UML 之 Sequence Diagram

Sam Xiao, Nov.20, 2017

Overview

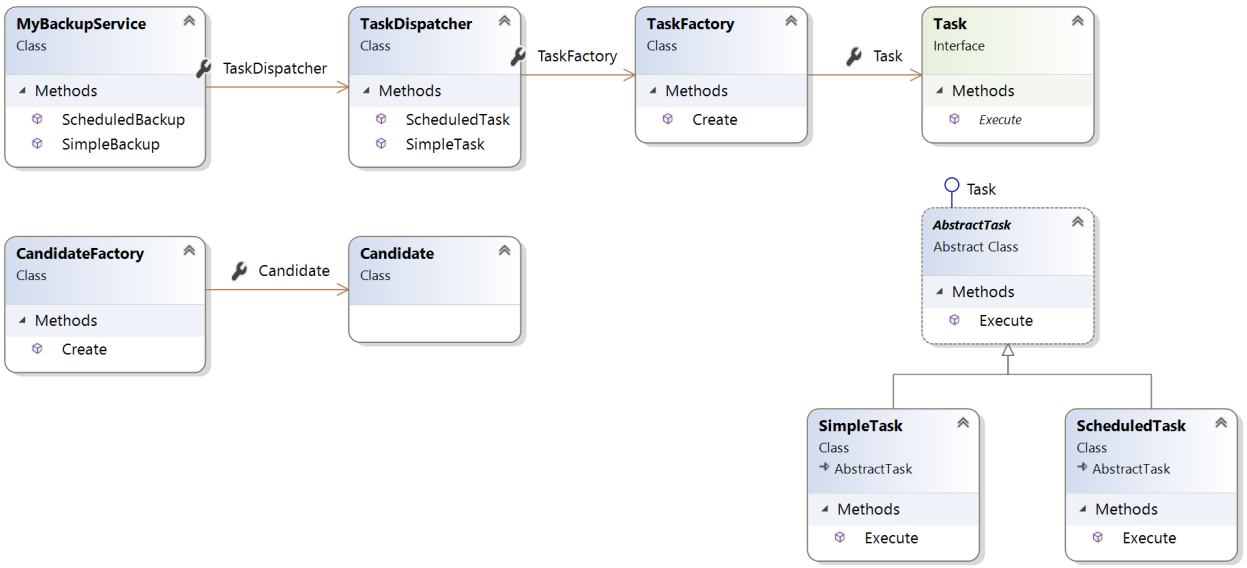
最常用來輔助 OOP 的是 class diagram，讓我們以更宏觀的角度來看 class 與 class 之間的關係。

但由 class diagram 我們看不出 method 間是如何互動的，因此還要搭配 sequence diagram 加以輔助。

Outline

UML 之 Sequence Diagram
- Overview
- Outline
- User Story
- Task
- Sequence Diagram
  - Actor
  - Lifeline
  - Participants
  - Synchronous
  - Creation
  - Reply
  - Activation
  - Asynchronous
- Summary
- Conclusion

User Story



在 homework 5，我們已經重構出 `MyBackup` 的大架構，但 user 仍然不知道 class 間是如何互動。

Task

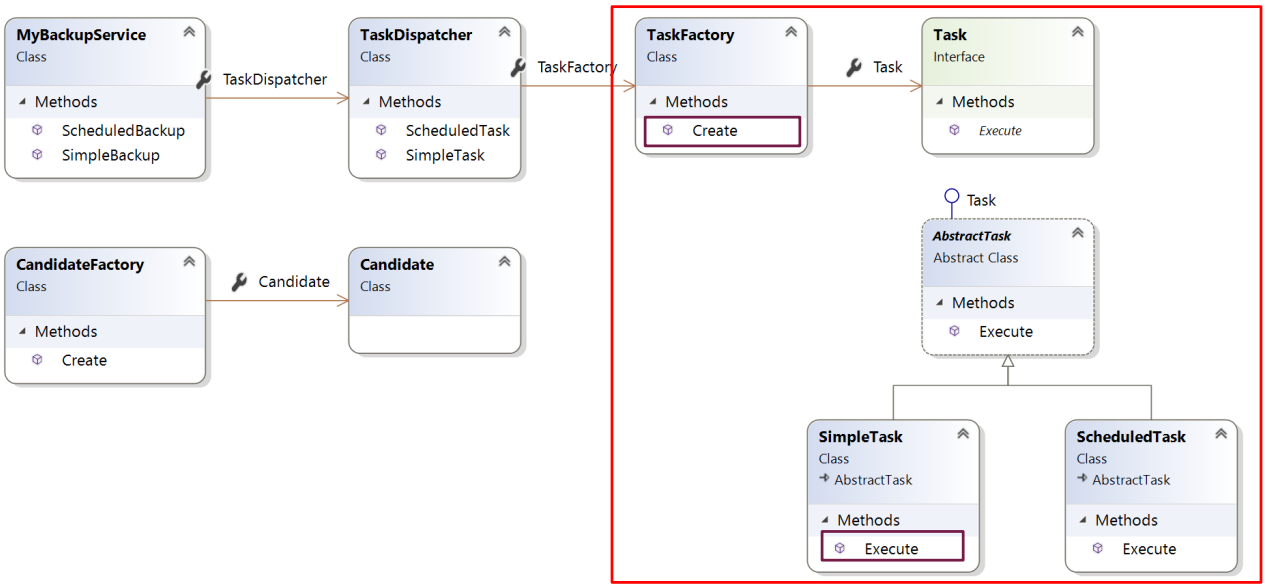
因此我們想補上 sequence diagram，詳細描述 class 間的互動。

Sequence Diagram

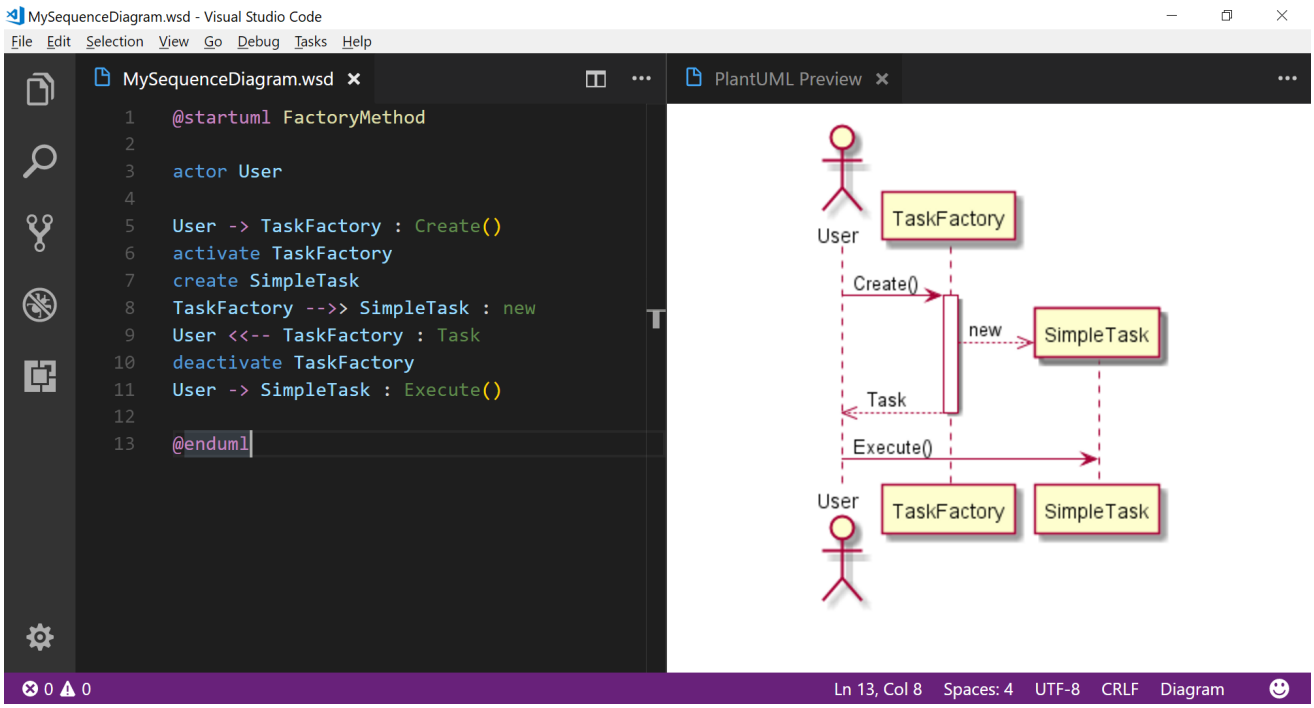
> 描述 class 間 method 的互動關係

Sequence diagram 不是用來描述 `演算法` 與 `流程`。

> UML 心法
>
> 演算法與流程該使用 activity diagram 或 flowchart，不該使用 sequence diagram

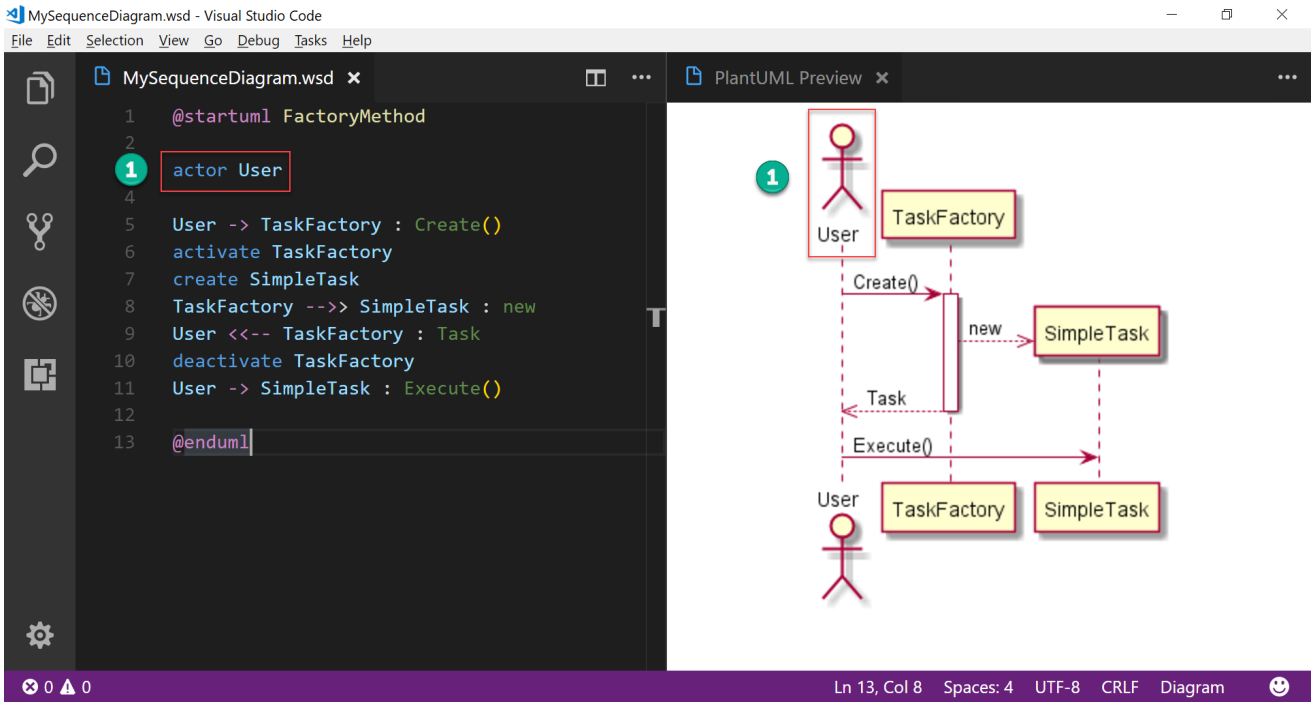



```
1 Task simpleTask = TaskFactory.Create("simple");
2 simpleTask.Execute();
```



這是個最簡單的 sequence diagram，用來描述 `Factory method` 如何建立 interface 多型的物件。

Actor

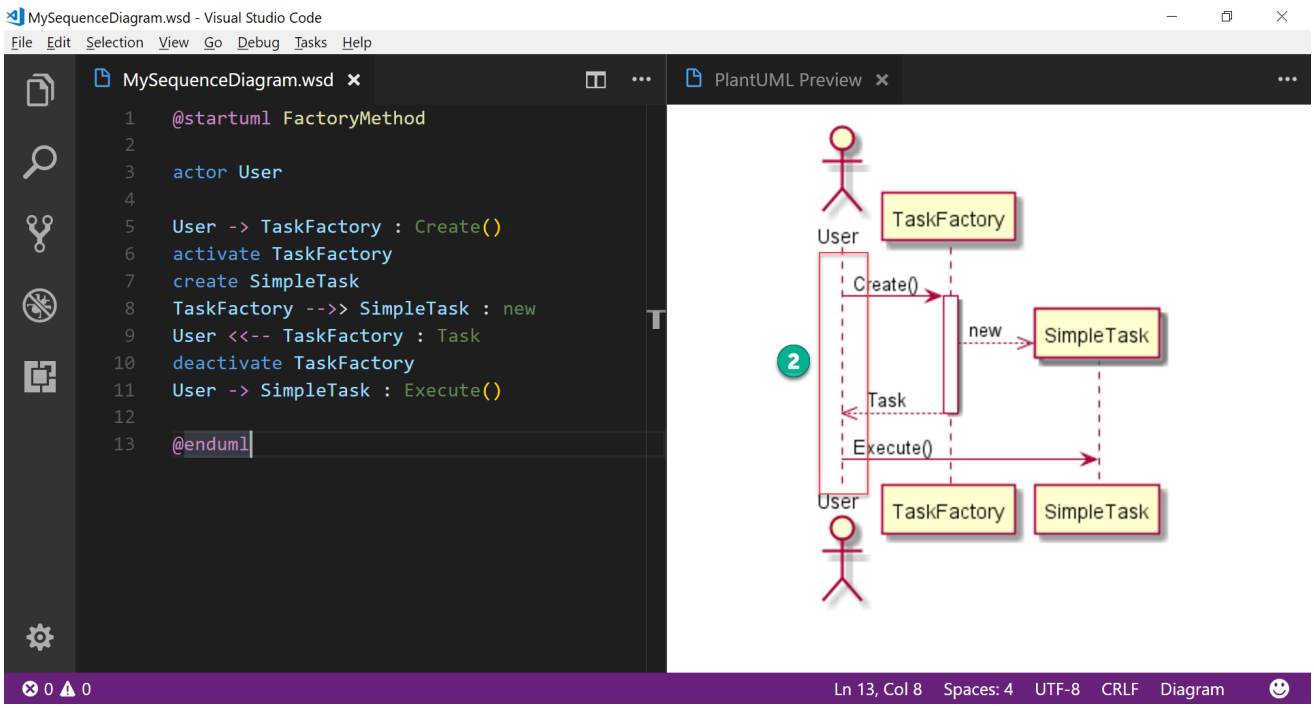


以 `人型物件` 表示。

```
actor User
```

以 `actor` 宣告使用端。

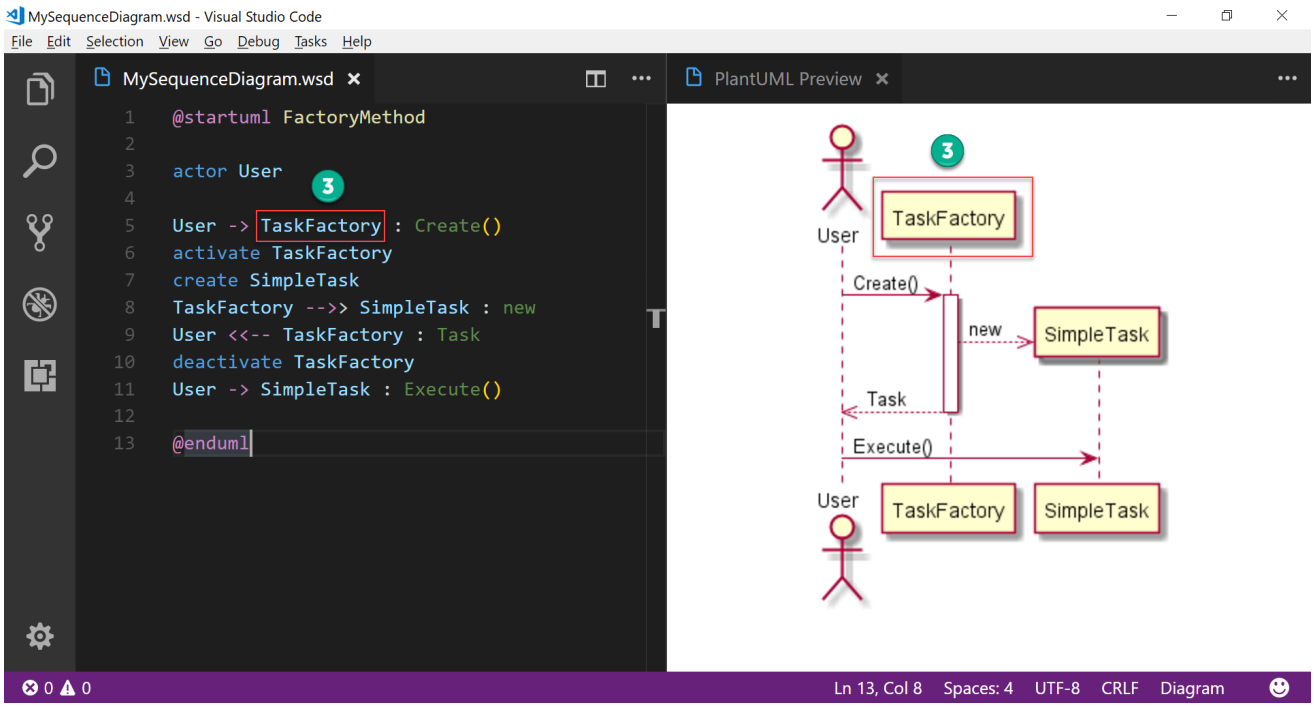
Lifeline



人形物件的 `虛線` ，表示用戶端的生命週期。

PlantUML 不用特別表示。

Participants

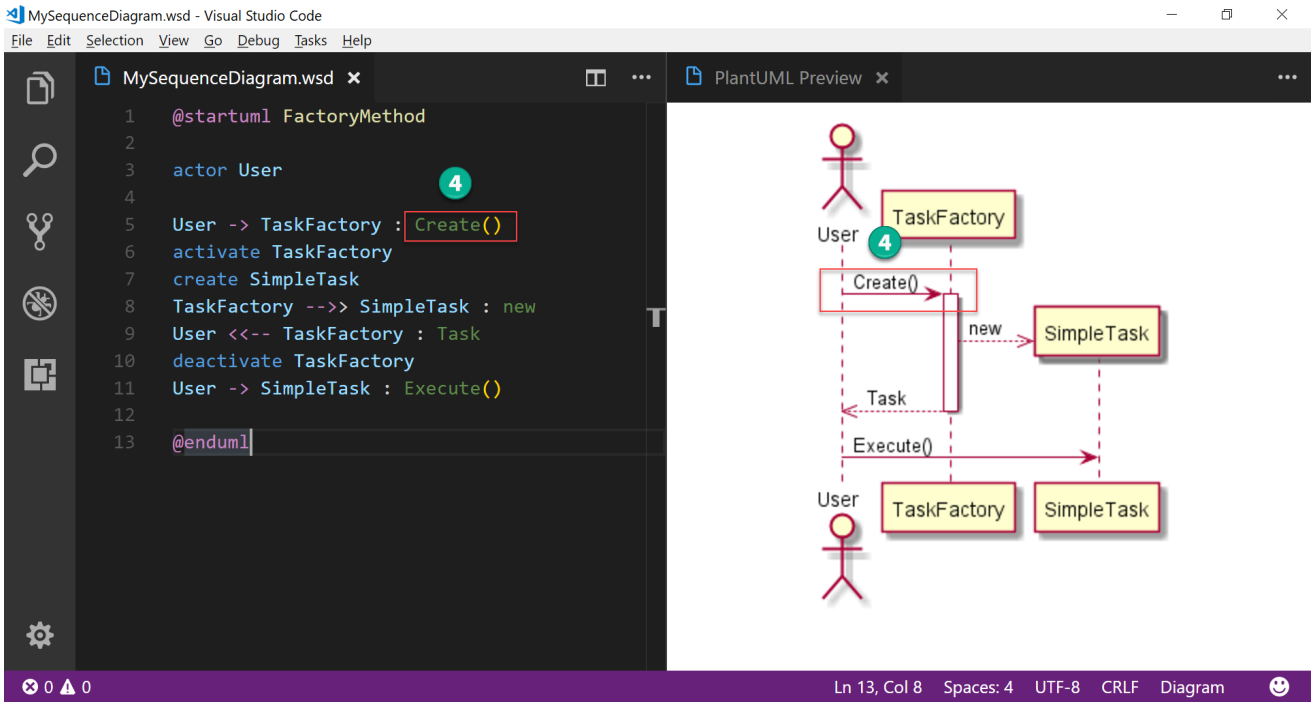


要描述 method 互動的 class。

```
User -> TaskFactory
```

寫在各種箭頭兩側。

Synchronous



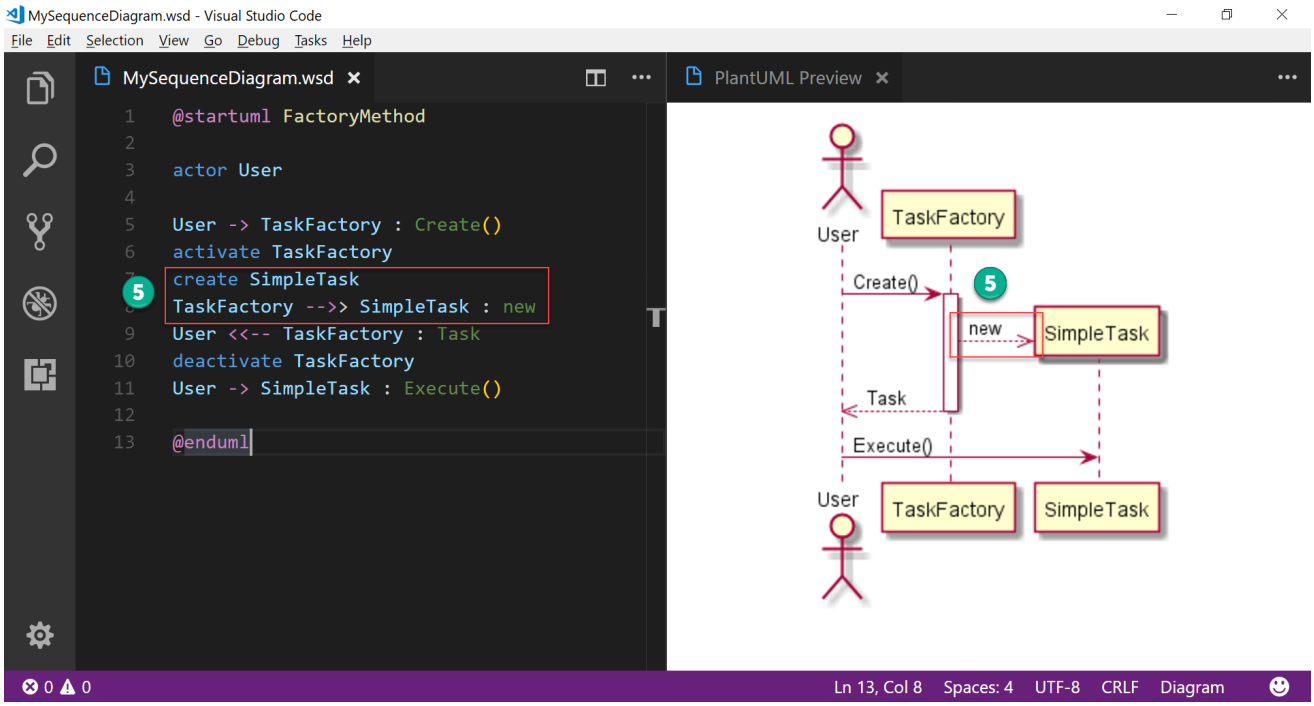
實線 + 實心箭頭 ，上方加上 method 名稱表示要 同步 呼叫的 method。

```
User -> TaskFactory : Create()
```

->; 表示 實線 + 實心箭頭，並在最後加上 : + method 名稱。

> UML 常犯錯誤 ->; 後面接的是 被呼叫端 的 method，而不是 呼叫端 的 method

Creation

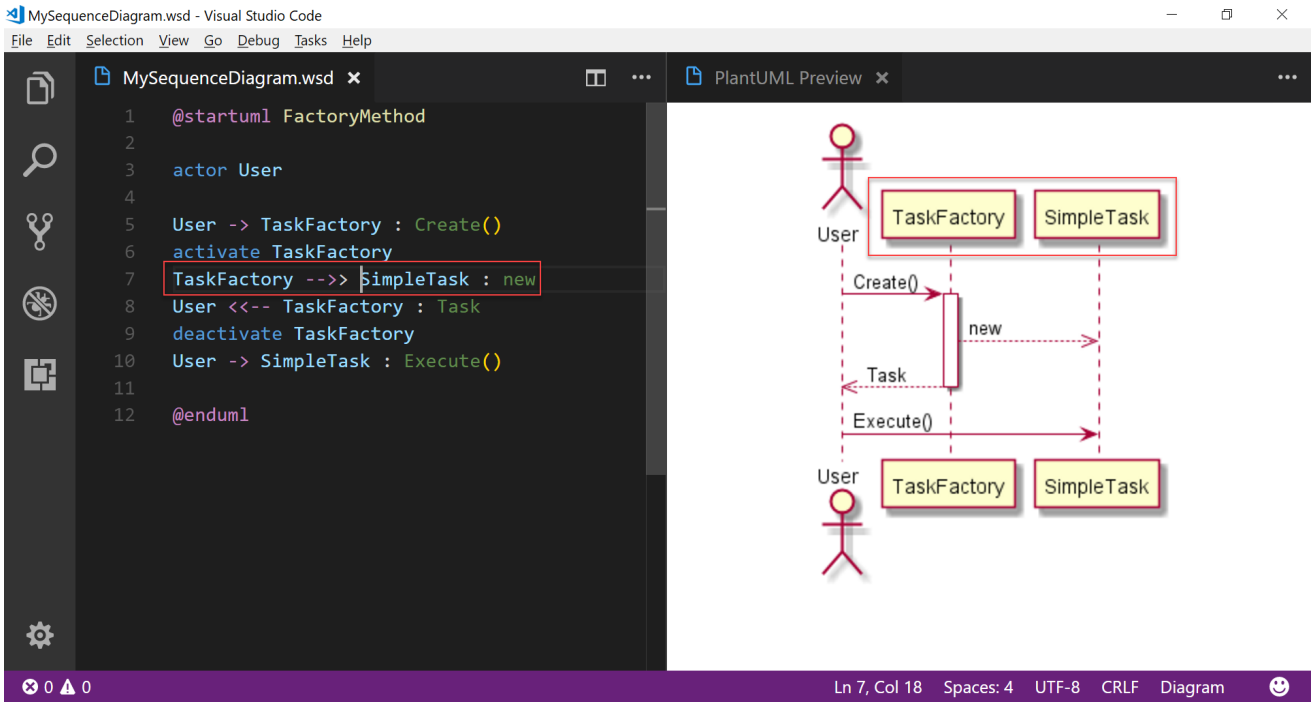


`虛線 + 空心箭頭` ，上方加上 `new` 表示建立物件。

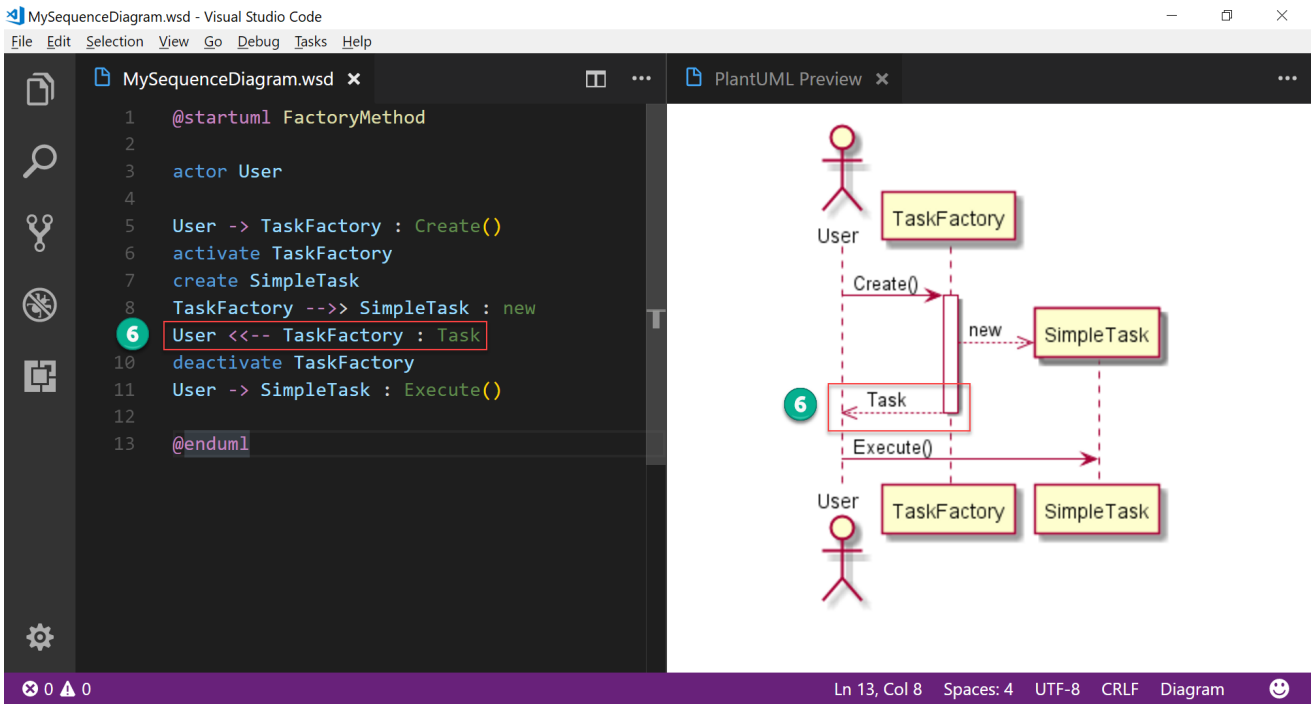
```
create SimpleTask
TaskFactory -->> SimpleTask : new
```

`-->;>;` 表示 `虛線 + 空心箭頭` ，並在最後加上 `: new` 。

> UML 常犯錯誤
>
> create SimpleTask 一定要加，否則 `SimpleTask` 會與 `TaskFactory` 同一列，其 lifeline 就無法展現由 `new` 所產生



Reply

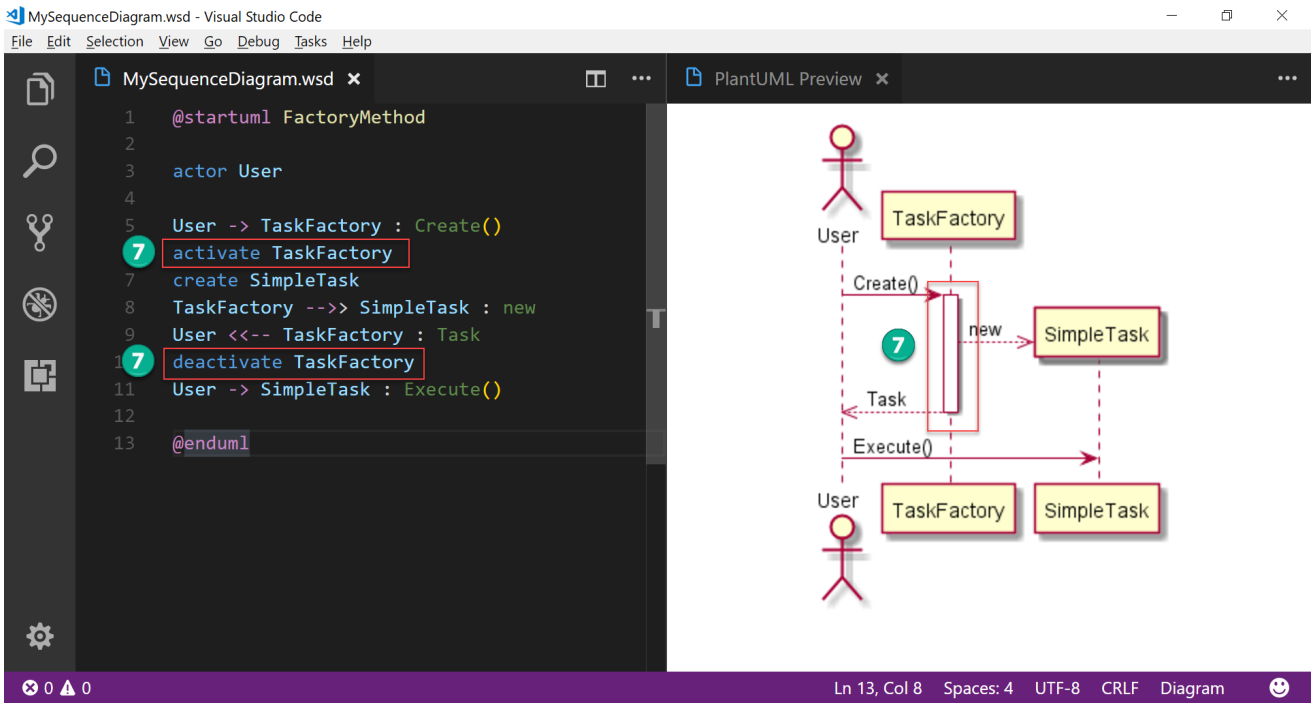


`虛線 + 空心箭頭` ，上方加上 class 表示回傳物件型別。

```
User <<-- TaskFactory : Task
```

`<;<;--` 表示 `虛線 + 空心箭頭` ，並在最後加上 `:` + class 名稱。

Activation



在 class 的 lifeline 加上 `垂直狹長矩形`，代表 method 的生命週期。

> 實務上不常使用 activation，除非 method 執行時間有特別意義 (如執行時間很長)

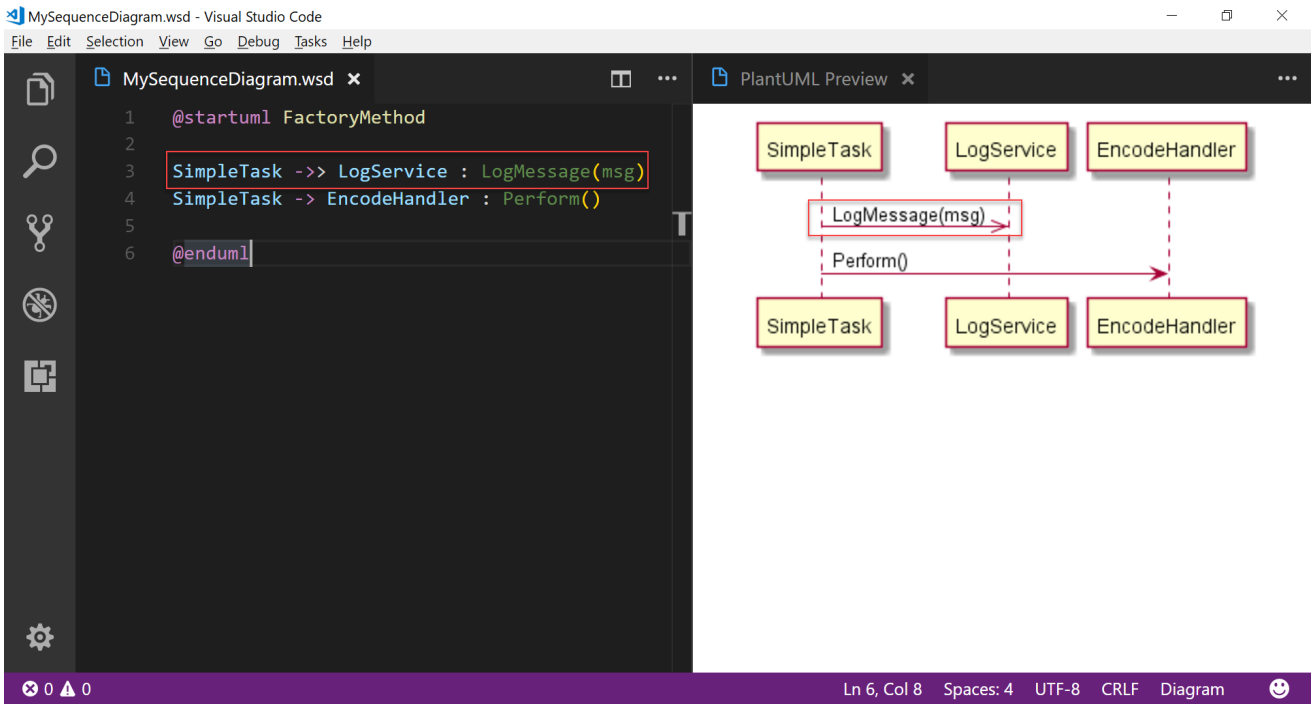
```
User -> TaskFactory : Create()
activate TaskFactory
create SimpleTask
TaskFactory -->> SimpleTask : new
User <<-- TaskFactory : Task
deactivate TaskFactory
```

如要顯示 `TaskFactory.Create()` 的生命週期，在 `User ->; TaskFactory : Create()` 之後加上 `activate TaskFactory`，表示在 `TaskFactory.Create()` 使用 activation。

並在 `User <;<;-- TaskFactory : Task` 回傳 `Task` 物件後，加上 `deactivate TaskFactory` 表示結束 `TaskFactory.Create()` 的 activation。

> UML 心法
>
> 由此範例可發現，原來 C# 只有 2 行程式碼，PlantUML 需要 8 行程式碼，所以實務上若大量的使用 sequence diagram，將很沒有效率也浪費時間，因此只需要在特別展示 class 互動時，才必須使用 sequence diagram

Asynchronous



`實線 + 空心箭頭` ，上方加上 method 名稱表示要 `非同步` 呼叫的 method。

```
SimpleTask ->> LogService : LogMessage(msg)
```

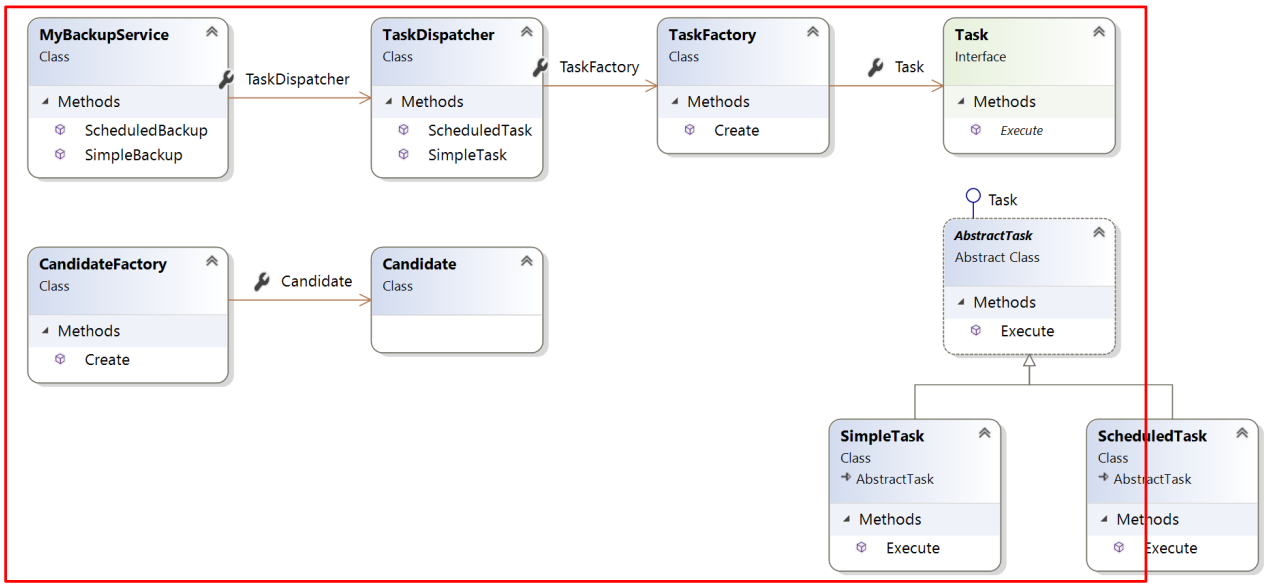
`->;>;` 表示 `實線 + 空心箭頭` ，並在最後加上 `:` + method 名稱。

Summary

- Class diagram 展現 class 間靜態的一面；而 sequence diagram 展現 class 動態的一面

- Sequence diagram 不適合畫得很複雜，因此不能拿來表示 `演算法` 與 `流程`，只能用來表示特定 class 間 method 互動。
- 不要試圖將所有的 method 互動都用 sequence diagram 表示，只要畫有 `代表性` 或 `需要特別解釋` 的 class 間 method 互動，否則會浪費太多時間，也沒有參考價值。
- Design Pattern 書通常都會有 class diagram 與 sequence diagram，也算是看書必備的知識。
- `if else` 與 `foreach()` 都不適合畫在 sequence diagram，會太複雜不方便理解與閱讀。
- `Synchronous`：`->;` 表示 `實線 + 實心箭頭`
- `Asynchronous`：`->;>;` 表示 `實線 + 空心箭頭`
- `Creation`：`-->;>;` 表示 `虛線 + 空心箭頭`，最後加 `: new`
- `Reply`：`<;<;--` 表示 `虛線 + 空心箭頭`，最後加上 `:` + class 名稱表回傳型別

Conclusion



- 請將 homework 6 從使用端經過 `MyBackupService` 到呼叫 `SimpleTask` 的過程以 sequence diagram 表示