**物件導向(Object-oriented programming)**

[**https://hackmd.io/@Aquamay/H1nxBOLcO/https%3A%2F%2Fhackmd.io%2F%40Aquamay%2FrkAMTLIjd**](https://hackmd.io/@Aquamay/H1nxBOLcO/https%3A%2F%2Fhackmd.io%2F%40Aquamay%2FrkAMTLIjd)

物件導向(Object-oriented programming,OOP)是程式開發的抽象方針，將物件作為程式的基本單元，將程式和資料封裝其中，以提高軟體的重用性、靈活性和擴充性。

## **類別(class)**

建立物件的藍圖，描述所建立的物件共同的屬性和方法。

可能包括：

* 屬性：為物件的資訊。
* 方法：能夠提供的服務功能。
* 建構子：程式執行時第一個去執行的方法，常用於設定程式的初始狀態。

## **物件(object)**

是一個記憶體位址，其中擁有值，這個位址可能有標識符指向此處。可以是一個變數，一個資料結構，或是一個函式。**物件是實體的，由定義好的類別去建立一個物件**。**物件是某一個類別的實例。**

可能包括：

* 屬性：需要記憶的資訊
* 方法：能夠提供的服務

舉個例子來說的話，可以把**學生** Student 當作一個類別，學生具有學號、姓名、科系這些是類別的屬性，有一個方法 getStudentInfo() 回傳學生的資訊。

class Student {

/\* 屬性 \*/

public int studentID;

public String name;

public String major;

/\* 建構子 \*/

public Student(int id, String name, String major) {

this.studentID = id;

this.name = name;

this.major = major;

}

/\* 方法 \*/

public String getStudentInfo() {

return "學號:"+studentID+",姓名:"+name+",主修:"+major;

}

}

有了學生這個類別之後，可以產出很多學生資料，每個學生都是物件，每一個物件是獨立的個體，彼此互不影響。

public class Test {

public static void main(String[] args) {

Student s\_one; // 物件宣告

s\_one = new Student(1,"王小明","資訊工程"); // 物件建構

Student s\_two = new Student(2,"陳小","資訊管理");

Student s\_thr = new Student(3,"李小仁","應用英語");

s\_one.getStudentInfo(); // 學號:1,姓名:王小明,主修:資訊工程

}

}

## **物件導向的三大特性**

物件導向的三大特性包括：封裝、繼承、多形。

### **封裝(Encapsulation)**

一個抽象(abstract)的概念，就是將物件內部的內容隱藏起來，只能透過物件本身提供的介面去取得內部屬性或方法，只需要理解外在，不需要理解內部的構造。

比如我們使用微波爐只要會操作就可以，不需要去理解微波爐裡面的構造；再比如剛剛舉的學生類別的例子，我們可以透過 getStudentInfo() 去取得學生資訊，但不需要知道它是如何取得的。

### **繼承(Inheritance)**

繼承者可以繼承被繼承者的屬性和方法，也能新增自己特有的屬性和方法。

假設有一個 人類 的類別，那麼學生、老師、醫生都是人類，都有名字、年齡、性別，所以不需要再把相同的內容重新宣告一遍。

### **多形**

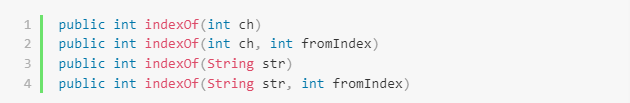
父類別可透過子類別衍伸成多種型態，而父類別為子類別的通用型態，比如 Student 跟 Teacher 繼承 Person 類別

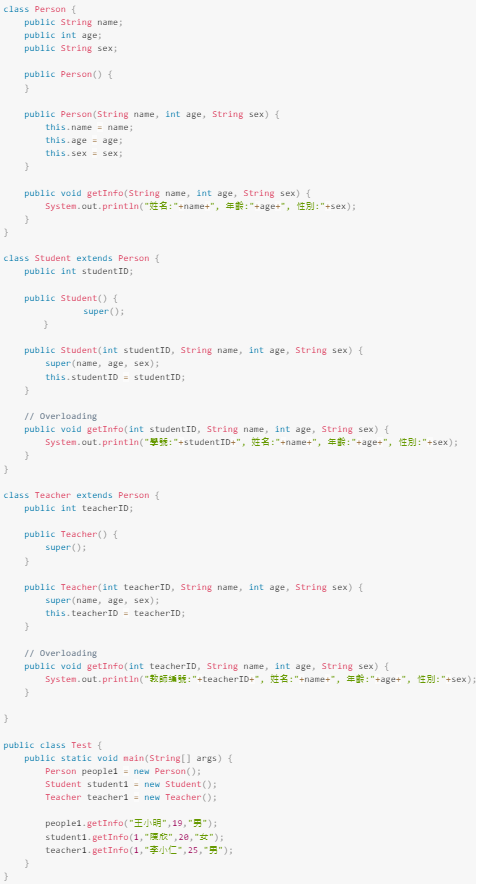
多形還包括 Overloading 和 Overriding

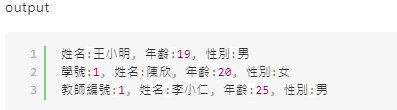
#### **多載(Overloading)**

多個相同名稱的方法，但是傳入不同的參數就會執行不

同的敘述。比如：

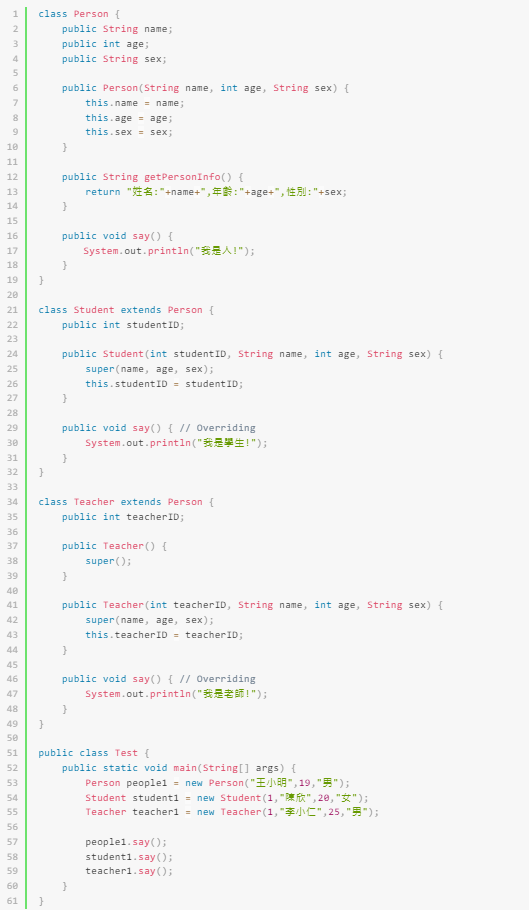


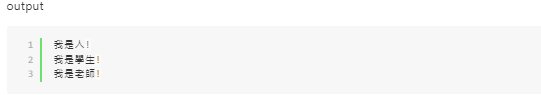




**覆寫(Overriding)**

#### 子類別可以覆寫父類別的方法內容，使其擁有不同於父類別的行為。





class Person {

public String name;

public int age;

public String sex;

public Person(String name, int age, String sex) {

this.name = name;

this.age = age;

this.sex = sex;

}

public String getPersonInfo() {

return "姓名:"+name+",年齡:"+age+",性別:"+sex;

}

}

class Student extends Person {

// 繼承了 Person的屬性和方法

public int studentID;

public Student() {

super();

}

// 定義建構方法

public Student(int studentID, String name, int age, String sex) {

super(name, age, sex);

this.studentID = studentID;

}

}

class Teacher extends Person {

// 繼承了 Person的屬性和方法

public int teacherID;

public Teacher() {

super();

}

// 定義建構方法

public Teacher(int teacherID, String name, int age, String sex) {

super(name, age, sex);

this.teacherID = teacherID;

}

public void setTeacherID(int teacherID) {

this.teacherID = teacherID;

}

}

public class Test {

public static void main(String[] args) {

Person people1 = new Person("王小明",19,"男");

System.out.println(people1.getPersonInfo());

// 姓名:王小明,年齡:19,性別:男

Teacher teacher1 = new Teacher(1,"王小明",19,"男");

System.out.println(teacher1.teacherID + " " + teacher1.getPersonInfo());

// 1 姓名:王小明,年齡:19,性別:男

teacher1.setTeacherID(3);

System.out.println(teacher1.teacherID + " " + teacher1.getPersonInfo());

// 3 姓名:王小明,年齡:19,性別:男

Teacher teacher2 = new Teacher();

System.out.println(teacher2.teacherID + " " + teacher2.getPersonInfo());

// 0 姓名:null,年齡:0,性別:null

teacher2.setTeacherID(4);

System.out.println(teacher2.teacherID + " " + teacher2.getPersonInfo());

// 4 姓名:null,年齡:0,性別:null

}

}

### **多形**

父類別可透過子類別衍伸成多種型態，而父類別為子類別的通用型態，比如 Student 跟 Teacher 繼承 Person 類別

多形還包括 Overloading 和 Overriding

#### **多載(Overloading)**

有多個相同名稱的方法，但是傳入不同的參數就會執行不同的敘述。比如：

public int indexOf(int ch)

public int indexOf(int ch, int fromIndex)

public int indexOf(String str)

public int indexOf(String str, int fromIndex)

以前面的例子接續演示 Overloading

class Person {

public String name;

public int age;

public String sex;

public Person() {

}

public Person(String name, int age, String sex) {

this.name = name;

this.age = age;

this.sex = sex;

}

public void getInfo(String name, int age, String sex) {

System.out.println("姓名:"+name+", 年齡:"+age+", 性別:"+sex);

}

}

class Student extends Person {

public int studentID;

public Student() {

super();

}

public Student(int studentID, String name, int age, String sex) {

super(name, age, sex);

this.studentID = studentID;

}

// Overloading

public void getInfo(int studentID, String name, int age, String sex) {

System.out.println("學號:"+studentID+", 姓名:"+name+", 年齡:"+age+", 性別:"+sex);

}

}

class Teacher extends Person {

public int teacherID;

public Teacher() {

super();

}

public Teacher(int teacherID, String name, int age, String sex) {

super(name, age, sex);

this.teacherID = teacherID;

}

// Overloading

public void getInfo(int teacherID, String name, int age, String sex) {

System.out.println("教師編號:"+teacherID+", 姓名:"+name+", 年齡:"+age+", 性別:"+sex);

}

}

public class Test {

public static void main(String[] args) {

Person people1 = new Person();

Student student1 = new Student();

Teacher teacher1 = new Teacher();

people1.getInfo("王小明",19,"男");

student1.getInfo(1,"陳欣",20,"女");

teacher1.getInfo(1,"李小仁",25,"男");

}

}

output

姓名:王小明, 年齡:19, 性別:男

學號:1, 姓名:陳欣, 年齡:20, 性別:女

教師編號:1, 姓名:李小仁, 年齡:25, 性別:男

#### **覆寫(Overriding)**

子類別可以覆寫父類別的方法內容，使其擁有不同於父類別的行為。

class Person {

public String name;

public int age;

public String sex;

public Person(String name, int age, String sex) {

this.name = name;

this.age = age;

this.sex = sex;

}

public String getPersonInfo() {

return "姓名:"+name+",年齡:"+age+",性別:"+sex;

}

public void say() {

System.out.println("我是人!");

}

}

class Student extends Person {

public int studentID;

public Student(int studentID, String name, int age, String sex) {

super(name, age, sex);

this.studentID = studentID;

}

public void say() { // Overriding

System.out.println("我是學生!");

}

}

class Teacher extends Person {

public int teacherID;

public Teacher() {

super();

}

public Teacher(int teacherID, String name, int age, String sex) {

super(name, age, sex);

this.teacherID = teacherID;

}

public void say() { // Overriding

System.out.println("我是老師!");

}

}

public class Test {

public static void main(String[] args) {

Person people1 = new Person("王小明",19,"男");

Student student1 = new Student(1,"陳欣",20,"女");

Teacher teacher1 = new Teacher(1,"李小仁",25,"男");

people1.say();

student1.say();

teacher1.say();

}

}

output

我是人!

我是學生!

我是老師!