

# 套件與存取修飾子

TYIC 桃高資訊社

# 套件

套件(**package**)就是個資料夾  
用來放置、區分不同用途的檔案

套件名稱通常使用全小寫

表示套件時，不同層級(**level**)的套件名稱使用 "." 連接

通常來說會先在原始碼(**source**，簡稱 **src**)根目錄(**root**)下  
以作者和功能命名新增幾層初始套件，避免與其他套件衝突

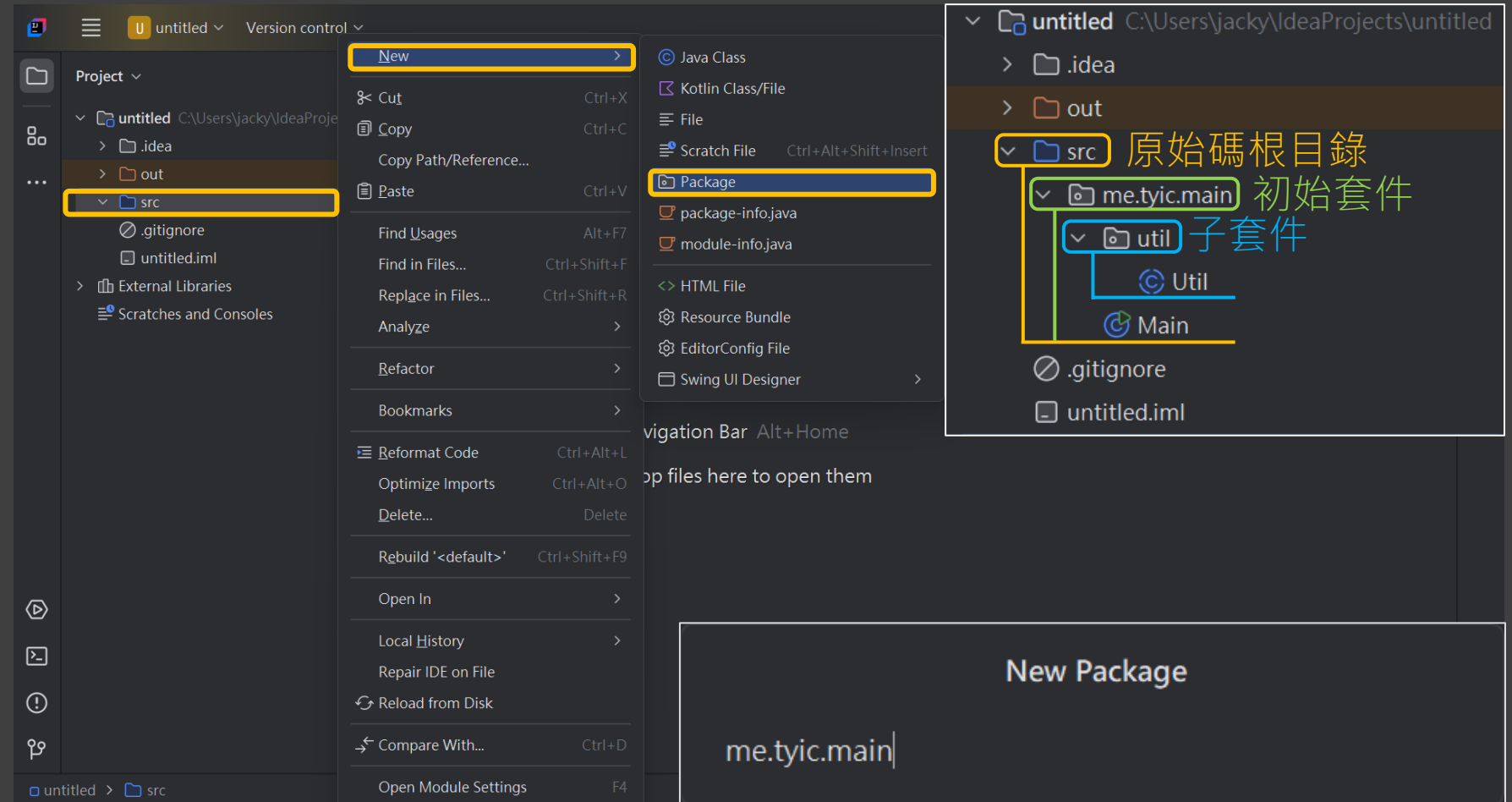
例如：**me.tyic.main**、**com.example.hello**

初始套件下的套件名稱依功能命名，如：**me.tyic.main.util**

且檔案皆放在初始套件及其子套件之下

# 套件

要新增 **package**  
只需要在資料夾上  
右鍵 -> New  
-> Package  
填入類別名稱  
按下 **Enter**  
即可創建



# 套件

若有使用**套件**，需要在每個檔案最上方加上一行標示：

**package** 檔案所屬**套件**；

java

```
package me.tyic.main;

import me.tyic.main.util.Util;

import java.util.Scanner;

public class Main {
    static void printIsPrime(int number) {
        if (Util.isPrime(number)) {
            System.out.printf("%d is prime%n", number);
            return;
        }
        System.out.printf("%d is not prime%n", number);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int base = scanner.nextInt();
        int power = scanner.nextInt();
        printIsPrime(base);
        printIsPrime(power);
        System.out.printf("%d ^ %d = %d", base, power, Util.pow(base, power));
    }
}
```

```
2 4
2 is prime
4 is not prime
2 ^ 4 = 16      console
```

java

```
package me.tyic.main.util;

public abstract class Util {
    public static boolean isPrime(int number) {
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) return false;
        }
        return true;
    }

    public static int pow(int base, int power) {
        int result = 1;
        if (power >= 0) {
            for (int i = 0; i < power; i++) {
                result *= base;
            }
            return result;
        }
        for (int i = -power; i > 0; i--) {
            result *= base;
        }
        return 1 / result;
    }
}
```

java

# 載入

若要使用其他套件的類別或介面，須使用以下格式：

套件. 套件中的類別或介面

java

若想省略前方的套件，就必須先載入(import)

import 套件. 套件中的類別或介面;

java

也可以載入靜態欄位

import static 套件. 套件中的類別或介面. 靜態欄位;

java

若想要載入套件中的所有類別或介面，可以使用以下格式：

import 套件.\*;

java

若是在同一套件下，則不需要載入即可直接使用

編譯器會自動載入 `java.lang.*`，所以該套件內的類別和介面

皆可直接使用，如：`System`、`String`、`Object`、包裝類別等

# 存取修飾子

存取修飾子有四種：

| 存取修飾子   | 類別內部 | 同個套件 | 子類別內 | 任何地方 |
|---|------|------|------|------|
| public  | ○    | ○    | ○    | ○    |
| protected                                       | ○    | ○    | ○    | ×    |
| (default)<br>(no modifier)<br>(package-private) | ○    | ○    | ×    | ×    |
| private   | ○    | ×    | ×    | ×    |

# 存取修飾子

```
package me.tyic.main;

import me.tyic.main.animals.Cat;
import me.tyic.main.animals.Dog;

public class Main {
    public static void main(String[] args) {
        Animal animal1 = new Cat("小貓", 2);
        Animal.printInfo(animal1);
        Animal animal2 = new Dog("小狗", 5);
        animal2.printInfo();
        Animal.makeSound(animal1);
        Animal.makeSound(animal2);
    }
}
```

java

名稱：小貓 年齡：2  
喵！  
名稱：小狗 年齡：5  
喵！  
汪！

output

```
package me.tyic.main;

import me.tyic.main.animals.Cat;
import me.tyic.main.animals.Dog;

public abstract class Animal {
    String name;
    int age;

    protected Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void printInfo() {
        System.out.printf("名稱：%s 年齡：%d %n", name, age);
    }

    public static void printInfo(Animal animal) {
        animal.printInfo();
        makeSound(animal);
    }

    public static void makeSound(Animal animal) {
        if (animal instanceof Cat cat) {
            cat.meow();
        } else if (animal instanceof Dog dog) {
            dog.bark();
        }
    }
}
```

java

```
package me.tyic.main.animals;

import me.tyic.main.Animal;

public class Cat extends Animal {
    public Cat(String name, int age) {
        super(name, age);
    }

    public void meow() {
        System.out.println("喵！");
    }
}
```

java

```
package me.tyic.main.animals;

import me.tyic.main.Animal;

public class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age);
    }

    public void bark() {
        System.out.println("汪！");
    }
}
```

java

# 存取修飾子

任何東西的存取權限  
在繼承或實作中  
只能擴大而不能縮小  
否則會編譯失敗

```
// 編譯失敗
class Person {
    ...

    protected void printInfo() {
        ...
    }
}

class Worker extends Person {
    ...

    @Override
    void printInfo() {
        ...
    }
}
```

```
// 編譯成功
class Person {
    ...

    protected void printInfo() {
        ...
    }
}

class Worker extends Person {
    ...

    @Override
    public void printInfo() {
        ...
    }
}
```