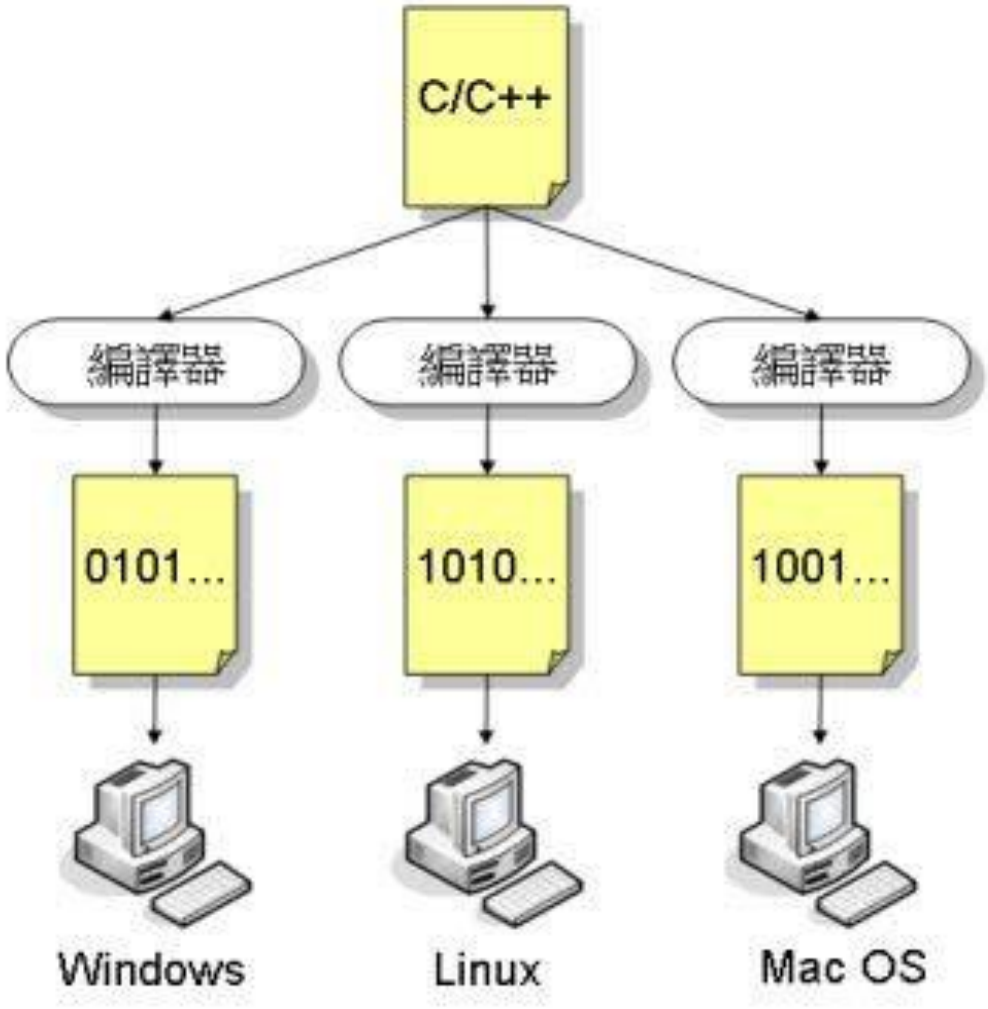
- 要有個「翻譯員」將你寫的C/C++程式，

翻譯為電腦看得懂的0101序列指令，

這個翻譯員就是所謂的「編譯器」（Compiler）

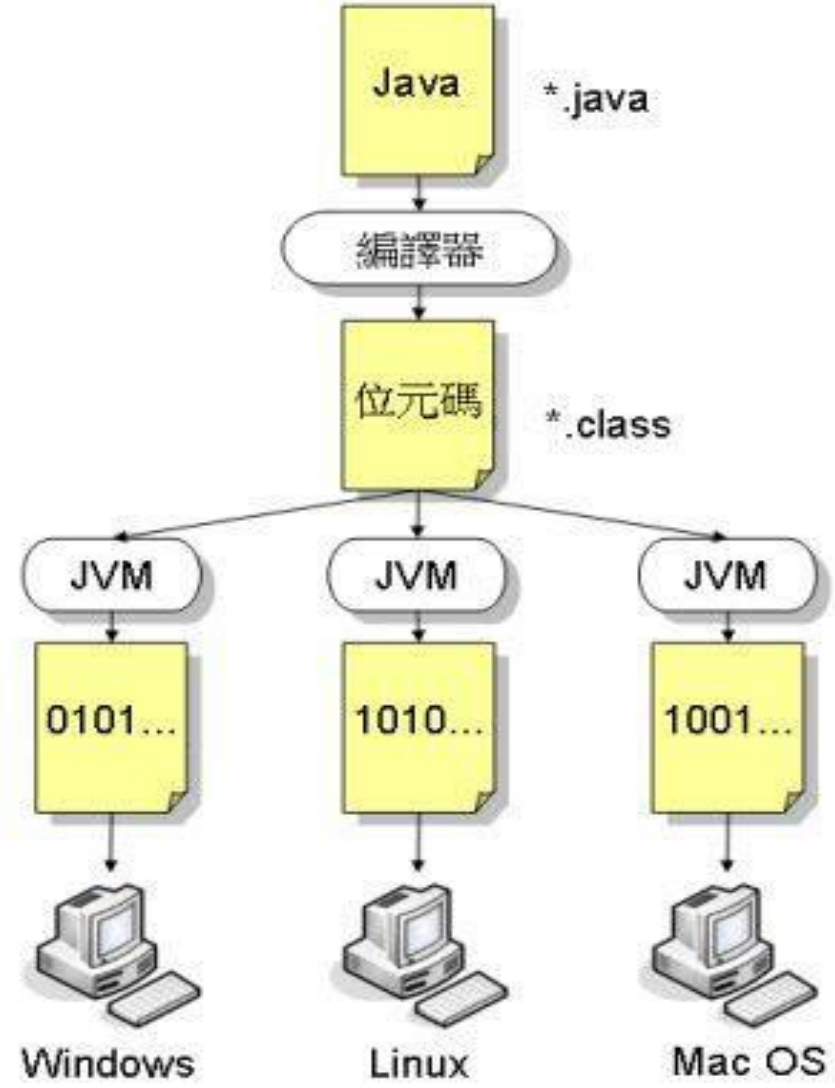
- 每個平台認識的0101序列並不一樣

不同的平台必須使用不同的編譯器來翻譯你的程式。



- Java編譯時，翻譯為中間格式的位元組碼（bytecode）。
- 如果想要執行這個位元組碼檔案，

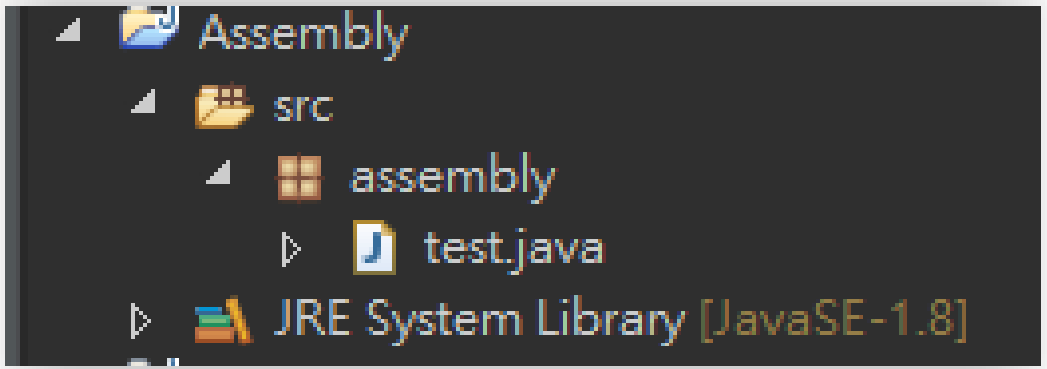
目標平台上必須安裝JVM。



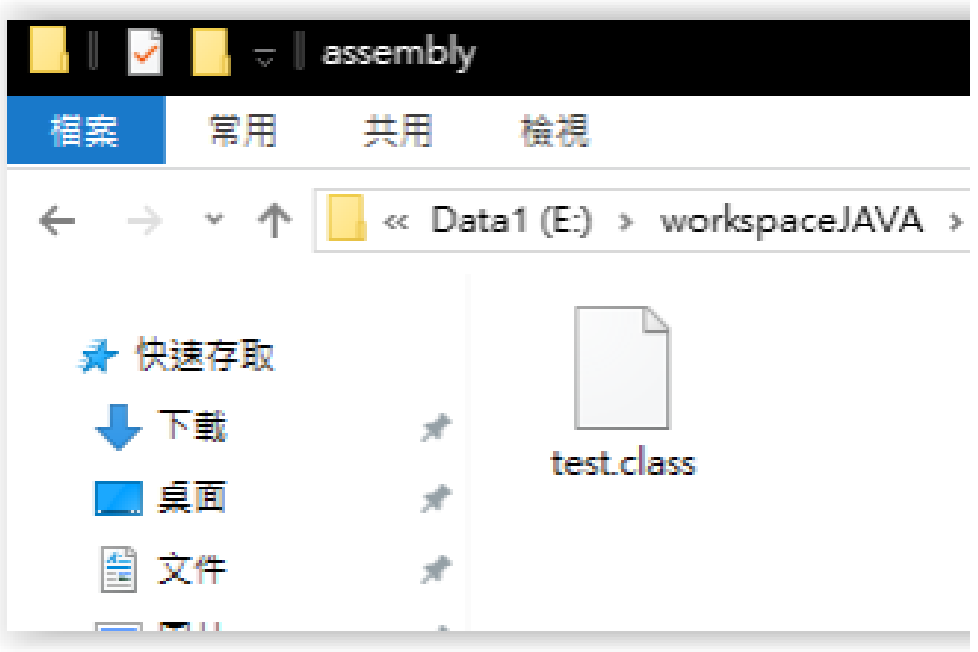
位元組碼(Bytecode)

Java的跨平台性使得編譯後的Bytecode可以直接在網路上流通，Bytecode被執行前並不需要重新編譯，JVM虛擬機會利用 Bytecode 解譯器執行 bytecode，或利用即時編譯器 (JIT) 將 Bytecode 進一步轉換成該電腦上的機器碼執行。

如果將 Java 與 C 語言的系統程式比較，會形成一個很好的對照，Java 的 javac 編譯器可以對照到C語言的 gcc 編譯器，bytecode 相當於虛擬機器上的目的檔，虛擬機器 JVM 則可對照到真實的 CPU。

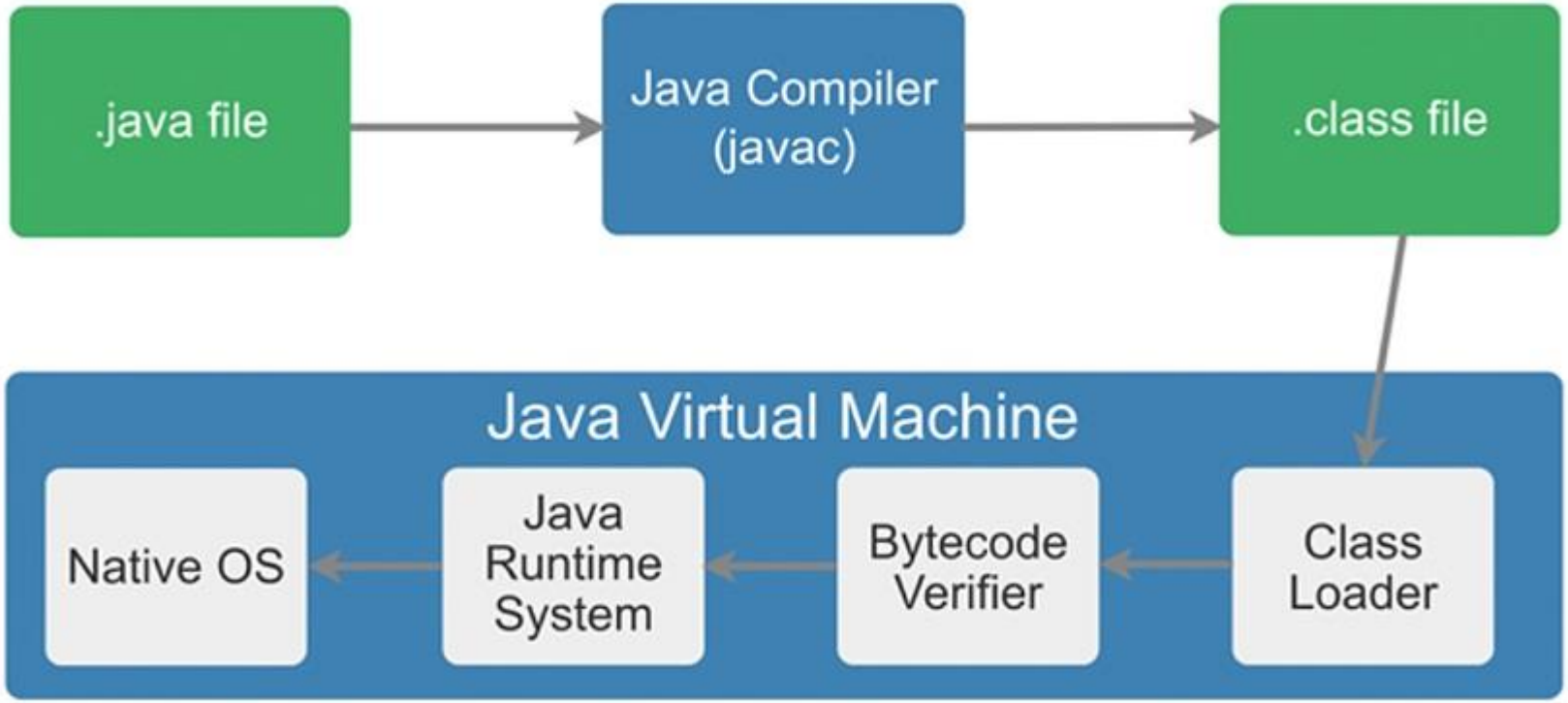


簡單來說，Bytecode就是
java程式碼編譯後的檔案，
編譯後的檔案為「.class」檔



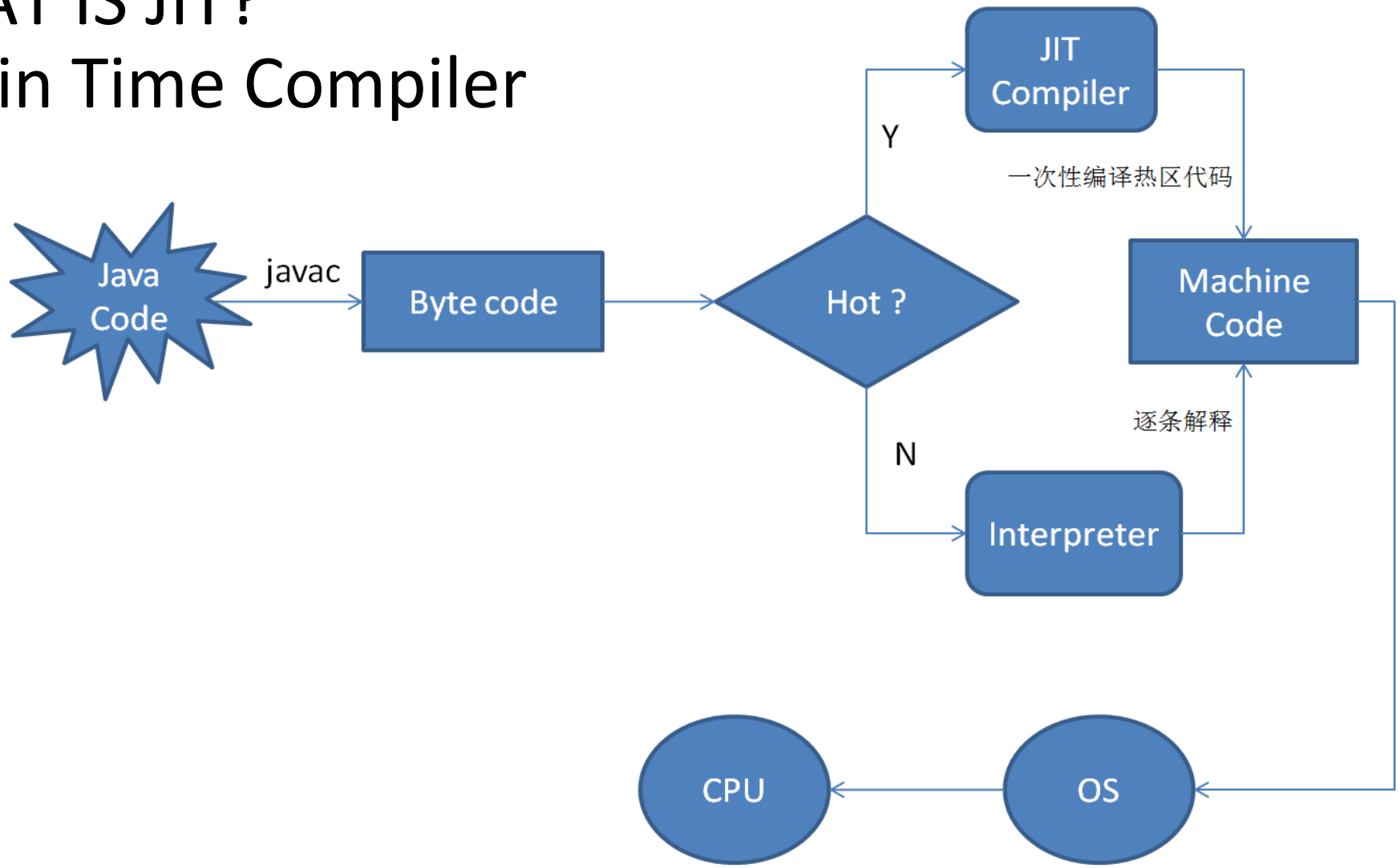
在Eclipse下面新增專案，
寫完的java檔案會編譯成.class檔

What is Java bytecode?



Source: http://www.techlila.com/write-programs-linux/

WHAT IS JIT?
Just in Time Compiler



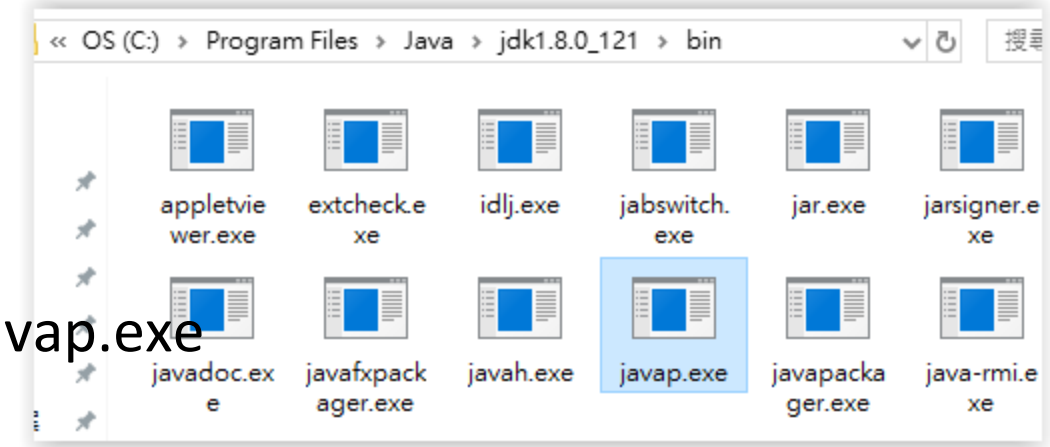
javap

Java class文件分解器，可以反編譯，
也可以查看java編譯器生成的位元組碼，
用於解析class文件

在Oracle公司所提供的 Java 開發工具 JDK 中，包含反組譯器 javap，但是不包含組譯器。使用Javap把byecode反組譯，使用者可以觀察 java 組合語言的設計方式。

Java Development Kit是Oracle公司針對Java Programmer發布的免費軟體開發套件。主要用於編譯、偵錯程式

Javap 在windows下的路徑
C:\Program Files\Java\jdk1.8.0_121\bin\javap.exe



位元組碼(Bytecode)

指令介紹、程式講解

Java 位元組碼（Java bytecode）是Java虛擬機器執行的一種指令格式。大多數操作碼都是一個位元組長，而有些操作需要參數，導致了有一些多位元組的操作碼。

指令可以基本分為以下幾類：

儲存指令（例如：aload_0, istore）

算術與邏輯指令（例如: ladd, fcmpl）

類型轉換指令（例如：i2b, d2i）

物件建立與操作指令（例如：new, putfield）

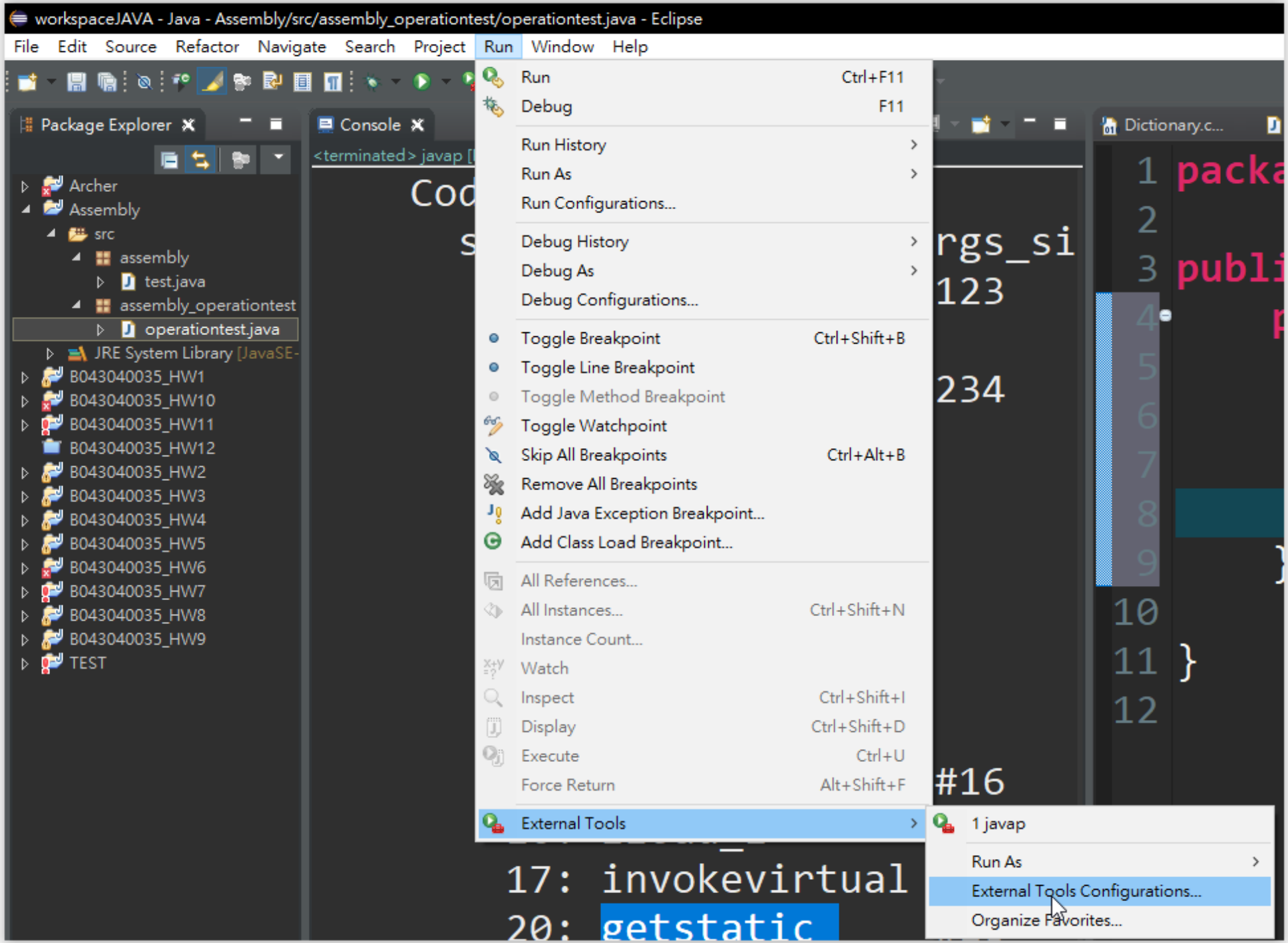
堆疊操作指令（例如：swap, dup2）

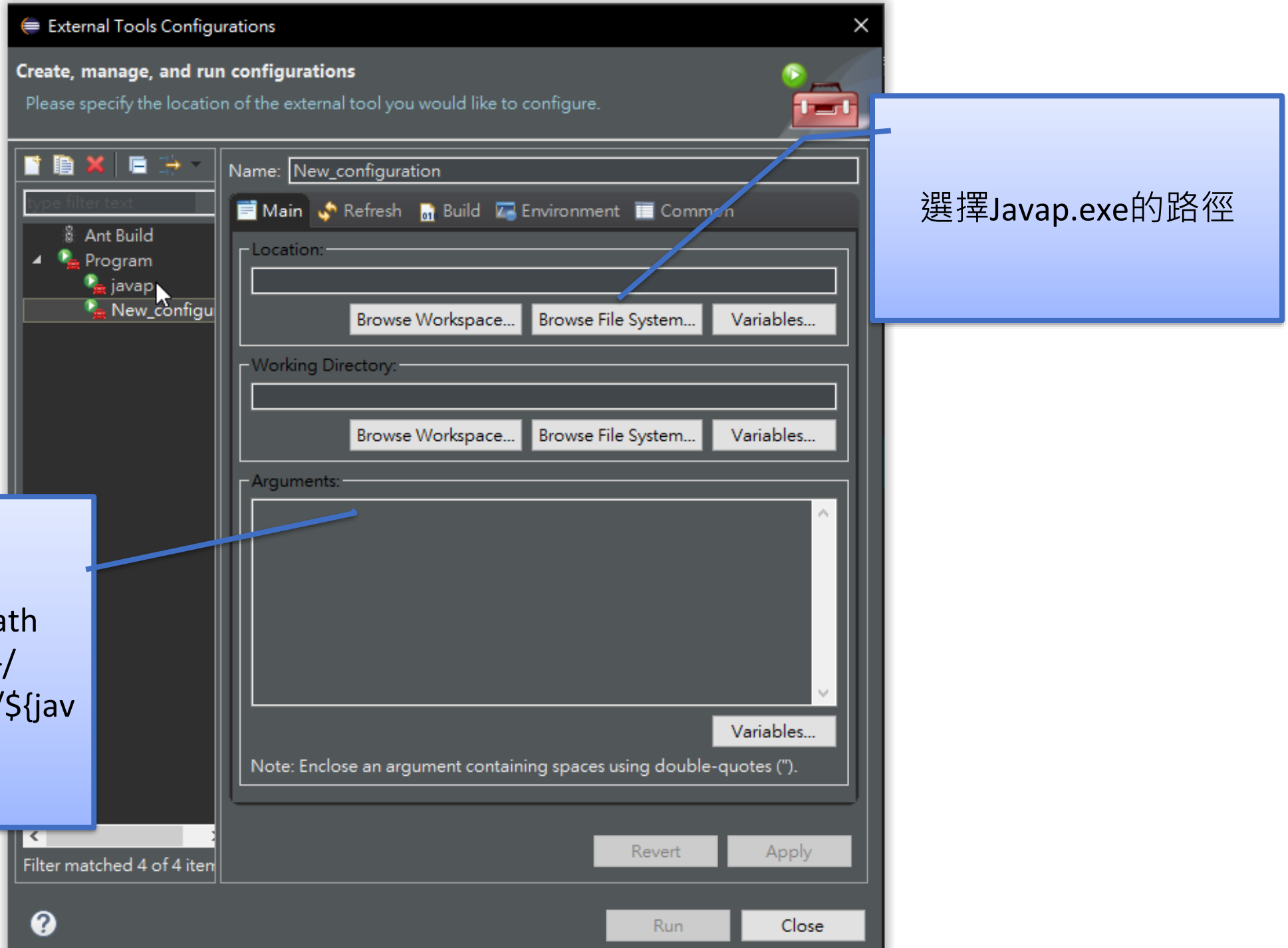
控制轉移指令（例如：ifeq, goto）

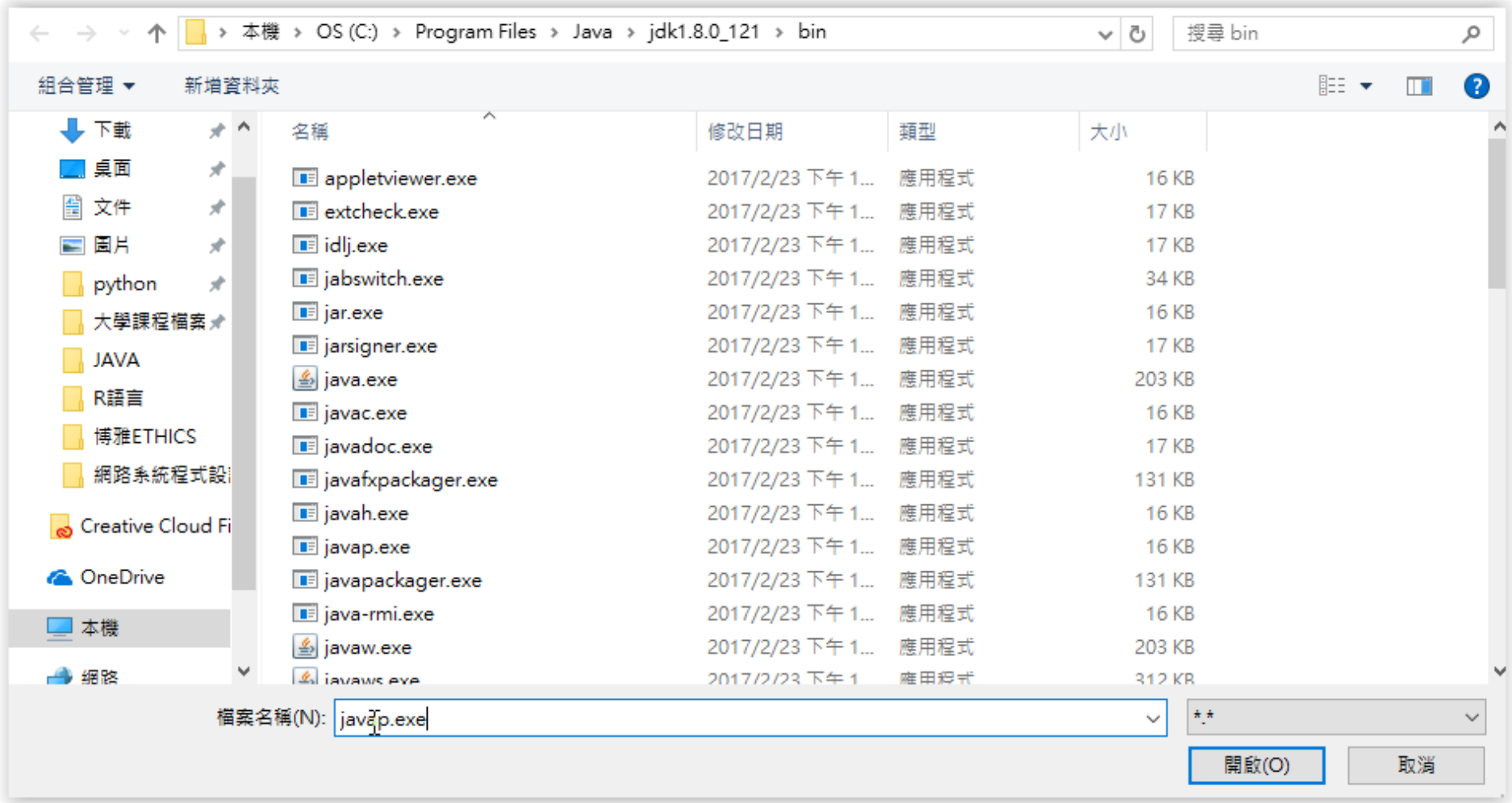
方法呼叫與返回指令（例如：invokespecial, areturn)

前/字尾	運算元類型
i	整數
l	長整數
s	短整數
b	位元組
c	字元
f	單精度浮點數
d	雙精度浮點數
z	布林值
a	參照

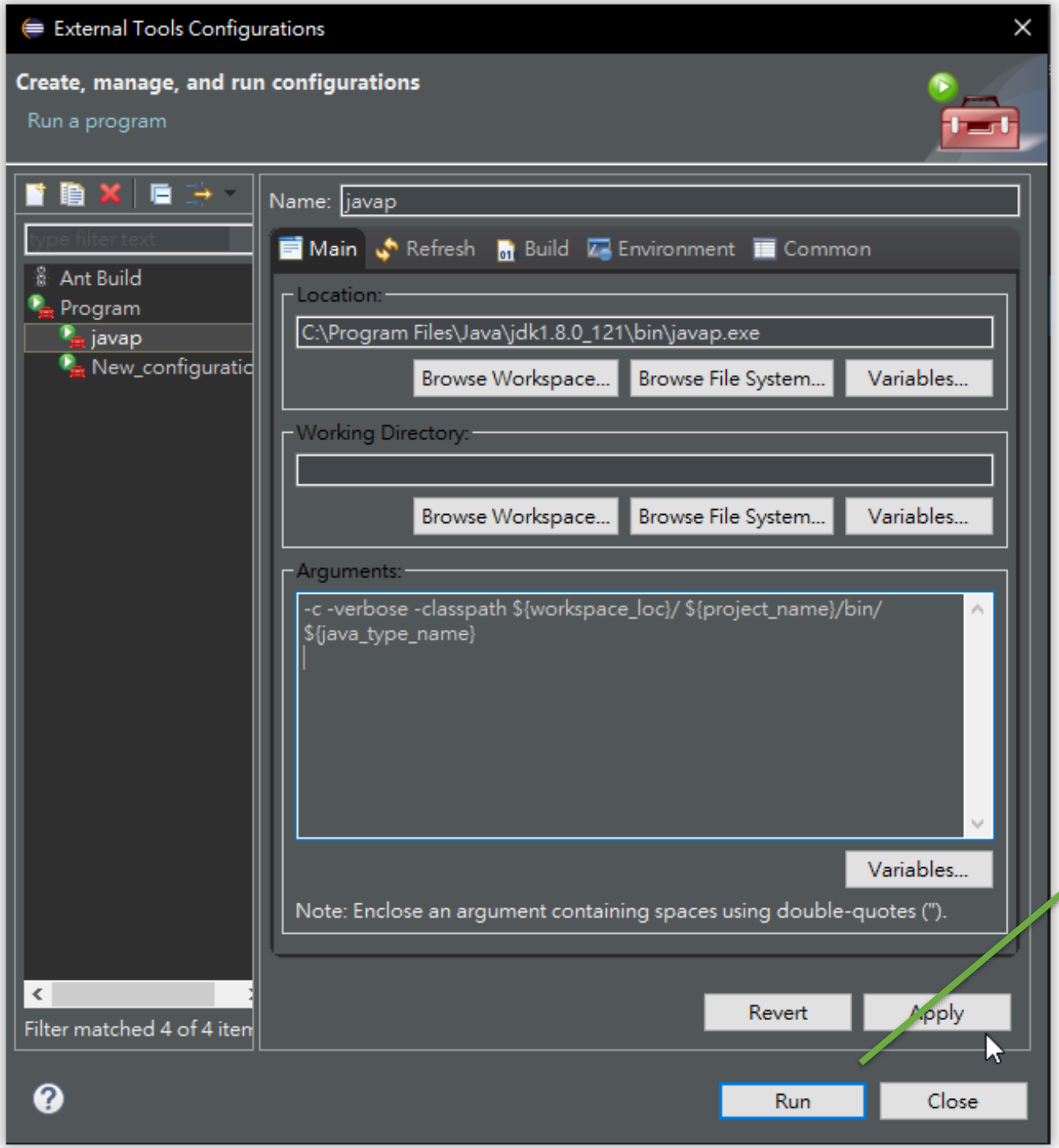
在Eclipse上面使用Javap反組譯查看bytecode內容


workspaceJAVA - Java - Assembly/src/assembly_operationtest/operationtest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Archer
Assembly
src
assembly
test.java
assembly_operationtest
operationtest.java
JRE System Library [JavaSE-
B043040035_HW1
B043040035_HW10
B043040035_HW11
B043040035_HW12
B043040035_HW2
B043040035_HW3
B043040035_HW4
B043040035_HW5
B043040035_HW6
B043040035_HW7
B043040035_HW8
B043040035_HW9
TEST
Console
<terminated> javap [
Run Ctrl+F11
Debug F11
Run History
Run As
Run Configurations...
Debug History
Debug As
Debug Configurations...
Toggle Breakpoint Ctrl+Shift+B
Toggle Line Breakpoint
Toggle Method Breakpoint
Toggle Watchpoint
Skip All Breakpoints Ctrl+Alt+B
Remove All Breakpoints
Add Java Exception Breakpoint...
Add Class Load Breakpoint...
All References...
All Instances... Ctrl+Shift+N
Instance Count...
Watch
Inspect Ctrl+Shift+I
Display Ctrl+Shift+D
Execute Ctrl+U
Force Return Alt+Shift+F
External Tools
1 javap
Run As
External Tools Configurations...
Organize Favorites...
Dictionary.c...
1 packa
2
3 publi
4
5
6
7
8
9 }
10
11 }
12
rgs_si
123
234
#16
17: invokevirtual
20: getstatic


External Tools Configurations
Create, manage, and run configurations
Please specify the location of the external tool you would like to configure.
Ant Build
Program
javap
New_configu
Filter matched 4 of 4 iten
Name: New_configuration
Main
Refresh
Build
Environment
Location:
Browse Workspace...
Browse File System...
Variables...
Working Directory:
Browse Workspace...
Browse File System...
Variables...
Arguments:
Variables...
Note: Enclose an argument containing spaces using double-quotes (").
Revert
Apply
Run
Close
選擇Javap.exe的路徑
輸入參數
-c -verbose -classpath
${workspace_loc}/
${project_name}/bin/${jav
a_type_name}

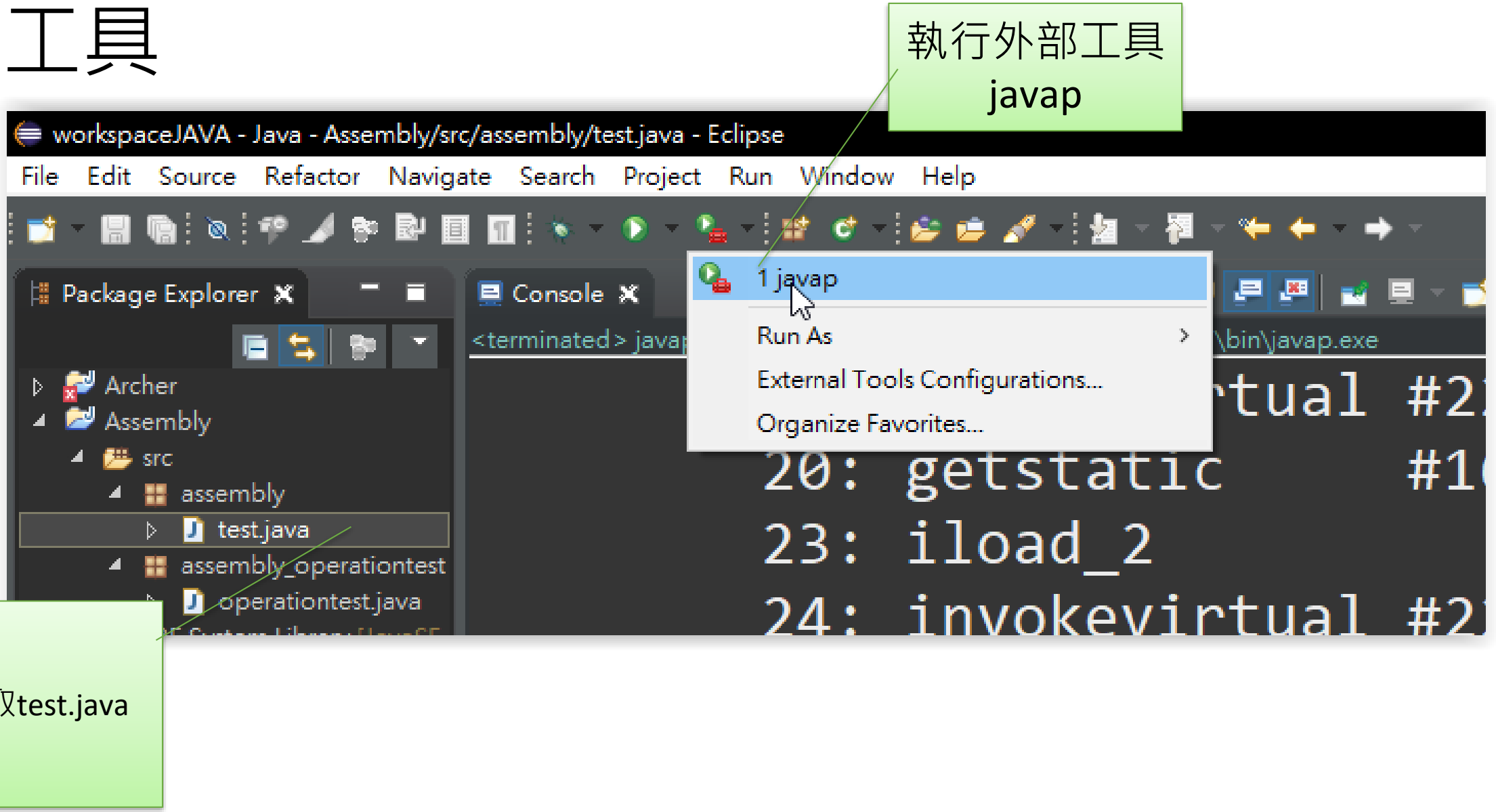


C:\Program Files\Java\jdk1.8.0_121\bin\javap.exe



Click ‘Apply’ and ‘Run’

選取已經編譯好的.java程式上 並執行外部工具



```java
package assembly;

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



        int a=123;
        int b=234;
        int temp;
        temp=a;
        a=b;
        b=temp;
        System.out.println(a);
        System.out.println(b);
    }
}
```

包括Class/String/Integer等各種基本Java資料型態

```
Classfile /E:/workspaceJAVA/Assembly/bin/assembly/test.class
  Last modified 2017/12/12; size 591 bytes
  MD5 checksum 14f2d15ced864f00b50494cfcc64c7c7
  Compiled from "test.java"
public class assembly.test
  minor version: 0
  major version: 52
  flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Class              #2             // assembly/test
   #2 = Utf8               assembly/test
   #3 = Class              #4             // java/lang/Object
   #4 = Utf8               java/lang/Object
   #5 = Utf8               <init>
   #6 = Utf8               ()V
   #7 = Utf8               Code
   #8 = Methodref          #3.#9          // java/lang/Object."<init>":()V
   #9 = NameAndType        #5:#6          // "<init>":()V
  #10 = Utf8               LineNumberTable
  #11 = Utf8               LocalVariableTable
  #12 = Utf8               this
  #13 = Utf8               Lassembly/test;
  #14 = Utf8               main
  #15 = Utf8               ([Ljava/lang/String;)V
  #16 = Fieldref           #17.#19        // java/lang/System.out:Ljava/io/Prin
  #17 = Class              #18            // java/lang/System
  #18 = Utf8               java/lang/System
  #19 = NameAndType        #20:#21        // out:Ljava/io/PrintStream;
  #20 = Utf8               out
  #21 = Utf8               Ljava/io/PrintStream;
  #22 = Methodref          #23.#25        // java/io/PrintStream.println:(I)V
  #23 = Class              #24            // java/io/PrintStream
  #24 = Utf8               java/io/PrintStream
  #25 = NameAndType        #26:#27        // println:(I)V
  #26 = Utf8               println
  #27 = Utf8               (I)V
```

```
public static void main(java.lang.String[]);
  descriptor: ([Ljava/lang/String;)V
  flags: ACC_PUBLIC, ACC_STATIC
  Code:
    stack=2, locals=4, args_size=1
       0: bipush        123
       2: istore_1
       3: sipush        234
       6: istore_2
       7: iload_1
       8: istore_3
       9: iload_2
      10: istore_1
      11: iload_3
      12: istore_2
      13: getstatic     #16
      16: iload_1
      17: invokevirtual #22
      20: getstatic     #16
      23: iload_2
      24: invokevirtual #22
      27: return
```

```java
package assembly;

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



        int a=123;
        int b=234;
        int temp;
        temp=a;
        a=b;
        b=temp;
        System.out.println(a);
        System.out.println(b);
    }
}

```

- 再練習一下另外一題

```
Code:
  stack=2, locals=3, args_size=1
     0: iconst_2
     1: istore_1
     2: goto          34
     5: iconst_2
     6: istore_2
     7: goto          19
    10: iload_1
    11: iload_2
    12: irem
    13: ifne          16
    16: iinc          2, 1
    19: iload_2
    20: iload_1
    21: if_icmplt     10
    24: getstatic     #16
    27: iload_1
    28: invokevirtual #22
    31: iinc          1, 1
    34: iload_1
    35: sipush        1000
    38: if_icmplt     5
    41: return
```

```java
package assembly_operationtest;

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



        for (int i = 2; i < 1000; i++) {
            for (int j = 2; j < i; j++) {
                if (i % j == 0)
                    continue ;
            }
            System.out.println (i);
        }
    }

}

```

- 也可以通過指令來使用javap

javap –c (.class檔)

```
E:\workspaceJAVA\Assembly\bin\assembly>javap -c test.class
Compiled from "test.java"
public class assembly.test {
  public assembly.test();
    Code:
       0: aload_0
       1: invokespecial #8                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: bipush        123
       2: istore_1
       3: sipush        234
       6: istore_2
       7: iload_1
       8: istore_3
       9: iload_2
      10: istore_1
      11: iload_3
      12: istore_2
      13: getstatic     #16                 // Field java/lang/System.out:Ljava/io/PrintStream;
      16: iload_1
      17: invokevirtual #22                 // Method java/io/PrintStream.println:(I)V
      20: getstatic     #16                 // Field java/lang/System.out:Ljava/io/PrintStream;
      23: iload_2
      24: invokevirtual #22                 // Method java/io/PrintStream.println:(I)V
      27: return
}

E:\workspaceJAVA\Assembly\bin\assembly>
```

指令

工作分配

整理資料 - 范真瑋

PPT - 陳奕元

DEMO - 林建業

End