

## 讀書會

- 組員：龍品勳、張耀文、時顯文
- 討論時間：2024/11/11 10:00 AM
- 地點：圖書館

照片：



# 心得報告

- 心得：之前有學過 C++ 的物件導向，所以 Java 的學起來感覺和 C++ 還滿像的。
- ChatGPT：
  1. 請問 public 和 protected 和 private 的區別？

在物件導向程式設計中（例如 C++、Java 和 Python），`public`、`protected` 和 `private` 是存取控制修飾詞（Access Modifiers），用來控制類別成員（屬性和方法）的存取範圍。這三者的差異如下：

## 1. `public`

- 描述：類別的 `public` 成員可以被任何地方存取，包括同一個類別、子類別（繼承的類別）以及類別外部的其他程式碼。
- 應用場景：用於想讓所有程式碼都能夠存取的成員，例如公共方法或屬性。

## 2. `protected`

- 描述：`protected` 成員僅能被該類別及其子類別（即使是在不同的 package/namespace 中）存取。外部程式碼無法直接存取。
- 應用場景：用於希望子類別能夠存取和使用的成員，但不希望完全公開給外部使用。常見於希望在繼承關係中保留某些可調整的屬性。

## 3. `private`

- 描述：`private` 成員只能在類別內部使用，外部和子類別都無法直接存取。
- 應用場景：用於限制成員的存取範圍，避免外部程式碼或子類別直接操作或依賴這些成員，通常是一些內部使用的變數或方法。

## 總結比較

存取控制修飾詞	本類別內部	子類別	其他類別
<code>public</code>	可以	可以	可以
<code>protected</code>	可以	可以	不可以
<code>private</code>	可以	不可以	不可以

## 注意

不同語言的細節可能略有不同。例如，Python 沒有強制的存取修飾詞，但可以使用前置底線（`_`）或雙底線（`__`）來表示「受保護」或「私有」屬性，作為提示給開發者。

🔊 📄 📌 🔗 ↺

2. 為什麼 Java 不支援運算子重載？



Java 不支援運算子重載，主要是出於語言設計的簡化與穩定性考量。以下是 Java 不支援運算子重載的幾個主要原因：

### 1. 語言簡單性與一致性

- Java 的設計理念是要提供一個簡單、一致的語言結構。運算子重載會讓語言變得更複雜，因為同一運算子在不同類別中可能具有不同的行為。這種情況容易讓程式碼變得難以閱讀和理解。
- 例如，如果每個類別都可以重載 `+`、`-` 或 `*` 等運算子，可能會出現大量不同的實作方式，使得程式碼在維護時難以追蹤每個運算子的真正行為。

### 2. 降低錯誤風險

- 運算子重載在某些情況下容易引發意外的行為，特別是在大型專案中。重載的運算子如果被誤用，可能會導致難以發現的邏輯錯誤。Java 的設計者選擇避免這種潛在錯誤風險，透過不支援運算子重載來提高程式的穩定性。
- 例如，`+` 在數學上通常表示加法運算，但如果重載後表示其他意涵（例如字串拼接），會增加調試和理解程式碼的難度。

### 3. 清晰的意圖

- Java 鼓勵開發者使用具有描述性的方法名稱來明確意圖，而非運算子重載。例如，可以定義 `add()`、`subtract()` 這樣的方法來進行特定操作，而不是重載 `+` 或 `-` 運算子。這樣的命名方式使程式碼意圖更清晰，也更便於維護。
- 對於新加入專案的開發者而言，具有明確方法名稱的程式碼更易於理解，而不是依賴運算子進行多義操作。

### 4. 避免效能問題

- 在某些情況下，運算子重載會增加效能開銷。為了支援運算子重載，編譯器和虛擬機器需要進行額外的處理，這可能會降低程式的執行效率。Java 選擇簡化這些操作，使虛擬機器更專注於提升效能，而非處理多義性的運算子操作。

### 總結

Java 的設計者認為，運算子重載會增加語言的複雜性和錯誤風險，且與 Java 的簡潔和穩定性原則不符。因此，Java 不支援運算子重載，鼓勵開發者使用具描述性的名稱來實現類似功能。

🔊 📄 🌟 📌 🔄

## 3. 請問 extends 在 Java 中的用法?



在 Java 中，`extends` 關鍵字用於類別繼承，讓子類別（subclass）從父類別（superclass）繼承屬性和方法。透過繼承，子類別可以重用父類別的程式碼，並能夠進行方法的重寫（override）以實現自己的行為。

## `extends` 用法

### 1. 基本語法

java

複製程式碼


```
class ParentClass {  
    // 父類別的成員變數和方法  
}  
  
class ChildClass extends ParentClass {  
    // 子類別可以使用父類別的成員，並且可以新增自己的成員  
}
```

### 2. `extends` 的特點

- 子類別會繼承父類別的所有 `public` 和 `protected` 成員（屬性和方法）。
- 子類別無法繼承父類別的 `private` 成員，但可以透過 `getter` 或 `setter` 間接存取。
- 子類別可以透過 **方法重寫（override）** 修改父類別的行為。
- Java 僅支援 **單一繼承**，即一個子類別只能有一個直接父類別，這是為了避免多重繼承帶來的複雜性。

## 範例：使用 `extends` 繼承

java

 複製程式碼

```
// 父類別
class Animal {
    String name;

    public Animal(String name) {
        this.name = name;
    }

    public void sound() {
        System.out.println("Some generic animal sound");
    }
}

// 子類別
class Dog extends Animal {
    public Dog(String name) {
        super(name); // 呼叫父類別的建構子
    }

    // 重寫父類別的方法
    @Override
    public void sound() {
        System.out.println(name + " says: Woof!");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal myAnimal = new Animal("Generic Animal");
        Dog myDog = new Dog("Buddy");

        myAnimal.sound(); // 輸出: Some generic animal sound
        myDog.sound();    // 輸出: Buddy says: Woof!
    }
}
```

## super 關鍵字

在子類別中，`super` 關鍵字可以用來呼叫父類別的建構子或方法：

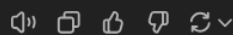
- 呼叫父類別建構子：使用 `super()` 呼叫父類別的建構子，通常在子類別的建構子中使用，以初始化從父類別繼承的屬性。
- 呼叫父類別方法：若子類別重寫了父類別的方法，仍可以透過 `super.methodName()` 呼叫父類別的原方法。

## 注意事項

- 子類別可以不重寫父類別的方法，若沒有重寫，則直接使用父類別的實現。
- 子類別可以新增屬性和方法，但無法縮小父類別成員的存取控制範圍。

## 總結

`extends` 是 Java 中用來實現類別繼承的關鍵字，允許子類別從父類別繼承屬性和方法，並通過方法重寫來實現多態行為。



- GitHub 程式連結：<https://github.com/WoemTheCat/CH3>
- GitHub 的截圖：

