OO的出現主要是要解決程式的易讀性、可維護性跟彈性等問題

這些多半不是再開發階段能立刻享受到的好處，但不要忘了80/20法則

80%的effort都是在維護期間出現的，這時候OO的優點就派上用場了。

Person p;

p.height = 180;

p.weight = 80;

p.walk();

至於為什麼可以有Person p這種宣告方式，其實也是很簡單的概念，

你把Person想成type、p想成變數，那上面那一個跟int x，不就是同等的道理。

1.多型（Polymorphism）

public void execute(Class1 c1) {

c1.doSomething();

}

public void execute(Class2 c2) {

c2.doSomething();

}

------------------------------------------------改成--------------------------------------

public void execute(ParentClass c) {

c.doSomething();

}

2 抽象類別（Abstract class）

僅宣告方法名稱而不實作當中的邏輯 「抽象類別僅能夠被繼承

如果一個方法中包括了抽象方法，則該類別稱之為「抽象類別」（Abstract class）

public abstract class AbstractCircle {

protected double radius;

public double getRadius() { return radius; }

public void setRedius(int radius) { this.radius = radius; }

public abstract void render();

}

public class ConcreteCircle extends AbstractCircle {

public ConcreteCircle() {}

public ConcreteCircle(double radius) {this.radius = radius;}

public void render() {

System.out.printf("畫一個半徑 %f 的實心圓\n", getRadius());

}

}

public class HollowCircle extends AbstractCircle {

public HollowCircle() {}

public HollowCircle(double radius) { this.radius = radius;}

public void render() {

System.out.printf("畫一個半徑 %f 的空心圓\n", getRadius());

}

}

3.介面（Interface）

介面有點像是完全沒有任何方法被實作的抽象類別，一個物件上可以實作多個介面

public interface IRequest {

public void execute();

}

public class HelloRequest implements IRequest {

private String name;

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

public void execute() {

System.out.printf("哈囉 %s！%n", name);

}

}

public class WelcomeRequest implements IRequest {

private String place;

public WelcomeRequest(String place) {this.place = place;}

public void execute() {

System.out.printf("歡迎來到 %s！%n", place);

}

}

雖然 HelloRequest 與 WelcomeRequest 是兩種不同的類型（類別）

但它們都實現了 IRequest，所以只要知道 IRequest 定義了什麼方法，

就可以操作 HelloRequest 與 WelcomeRequest 的實例

public class RequestDemo {

public static void main(String[] args) {

for(int i = 0; i < 10; i++) {

int n = (int) (Math.random() \* 10) % 2; // 隨機產生

switch (n) {

case 0:

doRequest(new HelloRequest("良葛格"));

break;

case 1:

doRequest(new WelcomeRequest("Wiki 網站"));

}

}

}

public static void doRequest(IRequest request) {

request.execute();

}

}