

IntelliJ IDEA

TYIC桃高資訊社

IDE

功能多
不易上手



```
// class 名稱必須跟檔案名稱一樣
public class Main {
    // 任何一個 Java 程式都需要一個主函式(main 函式)
    public static void main(String[] args) {
        // 在 Java 中，使用 System.out.println() 來輸出資料
        System.out.println("Hello, World!");
    }
}
```

整合式開發環境
(Integrated Development Environment, 簡稱 IDE)

通常是針對特定的程式語言設計
並且整合了許多東西，包含：
文字編輯器、除錯器(debugger)、
自動組建工具(build automation)
，部分還有版本控制系統(Version Control System, 簡稱 VCS)
如：PyCharm、Visual Studio、
Code::Blocks、Dev-C++、
Eclipse、**IntelliJ IDEA**

一款好的 IDE 能很大程度加速開發

Java IDE

常見的 Java IDE 如下：

86,544 responses



2023 Stack Overflow 調查

IntelliJ IDEA 26.82%

Eclipse 9.9%

Netbeans 3.19%

IntelliJ IDEA



IntelliJ IDEA
JETBRAINS IDE

作者：JetBrains 公司
免費版：自由開源軟體
旗艦版：專有軟體
使用 Java 編寫

Eclipse



eclipse

作者：Eclipse 基金會
自由開源軟體
使用 Java 編寫

Netbeans



Apache
NetBeans IDE

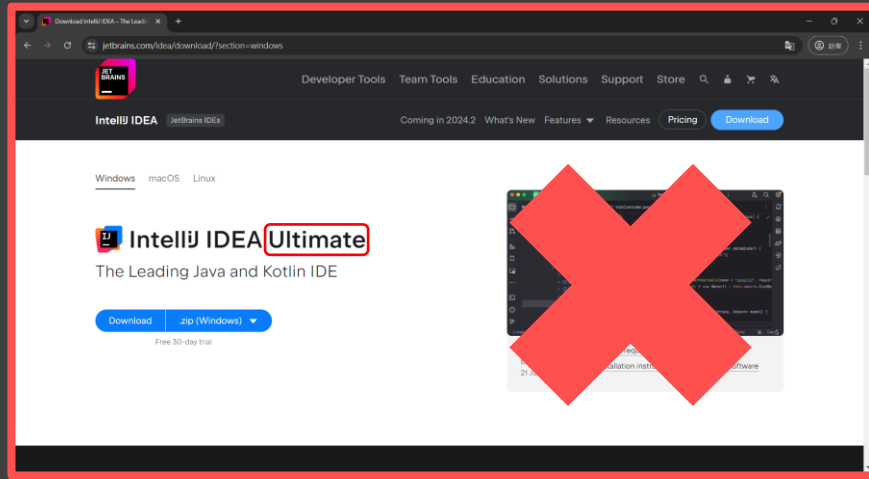
作者：Apache 基金會
自由開源軟體
使用 Java 編寫

IntelliJ IDEA 是這學年會用的

還有許多 Java IDE，但極為少見

下載/安裝

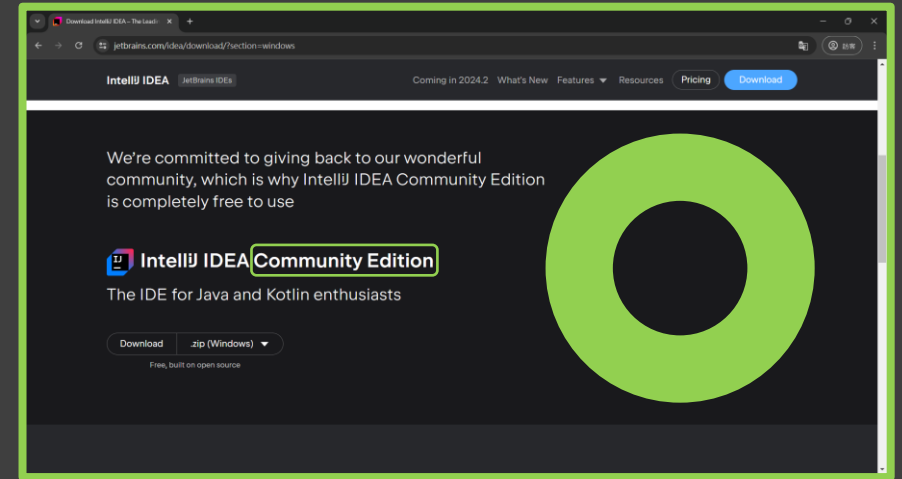
<https://www.jetbrains.com/idea/download/>



付費版(旗艦版)



往下捲動



免費版(社區版)

選擇 **.exe** 後就會開始下載安裝程式
安裝過程非常簡單
按照安裝程式的說明即可



新增專案

開啟後的介面



新增專案

開啟專案資料夾

從版本控制系統獲取

新增專案

點擊 "New Project" 後
會出現右圖視窗
填好名稱後就可以創建專案

注意：不要勾選
"Add Sample Code"

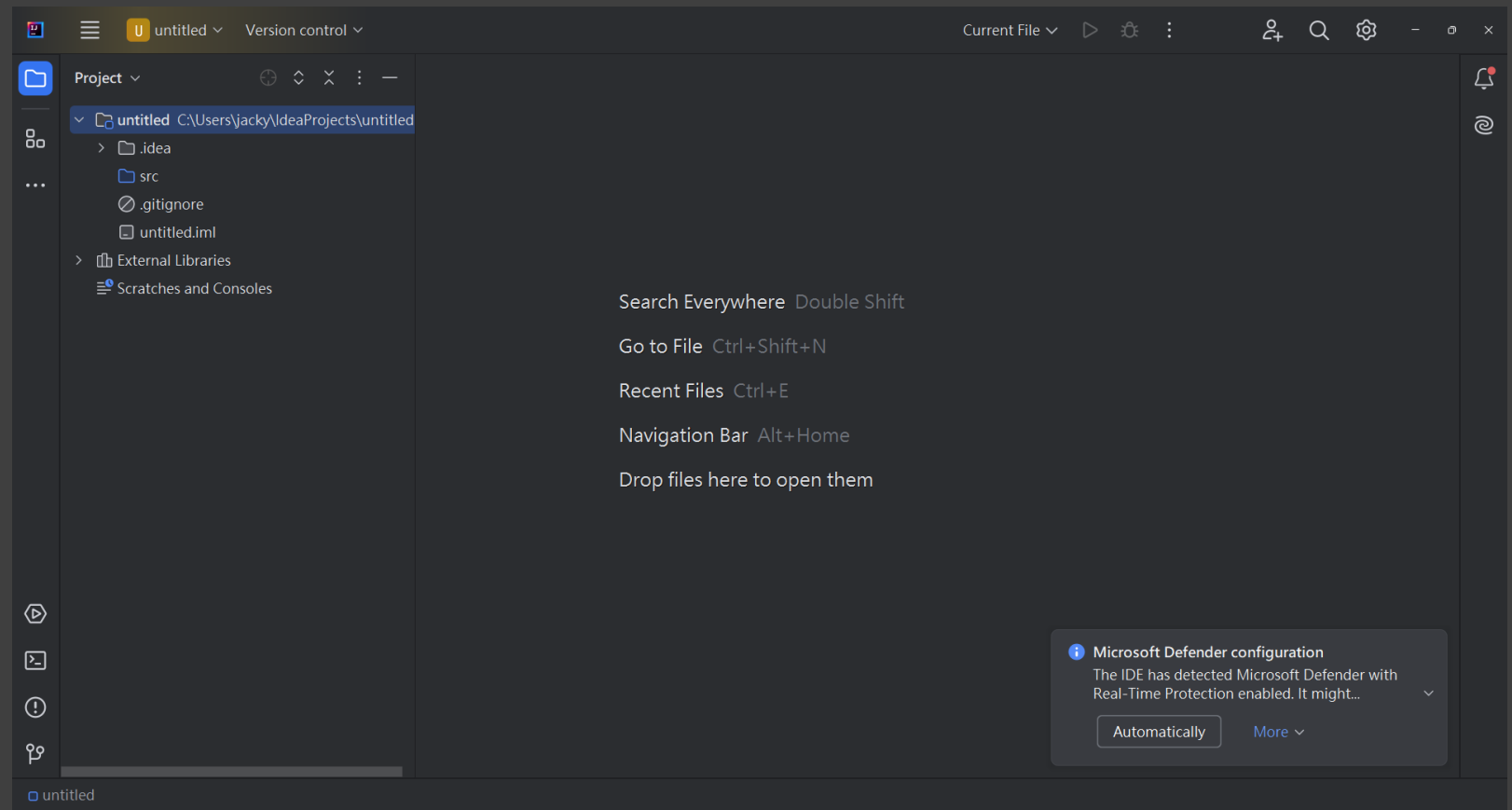
The screenshot shows the 'New Project' dialog in IntelliJ IDEA. The left sidebar lists project types: Java (selected), Kotlin, Groovy, Empty Project, and a section for Generators including Maven Archetype, JavaFX, Spring, and Compose for Desktop. The main area contains the following fields and options:

- 專案名稱 (Name):** A text field containing 'untitled'.
- 儲存位置 (Location):** A text field containing '~\IdeaProjects'.
- Project will be created in:** ~\IdeaProjects\untitled
- Create Git repository:** An unchecked checkbox.
- Build system:** Three buttons: IntelliJ (selected), Maven, and Gradle.
- JDK:** A dropdown menu showing 'temurin-21 Eclipse Temurin 21.0.3'.
- Add sample code:** An unchecked checkbox, highlighted with a yellow box and the text '不要勾' (Don't check).
- Generate code with onboarding tips:** A checked checkbox.
- Advanced Settings:** A collapsed section indicated by a right-pointing arrow.
- Buttons:** 'Create' (highlighted with a yellow box) and 'Cancel'.

創建專案

新增專案

創建完後
就會顯示專案



新增檔案

要新增 **class**

只需要在資料夾上

右鍵 -> **New**

-> **Java Class**

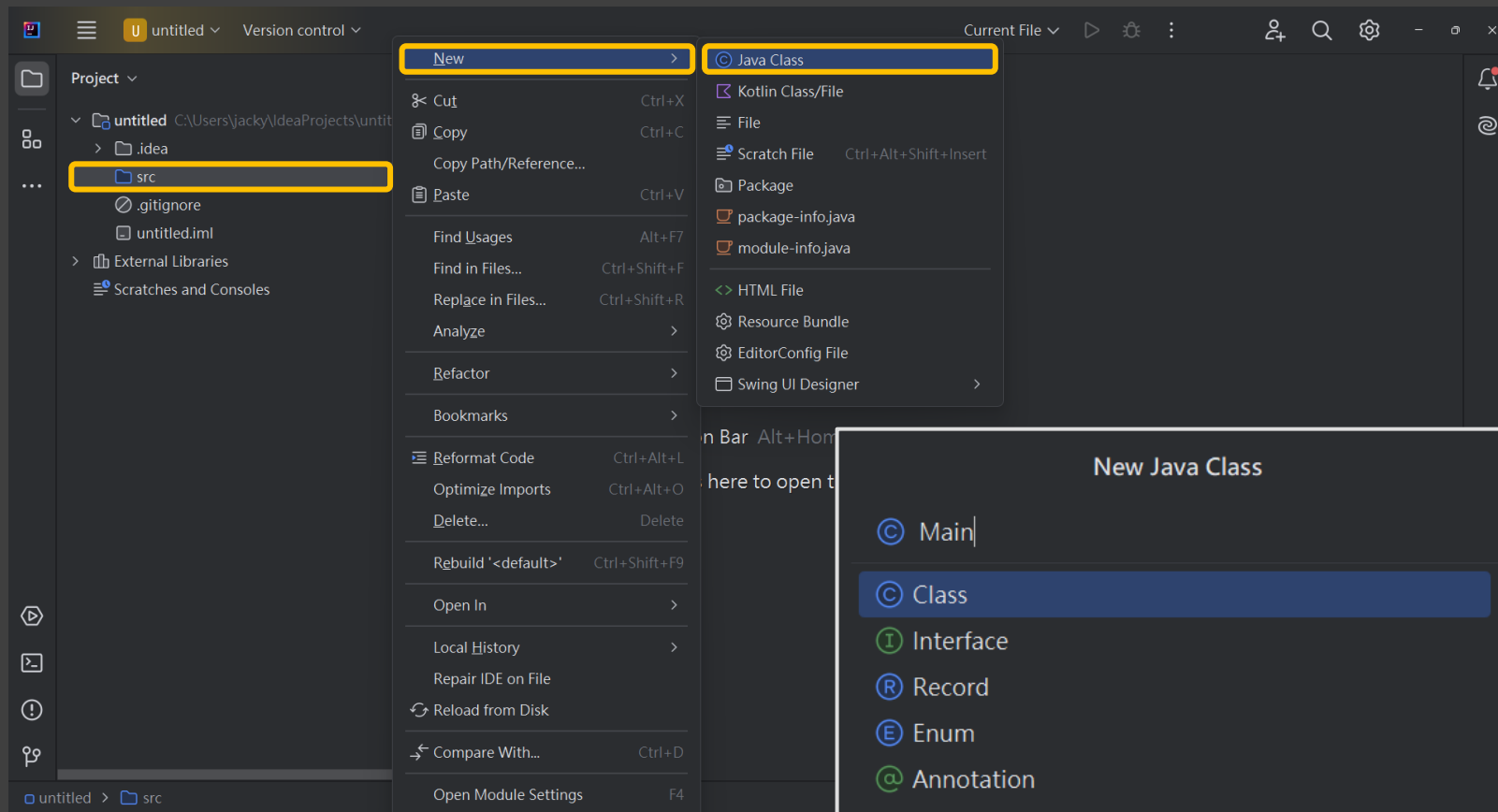
填入類別名稱

按下 **Enter**

即可創建

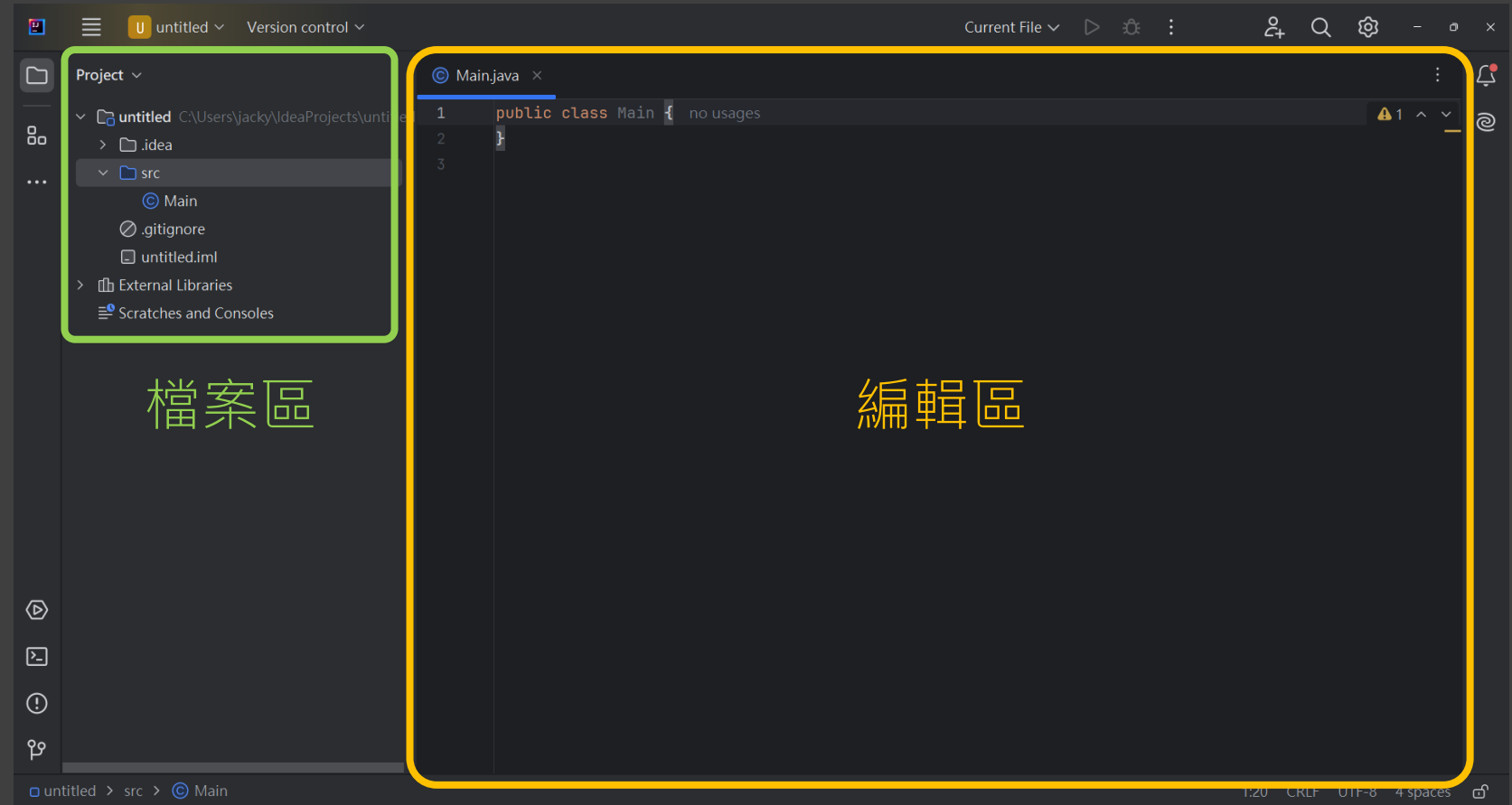
目前 **class**

皆放在 **"src"** 下



新增檔案

創建完後
會自動開啟檔案
即可在編輯區編輯
所有專案檔案
都會出現在檔案區



編輯

嘗試將上一個程式打上去
並觀察打字過程發生了什麼

```
01  import java.util.Scanner; // 載入套件
02
03  public class Main {
04      public static void main(String[] args) {
05          Scanner scanner = new Scanner(System.in); // 創建新的 Scanner 實例
06          System.out.print("姓名 學號 身高 :");
07          String name = scanner.next(); // 讀入下一個字串並存入變數 name
08          int studentId = scanner.nextInt(); // 讀入下一個 int 並存入變數 studentId
09          double height = scanner.nextDouble(); // 讀入下一個 double 並存入變數 height
10          System.out.printf("姓名 :%s 學號 :%d 身高 :%.2f\n", name, studentId, height);
11      }
12  }
```

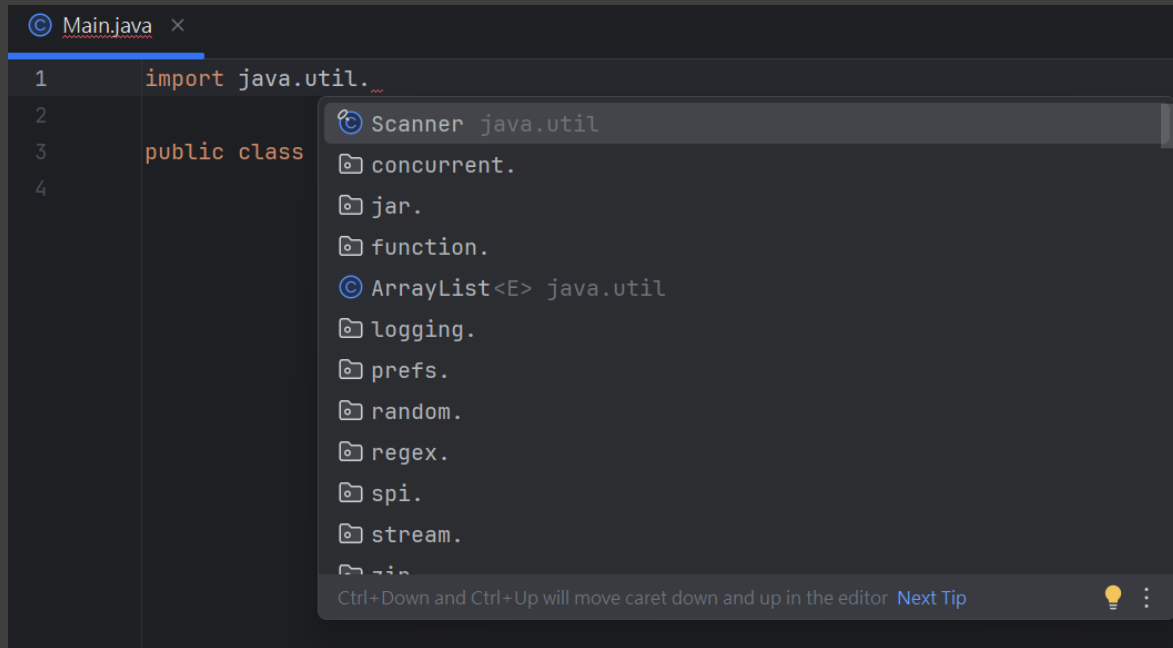


java

自動補全

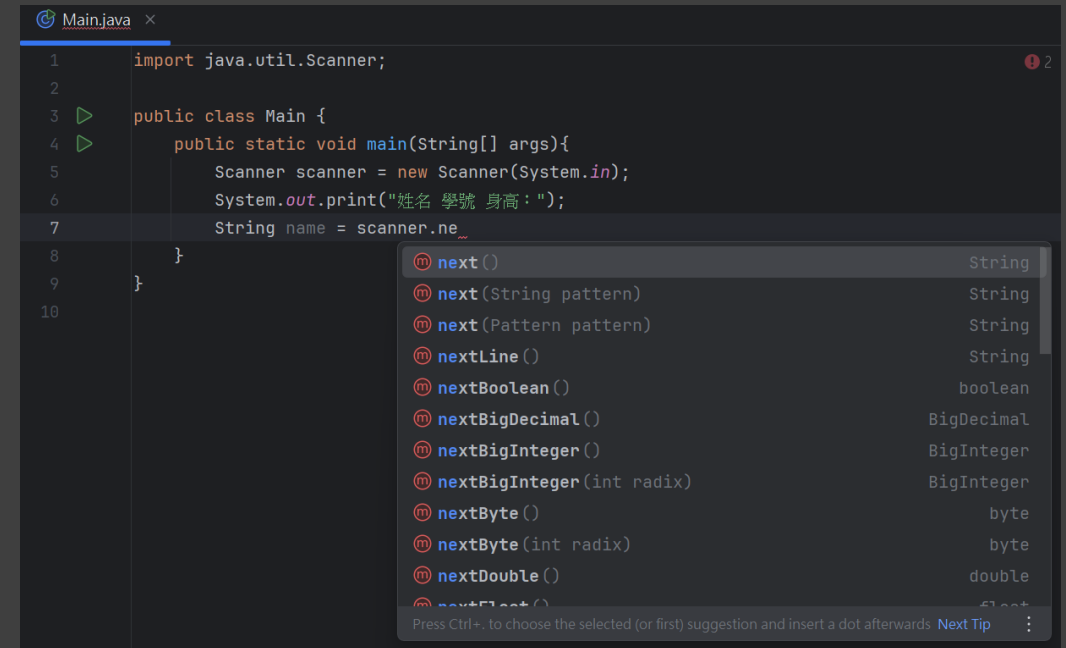
打字過程中出現了些許變化：

1. 打了左括號，會自動補右括號，雙引號也是
2. 會列出可以接什麼



This screenshot shows an IDE window titled 'Main.java'. The code contains an import statement for 'java.util.' followed by a 'public class' declaration. A dropdown menu is open, listing various classes and packages under 'java.util.', including 'Scanner', 'concurrent.', 'jar.', 'function.', 'ArrayList<E>', 'logging.', 'prefs.', 'random.', 'regex.', 'spi.', and 'stream.'. The 'Scanner' class is highlighted at the top of the list. A 'Next Tip' message is visible at the bottom of the dropdown, stating: 'Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip'.

```
1 import java.util.  
2  
3 public class  
4
```



This screenshot shows the same IDE window 'Main.java' with the code completed to: 'import java.util.Scanner;', 'public class Main {', 'public static void main(String[] args){', 'Scanner scanner = new Scanner(System.in);', 'System.out.print("姓名 學號 身高:");', 'String name = scanner.ne...'. A dropdown menu is open for the 'next' method, listing various overloaded methods and their return types: 'next()', 'next(String pattern)', 'next(Pattern pattern)', 'nextLine()', 'nextBoolean()', 'nextBigDecimal()', 'nextBigInteger()', 'nextBigInteger(int radix)', 'nextByte()', 'nextByte(int radix)', 'nextDouble()', and 'nextFloat()'. The 'next()' method is highlighted at the top of the list. A 'Next Tip' message is visible at the bottom of the dropdown, stating: 'Press Ctrl+, to choose the selected (or first) suggestion and insert a dot afterwards Next Tip'.

```
1 import java.util.Scanner;  
2  
3 public class Main {  
4     public static void main(String[] args){  
5         Scanner scanner = new Scanner(System.in);  
6         System.out.print("姓名 學號 身高:");  
7         String name = scanner.ne...  
8     }  
9 }  
10
```

自動補全

不只列出可以接什麼
還可以通過上下鍵
選擇要輸入的
然後按 **Tab** 或 **Enter**
讓 **IDE** 幫你自動補全



The screenshot shows a code editor window titled 'Main.java'. The code is as follows:

```
1 import java.util.Scanner;  
2  
3 public class Main {  
4     public static void main(String[] args) {  
5         Scanner scanner = new Scanner(System.in);  
6         System.out.print("姓名 學號 身高: ");  
7         String name = scanner.ne  
8     }  
9 }
```

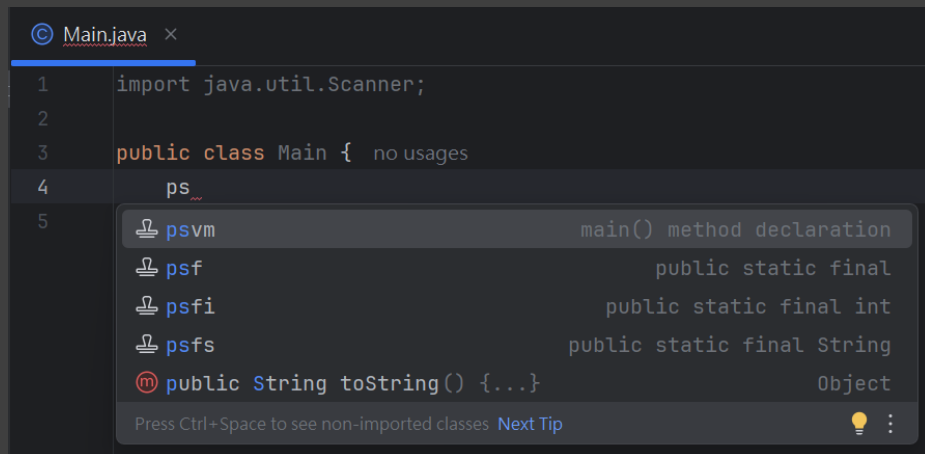
An auto-completion dropdown menu is visible, listing the following methods and their return types:

- `nextBoolean()` boolean
- `nextBigDecimal()` BigDecimal
- `nextBigInteger()` BigInteger
- `nextBigInteger(int radix)` BigInteger
- `nextByte()` byte
- `nextByte(int radix)` byte
- `nextDouble()` double
- `nextFloat()` float
- `nextInt()` int** (highlighted)
- `nextInt(int radix)` int
- `nextLong()` long

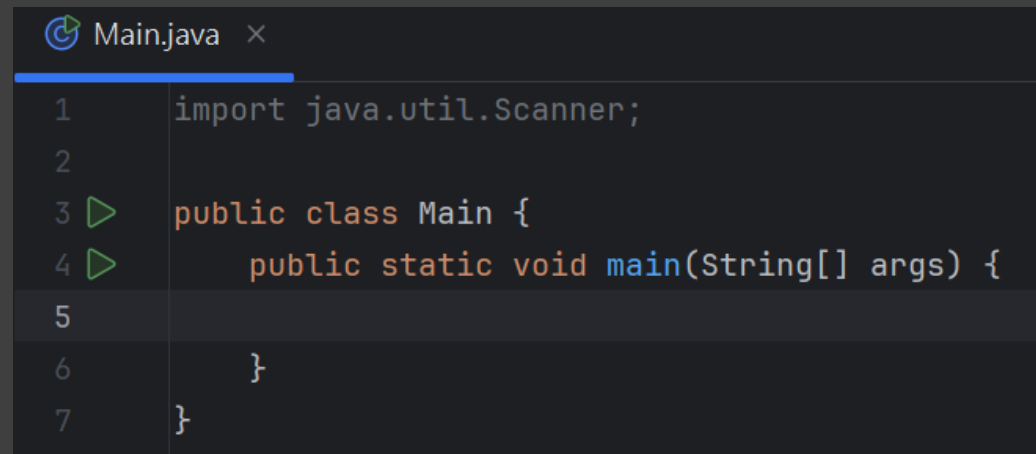
At the bottom of the dropdown, there is a tip: 'Ctrl+Down and Ctrl+Up will move caret down and up in the editor. Next Tip'.

自動補全

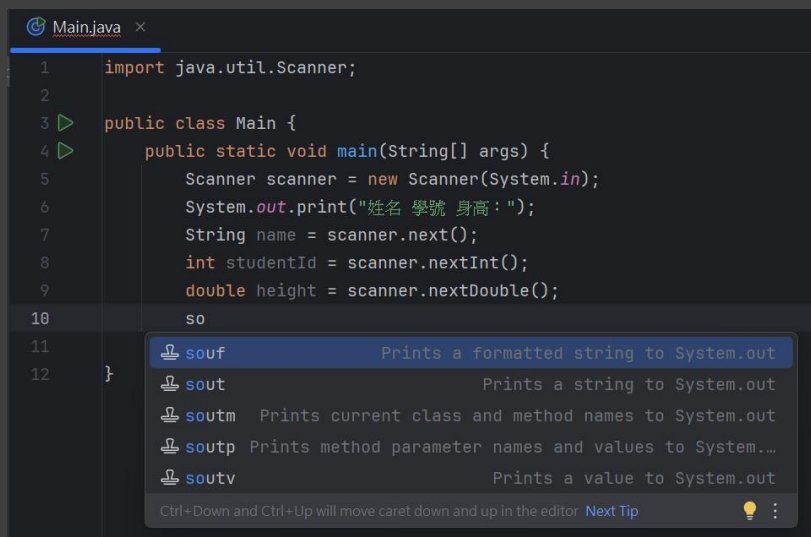
甚至，**IDEA** 還有內建許多快捷縮寫，如 **psvm**、**sout** 等



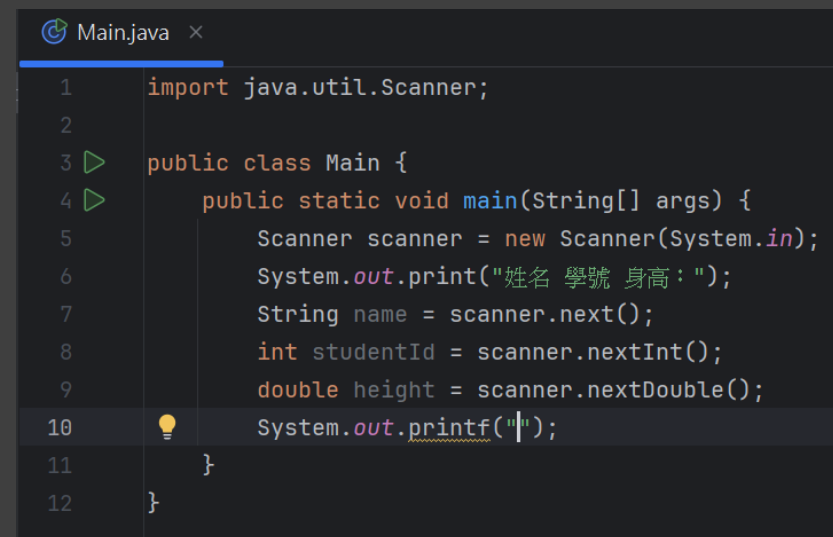
```
1 import java.util.Scanner;
2
3 public class Main { no usages
4     ps
5     [psvm main() method declaration
    [psf public static final
    [psfi public static final int
    [psfs public static final String
    [public String toString() {...} Object
    Press Ctrl+Space to see non-imported classes Next Tip
```



```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5
6     }
7 }
```



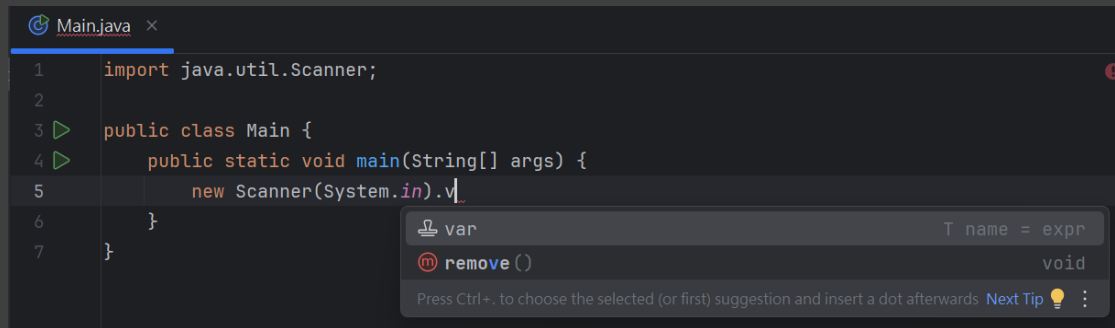
```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("姓名 學號 身高: ");
7         String name = scanner.next();
8         int studentId = scanner.nextInt();
9         double height = scanner.nextDouble();
10        so
11    }
12 }
```



```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("姓名 學號 身高: ");
7         String name = scanner.next();
8         int studentId = scanner.nextInt();
9         double height = scanner.nextDouble();
10        System.out.print("|");
11    }
12 }
```

自動補全

還能通過 `.var` 等縮寫讓 **IDE** 幫你補更多東西

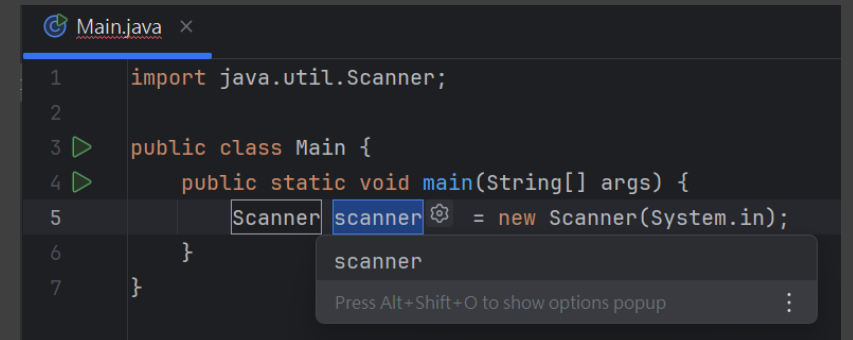


```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         new Scanner(System.in).v
6     }
7 }
```

The IDE shows a completion list for the partial code `new Scanner(System.in).v`. The suggestions are:

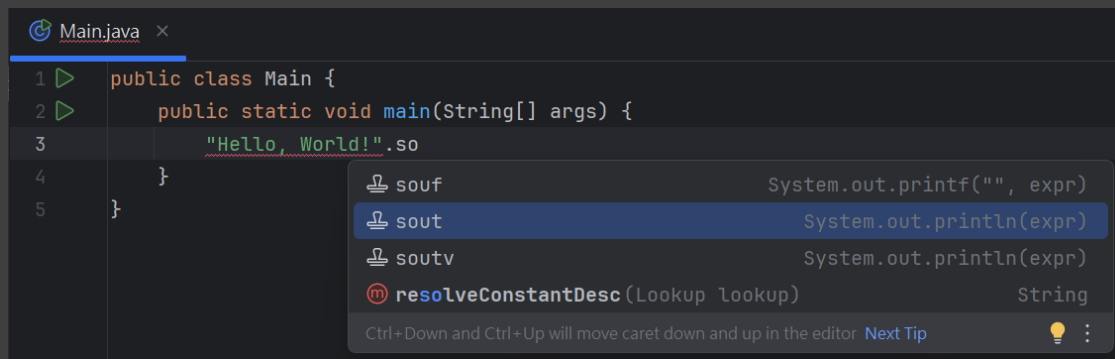
- `var` (with a tooltip `T name = expr`)
- `remove()` (with a tooltip `void`)

Below the suggestions, it says: "Press Ctrl+., to choose the selected (or first) suggestion and insert a dot afterwards Next Tip ⚡ ⋮"



```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6     }
7 }
```

The IDE shows the result of the first code completion. The code is now `Scanner scanner = new Scanner(System.in);`. A tooltip for `scanner` is visible, showing the text `scanner` and the instruction "Press Alt+Shift+O to show options popup ⋮".

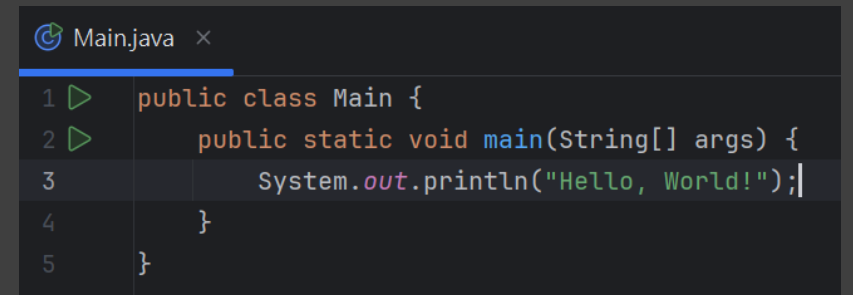


```
1 public class Main {
2     public static void main(String[] args) {
3         "Hello, World!".so
4     }
5 }
```

The IDE shows a completion list for the partial code `"Hello, World!".so`. The suggestions are:

- `souf` (with a tooltip `System.out.printf("", expr)`)
- `sout` (with a tooltip `System.out.println(expr)`)
- `soutv` (with a tooltip `System.out.println(expr)`)
- `resolveConstantDesc(Lookup lookup)` (with a tooltip `String`)

Below the suggestions, it says: "Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip ⚡ ⋮"

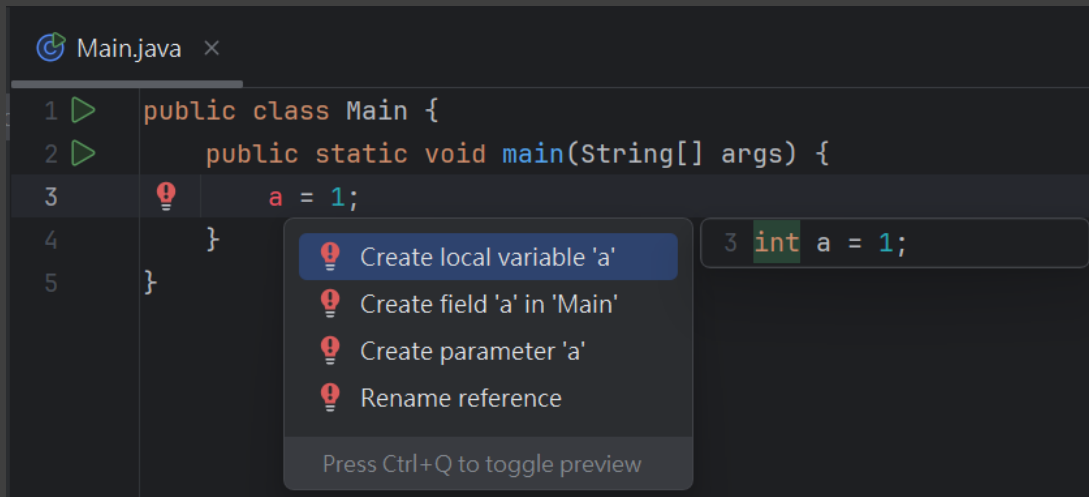


```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```

The IDE shows the result of the second code completion. The code is now `System.out.println("Hello, World!");`.

動作

文字游標在某些地方時
還可以按 **Alt + Enter** 開啟動作選單
可以選擇並執行動作



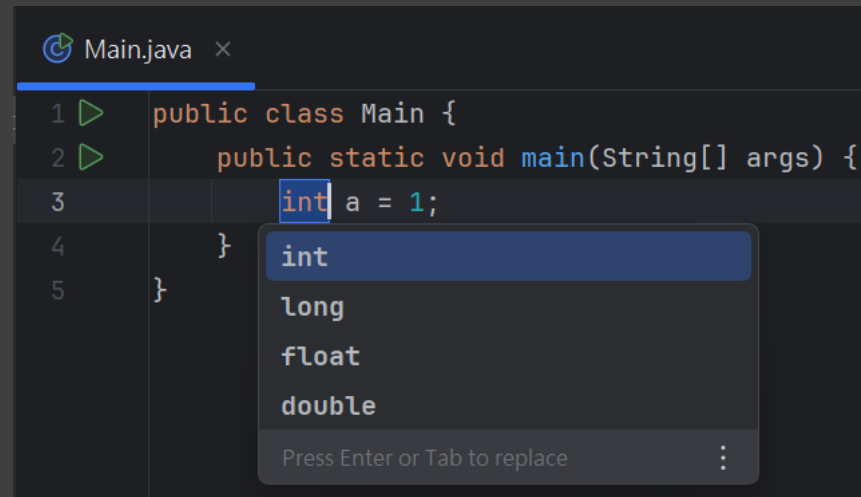
The screenshot shows a code editor with a file named 'Main.java'. The code is as follows:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         a = 1;  
4     }  
5 }
```

The cursor is positioned at line 3, column 15, just before the variable 'a'. A red lightbulb icon indicates an action menu. The menu is open, showing the following options:

- Create local variable 'a'
- Create field 'a' in 'Main'
- Create parameter 'a'
- Rename reference

At the bottom of the menu, it says 'Press Ctrl+Q to toggle preview'. To the right of the menu, a preview box shows the result of the 'Create local variable' action: `3 int a = 1;`.



The screenshot shows the same code editor as the left one, but now the 'int' type has been selected from the action menu. The code is now:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int a = 1;  
4     }  
5 }
```

The action menu is still open, showing the following options:

- int
- long
- float
- double

At the bottom of the menu, it says 'Press Enter or Tab to replace'.

自動格式化

如果想要讓程式碼更容易閱讀
可以開啟自動格式化
讓 IDE 在檔案儲存時幫你格式化

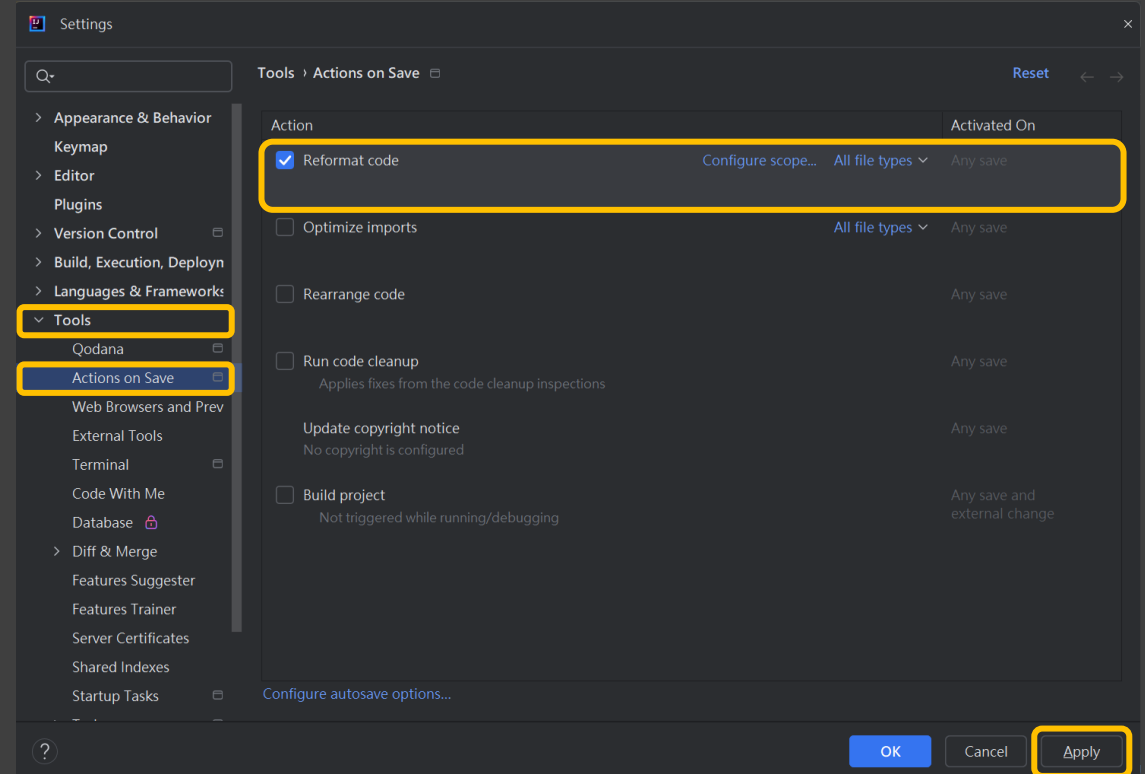
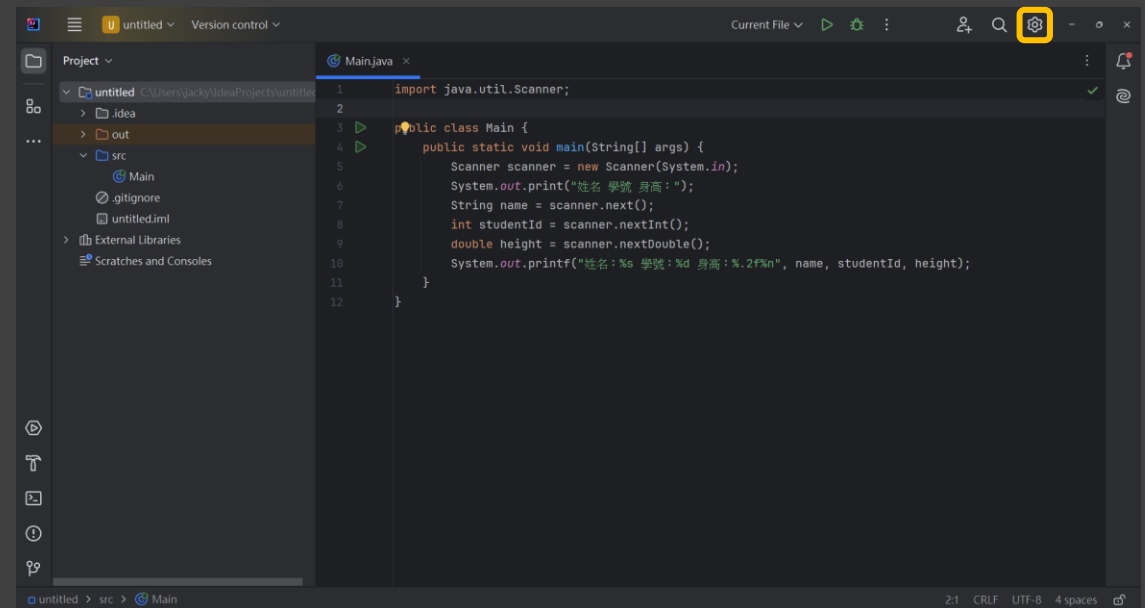
開啟方法：

Setting -> Tools

-> Actions on Save

-> Reformat code

然後按下 Apply 即可



自動格式化

格式化前：

```
Main.java ×
1 import java.util.Scanner;
2 public class Main{public static void main(String[]args){
3     Scanner scanner=new Scanner(System.in);
4         System.out.print("姓名 學號 身高:");
5         String name=scanner.next();
6     int studentId=scanner.nextInt();double height=scanner.nextDouble();
7     System.out.printf("姓名:%s 學號:%d 身高:%.2f\n",name,studentId,height);
8 }}
```

格式化後：

```
Main.java ×
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("姓名 學號 身高:");
7         String name = scanner.next();
8         int studentId = scanner.nextInt();
9         double height = scanner.nextDouble();
10        System.out.printf("姓名:%s 學號:%d 身高:%.2f\n", name, studentId, height);
11    }
12 }
```

快捷鍵

除了支援之前介紹的快捷鍵之外
還有許多好用的快捷鍵

如 **Ctrl + D** 複製貼上該行程式碼

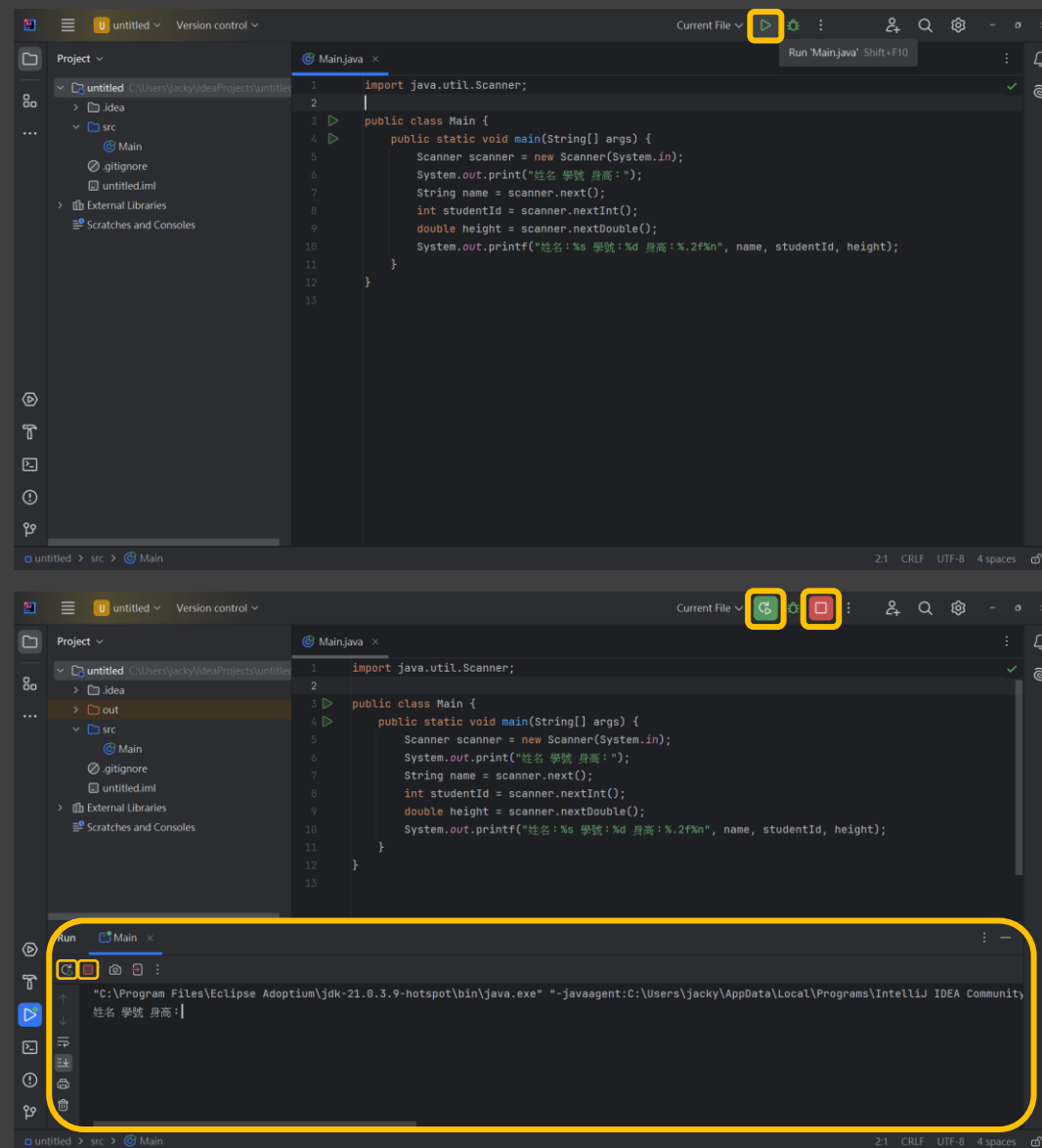
Ctrl + / 單行註解

Ctrl + Shift + / 多行註解

執行

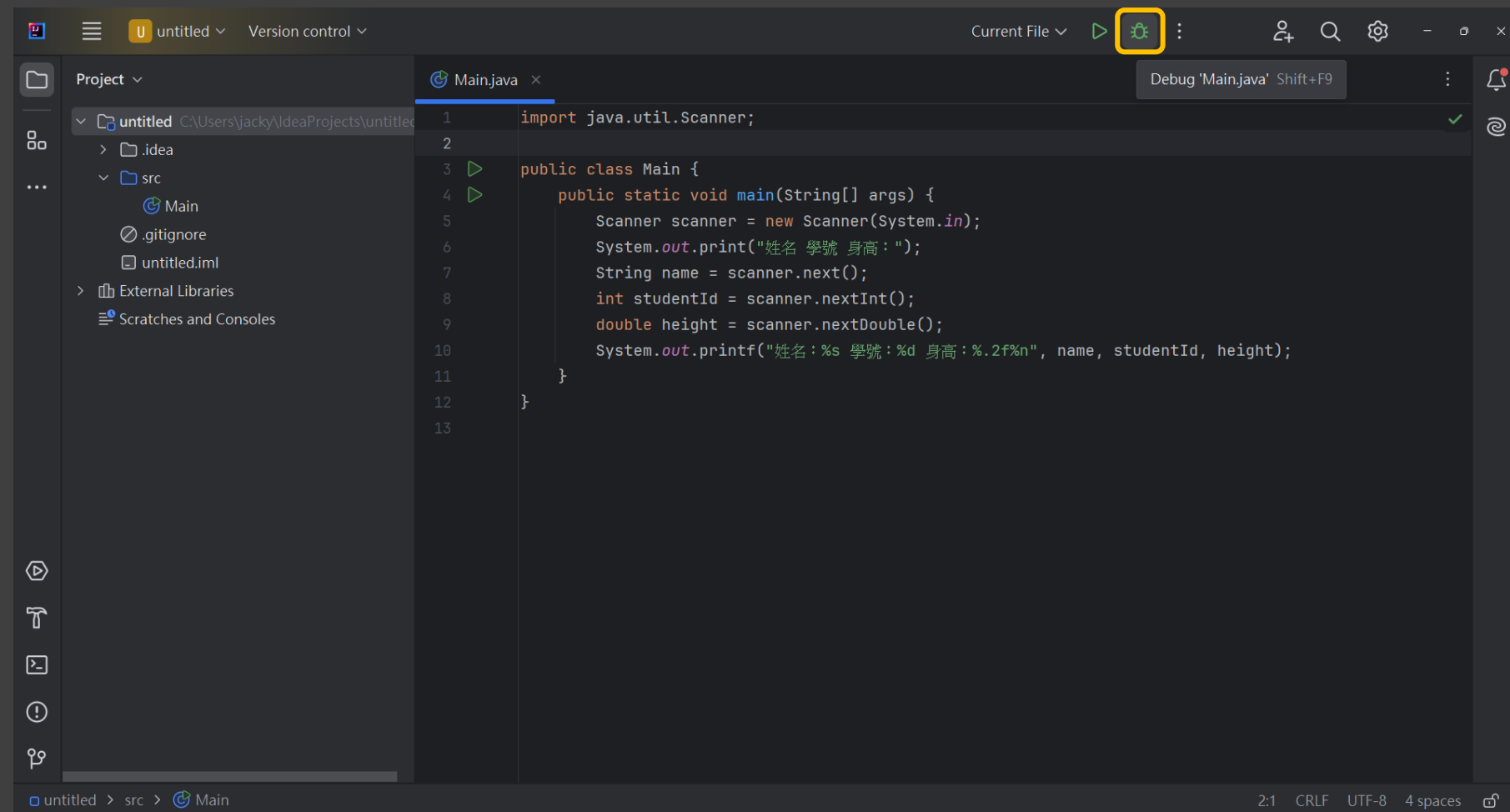
按下右上角的執行按鈕
即可編譯與執行
無需自行輸入指令

執行的輸入和輸出在下方的主控台
按下停止按鈕可停止執行
按下重啟按鈕可重新執行



除錯

按下右上角的
除錯按鈕
即可開始進行除錯



除錯

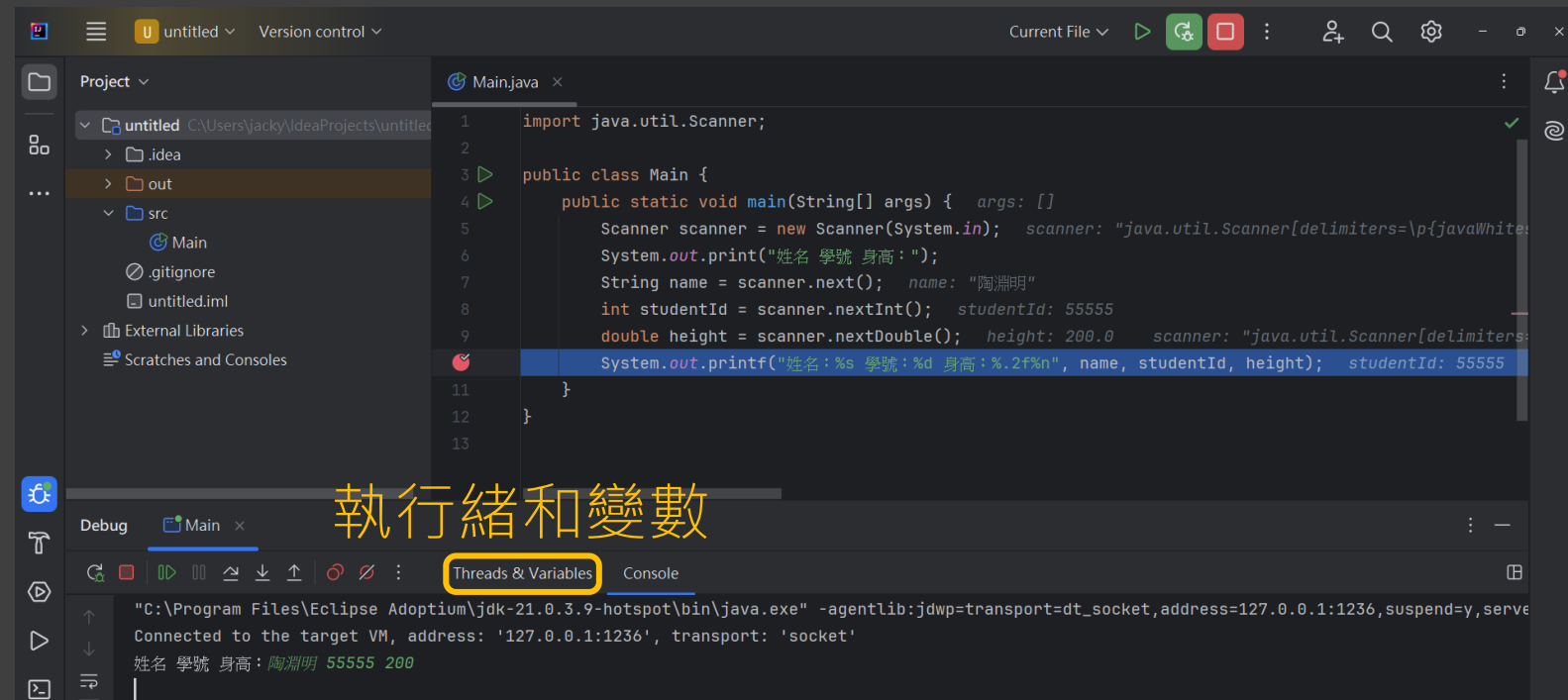
除錯(debug)比一般執行多的功能在於可以下斷點(breakpoint)
程式在下斷點後，當執行到下斷點的行前，就會先暫停
只需要在行編號上點擊左鍵即可下斷點，再按一次即可移除斷點



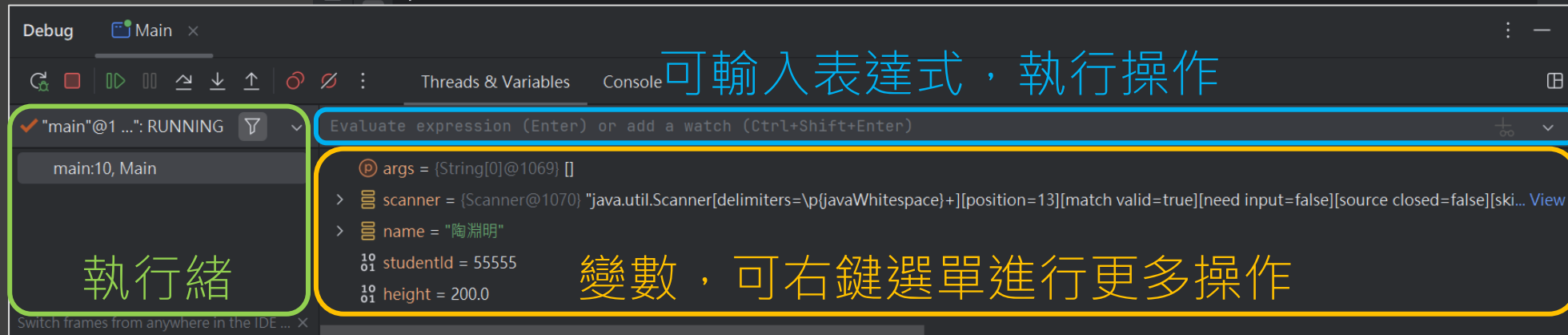
```
1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          System.out.print("姓名 學號 身高:");
7          String name = scanner.next();
8          int studentId = scanner.nextInt();
9          double height = scanner.nextDouble();
10         System.out.printf("姓名:%s 學號:%d 身高:%.2f\n", name, studentId, height);
11     }
12 }
```

除錯

當程式停下後
即可進行許多操作
如查看、修改變數的值
也可以移除或加新斷點



執行緒和變數



可輸入表達式，執行操作

執行緒

變數，可右鍵選單進行更多操作

除錯

若想要繼續執行，可以選擇：

恢復(resume)、步過(step over)、
步入(step in)、步出(step out)等

恢復就是程式繼續執行

步過就是執行該行，然後繼續暫停

步入與步出在更複雜的程式碼才能體現效果，之後會介紹

另外，還可以暫時忽略所有斷點

