# **測試環境設定**

# **基本 Java 程式設計與 AI 協作**

* View(檢視)->Command Palette(命令選擇區)..(按Ctrl + Shift + P)
* `
* Java:Create Java Project → Maven → No Archetype…
* →com.example→ sample2

## 

## **寫簡單 Java 程式**



## 

## **測試範例**

撰寫一個簡單的 Java 程式，計算數字的階乘，並觀察 Copilot 如何協助產生程式碼。

**提示：**

Java

// 寫一個方法計算數字的階乘

// 寫一個方法計算數字的階乘

package com.example;

public class Main {

public static void main(String[] args) {

int number = 5; // 範例數字

long result = factorial(number);

System.out.println("The factorial of " + number + " is: " + result);

}

// 計算階乘的方法

public static long factorial(int n) {

if (n < 0) {

throw new IllegalArgumentException("Number must be non-negative.");

}

long fact = 1;

for (int i = 1; i <= n; i++) {

fact \*= i;

}

return fact;

}

}

//解說:

// 這段程式碼定義了一個計算階乘的方法 `factorial`，並在 `main` 方法中示範如何使用它。階乘是指從 1 乘到該數字的所有正整數的乘積。例如，5 的階乘是 1\*2\*3\*4\*5 = 120。

// 如果輸入的數字是負數，方法會拋出一個非法參數異常。

// 你可以修改 `number` 變數的值來計算其他數字的階乘。

// 這個程式使用迴圈來計算階乘，並返回結果。;

// 最後在控制台輸出結果

**Copilot 產生：** Copilot 會自動產生 factorial 方法與 main 方法，減少撰寫樣板程式碼的時間。透過這種小型程式測試，可確保 Copilot 設定正確，並準備好處理更複雜的任務。

Copilot 能快速產生常見任務的程式碼，例如計算、字串處理等。以下為判斷質數的範例：

**提示：**

Java

// 寫一個 Java 方法判斷數字是否為質數

Main.java

package com.example;

// 寫一個 Java 方法判斷數字是否為質數

public class Main {

public static void main(String[] args) {

int number = 29; // 範例數字

if (isPrime(number)) {

System.out.println(number + " 是質數");

} else {

System.out.println(number + " 不是質數");

}

}

public static boolean isPrime(int n) {

if (n <= 1) {

return false; // 小於等於1的數字不是質數

}

for (int i = 2; i <= Math.sqrt(n); i++) {

if (n % i == 0) {

return false; // 找到因數，非質數

}

}

return true; // 沒有找到因數，是質數

}

}

//解說:

// 這段程式碼定義了一個名為 isPrime 的方法，用來判斷一個整數是否為質數。質數是指大於1且只能被1和自身整除的數字。方法首先檢查輸入的數字是否小於或等於1，如果是，則返回 false。接著，使用一個 for 迴圈從2開始檢查到該數字的平方根，若在此範圍內找到任何能整除該數字的因數，則返回 false。若迴圈結束後沒有找到任何因數，則返回 true，表示該數字是質數。在 main 方法中，我們測試了這個方法並輸出結果。

// 你可以修改 number 變數來測試其他數字。

// 例如，將 number 設為 15，程式會輸出 "15 不是質數"。

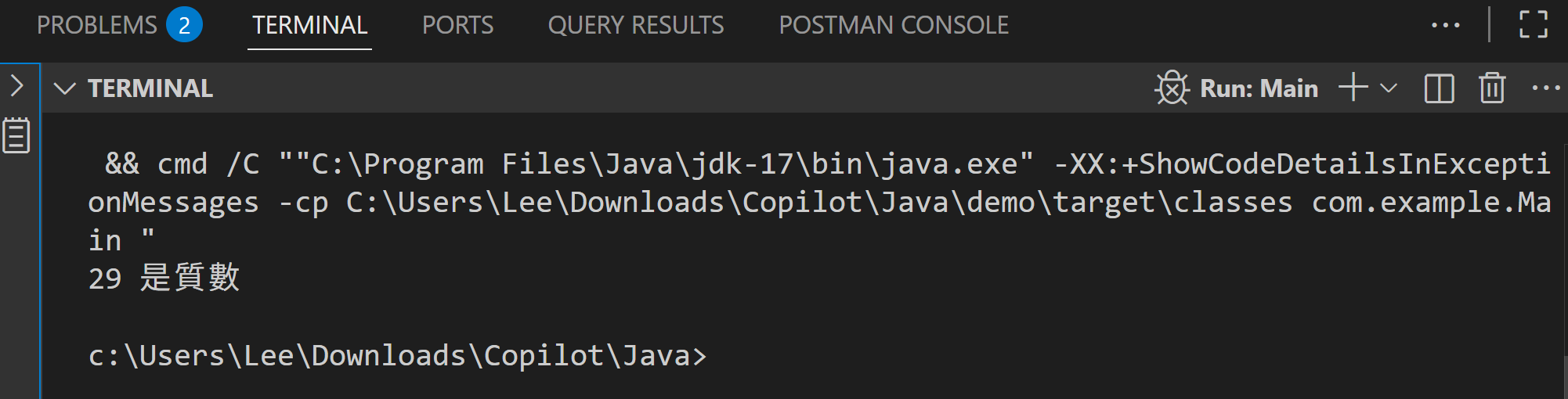
// 將 number 設為 13，程式會輸出 "13 是質數"。

// 這樣你就可以輕鬆判斷任意整數是否為質數了。

**Copilot 產生：** Copilot 會產生 isPrime(int n) 方法，並優化效率，例如提前排除 1 與 2、3 的倍數，並使用 6k±1 規則減少不必要的迴圈。main 方法則示範如何呼叫並印出結果。

Main.java 程式碼位置 → 右鍵 →Run Java

## 



# **使用 Java 集合與串流**

## **範例一：過濾使用者年齡**

**提示：**

Java

// 寫一個方法，從 User 物件清單中過濾出符合年齡條件的使用者

package com.example;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

// 寫一個方法，從 User 物件清單中過濾出符合年齡條件的使用者

class User {

private String name;

private int age;

public User(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

public class Main {

public static List<User> filterUsersByAge(List<User> users, int minAge) {

return users.stream()

.filter(user -> user.getAge() >= minAge)

.collect(Collectors.toList());

}

public static void main(String[] args) {

List<User> users = new ArrayList<>();

users.add(new User("Alice", 30));

users.add(new User("Bob", 20));

users.add(new User("Charlie", 25));

List<User> filteredUsers = filterUsersByAge(users, 25);

for (User user : filteredUsers) {

System.out.println(user.getName() + " - " + user.getAge());

}

}

}

//解說:

// 這段程式碼定義了一個 User 類別，包含使用者的名稱和年齡屬性。Main 類別中有一個靜態方法 filterUsersByAge，

// 該方法接受一個 User 物件清單和一個最小年齡參數，並使用 Java Stream API 過濾出年齡大於或等於指定值的使用者。

// main 方法中創建了一個 User 清單，並呼叫 filterUsersByAge 方法來獲取符合條件的使用者，最後將結果列印出來。

// 執行結果將顯示年齡大於或等於 25 歲的使用者名稱和年齡。

// 執行結果:

// Alice - 30

// Charlie - 25

**Copilot 產生：**

* 使用 stream() 與 filter()，快速過濾出年齡大於等於指定值的使用者。
* 產生 User 類別與 filterUsersByAge 方法，並在 main 方法示範如何使用。

Main.java 程式碼位置 → 右鍵 →Run Java

## 

## 

## **範例二：產生隨機產品名稱**

**提示：**

Java

// 寫一個方法，隨機產品名稱清單,產品要有ID,名稱,價格屬性

// 寫一個方法，隨機產品名稱清單,產品要有ID,名稱,價格屬性

package com.example;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

public class Main {

public static void main(String[] args) {

List<Product> products = generateRandomProducts(10);

for (Product product : products) {

System.out.println(product);

}

}

public static List<Product> generateRandomProducts(int count) {

List<Product> products = new ArrayList<>();

Random random = new Random();

String[] sampleNames = {"Laptop", "Smartphone", "Tablet", "Monitor", "Keyboard", "Mouse", "Printer", "Camera", "Headphones", "Speaker"};

for (int i = 0; i < count; i++) {

int id = i + 1;

String name = sampleNames[random.nextInt(sampleNames.length)];

double price = 50 + (500 - 50) \* random.nextDouble(); // Price between 50 and 500

products.add(new Product(id, name, Math.round(price \* 100.0) / 100.0)); // Round to 2 decimal places

}

return products;

}

}

class Product {

private int id;

private String name;

private double price;

public Product(int id, String name, double price) {

this.id = id;

this.name = name;

this.price = price;

}

@Override

public String toString() {

return "Product{" +

"id=" + id +

", name='" + name + '\'' +

", price=" + price +

'}';

}

}

//解說

// 這個程式碼定義了一個產品類別，包含ID、名稱和價格屬性，並提供了一個方法來生成隨機產品清單。

// 主方法中調用該方法生成10個隨機產品並打印出來。價格在50到500之間隨機生成，並四捨五入到小數點後兩位。

// 產品名稱從預定義的樣本名稱中隨機選取。

// 這樣可以用於測試或模擬產品數據。

**Copilot 產生：**

* 使用形容詞與名詞陣列，隨機組合產生產品名稱。
* 產生 generateRandomProductNames 方法，並在 main 方法示範如何使用。

