

# CH2

111360138 電子三甲 蔣安聖

## 讀書會

組員：

111360127 林煒哲

111360137 許鎧晏

111360138 蔣安聖

討論時間：2024/10/27 晚上 10:00

地點:Discord



## 詢問 GPT 問題

1. `int[] intArray = (int[]) Array.newInstance(int.class, 3);` 用來做什麼？

在這段程式碼中：

```
java
int[] intArray = (int[]) Array.newInstance(int.class, 3);
```

這行程式碼的作用是利用 Java 的反射機制動態地創建一個整數陣列 `intArray`。以下是具體解釋：

1. `Array.newInstance(int.class, 3);`
  - `Array.newInstance` 是 Java 反射 API 中的一個方法，用來創建陣列實例。
  - `int.class` 指定了要創建的陣列的元素類型，這裡是 `int`。
  - `3` 指定了陣列的長度（此處是 3），所以 `intArray` 最終是一個長度為 3 的 `int` 陣列。
2. `(int[])`
  - `Array.newInstance` 返回一個 `Object` 類型的物件，因此需要將返回的物件轉型為 `int[]` 類型，這樣才能進行後續的操作。

### 程式效果

這行程式碼的效果等同於直接宣告 `int[] intArray = new int[3];`，但它是透過反射機制動態創建的。

### 反射用途

反射方式適用於在執行期決定類型的情況，可以在不明確指定類型的前提下創建陣列或執行操作。在一般情況下，直接使用 `new int[3]` 更為簡單。

🔗 📄 🗣️ 🔄

`int[] intArray = (int[]) Array.newInstance(int.class, 3);` 是利用 java 中的反射機制動態創建陣列，該種宣告方式其結果與 `int[] intArray = new int[3];` 相同，不過創建的方法不同。

## 2. 細說 Java 的反射機制



Java 的反射機制是一組用來在執行期動態檢查和操作類別、方法、屬性等元件的 API。反射允許程式在執行時對類別的結構進行檢查和操作，例如讀取屬性、調用方法，甚至動態創建物件。這使得反射在需要靈活的動態行為或框架構建中非常有用，例如框架（如 Spring）和序列化工具。

### 1. 反射的基礎類別

Java 提供了幾個核心類別和介面來支持反射機制：

- **Class**：表示類或介面的結構，可以獲取類名、父類、介面、方法、屬性等資訊。
- **Method**：表示類的方法，可以用來調用方法或取得方法資訊。
- **Field**：表示類的屬性，可用來讀寫屬性值或取得屬性相關的資訊。
- **Constructor**：表示類的構造方法，可用來創建類的新實例。
- **Array**：用於創建和操作陣列。

### 2. 如何獲取 Class 物件

獲取 `Class` 物件是反射操作的第一步，可以透過以下三種方式獲取：

```
java
// 方式 1：透過類的靜態屬性 "class"
Class<?> cls = MyClass.class;

// 方式 2：使用 "getClass()" 方法
MyClass obj = new MyClass();
Class<?> cls = obj.getClass();

// 方式 3：使用 "Class.forName()"，通常用於動態載入類
Class<?> cls = Class.forName("com.example.MyClass");
```

### 3. 反射的常見操作

#### 1. 檢查類別的基本資訊

可以用反射取得類名、父類、介面等資訊。

```
java
System.out.println("類別名稱：" + cls.getName());
System.out.println("父類名稱：" + cls.getSuperclass().getName());
```

#### 2. 取得並操作屬性

使用 `getDeclaredFields()` 方法獲取所有屬性，並使用 `Field` 類來讀取和寫入屬性值。

```
java
Field field = cls.getDeclaredField("name");
field.setAccessible(true); // 這為私有屬性設定可訪問
field.set(obj, "新值"); // 設置屬性值
System.out.println(field.get(obj)); // 讀取屬性值
```

#### 3. 取得並調用方法

使用 `getDeclaredMethods()` 方法獲取類別的所有方法，並使用 `Method.invoke()` 來動態調用方法。

```
java
Method method = cls.getDeclaredMethod("sayHello");
method.setAccessible(true);
method.invoke(obj); // 調用該方法
```

### 4. 動態創建物件

可以使用 `Constructor` 類來創建物件，或使用 `newInstance()` 方法直接創建無參數構造的物件。

```
java
Constructor<?> constructor = cls.getConstructor(String.class);
Object obj = constructor.newInstance("參數");
```

### 4. 反射的應用場景

- 框架設計：Spring 等框架使用反射來掃描注解、創建物件並注入依賴。
- 序列化與反序列化：將物件轉換為 JSON 或 XML。
- 動態代理：生成代理類，例如 JDK 動態代理。
- 工具類庫：如單元測試框架 (JUnit) 用來執行標註了特定注解的方法。

### 5. 反射的缺點

- 效能開銷：反射操作較慢，會影響執行效率。
- 安全性：可能繞過訪問控制檢查，帶來安全風險。
- 不安全性：反射在編譯時無法檢查錯誤，容易在執行時發生錯誤。

### 總結

反射機制是一種強大的動態操作工具，提供了靈活的操作類、方法、屬性和陣列的方式。在編寫需要動態行為的框架或通用工具時非常有用，但應謹慎使用，避免對效能和安全性造成影響。

🔗 📄 📄 📄 📄

Java 的反射機制有點複雜，總得來說是一種可在程式執行時動態操作的工具，給予開發者更多的自由。

### 3. Arrays.sort(test, Collections.reverseOrder());是什麼意思?



在這段程式碼中：

```
java
Arrays.sort(test, Collections.reverseOrder());
```

這行程式的作用是將 `test` 陣列按照降序排序。以下是具體的解析：

1. `Arrays.sort(test, Collections.reverseOrder());`

- `Arrays.sort()` 是 Java 標準庫中的一個方法，用於對陣列進行排序。
- `test` 是要排序的陣列，此處是 `Integer[]` 類型。Java 中的基本型別（如 `int`）無法直接使用 `Collections.reverseOrder()`，所以此處使用了 `Integer`（即 `Integer[]`）。
- `Collections.reverseOrder()` 是一個比較器（Comparator），用於指定排序的順序。
  - 當它作為參數傳入 `Arrays.sort()` 時，告訴排序方法使用降序排序。
  - 如果不傳入 `Collections.reverseOrder()`，預設會按升序排序。

#### 總結

這行程式的作用是對 `test` 陣列中的數字進行降序排序，讓分數由高到低排列。

🔊 📄 📌 🔗 ↺

`Arrays.sort()` 是 java 中標準函式庫中的一個函式，當我們 import `java.util.Arrays` 就可使用該函式，而 `Collections.reverseOrder()` 則是從名為 `java.util.Collections` 的函式庫來的，其功能為比較器，該範例中使用降序排序。

## 心得

這周我們繼續學習 Java，在這次的課程中，我們學到基礎的迴圈以及邏輯判斷，while for do-while if switch 等，基本上與 C 語言相同，唯獨在陣列宣告上，有一個很特別的方式，是使用 java 反射機制來宣告，雖然我去查了一下什麼是反射機制，但依然不太了解，我可能要先去學習物件導向的概念，以及程式執行時的各種事件，例如資料回收、生存時間等概念再來了解反射機制會比較好。我覺得在學習 java 的過程中，不太像 C 語言那樣如此的接近硬體，相對來說比較不會回收機制等概念需要了解，我自己是學 C 語言的，我感覺 C 對我來說比 java 好懂，尤其是回收機制等複雜的概念我就真的不了解了。

# GitHub 程式連結與截圖

<https://github.com/DolphinBlast/JavaHW04>

