**國立空中大學 管理與資訊學系**

**109 下「Java 程式語言與系統開發」第 2 次作業（109142955潘揚斌）**

1. **請說明 Java 的類別與物件之間的關係？(20 分)**

類別（Class）是用來將物件（Object）分類的，屬於同一類別的物件具有很多相同的特征。

1. **何謂「重訂(Override」？何謂「重載(Overloading」？並 舉例說明之。(20 分)**
2. 子類別可以變更繼承自父類別值方法的功能，稱作「重訂(Override」。
3. 同樣的類別中含有多個名稱相同的方法，卻具有不同的功能，系統可從這些方法的呼叫參數來辨別其差異，稱作「重載(Overloading」。

//重載的方式

class overloadCar extends Car{

String LicensePlate;

String Dealer;

void accelerate() {

System.out.println("Just accelearate!\n");

}

Void accelerate(int a){

System.out.println("Accelerate to 100 mph in "+ a + " seconds!\n");

}

}

//子類別變更繼承自父類別方法的功能

class overRideCar extends Car{

String LicensePlate;

String Dealer;

void EstimatePrice() {

if(year>1985) {

System.out.println("This car costs more than 550000 dollars.\n");

}

else {

System.out.println("This car costs less then 550000 dollars.\n");

}

}

}

**3. 物件參考(object reference)的觀念與指標(pointer)的觀念 有何差異？(20 分)**

(1)物件參考：參考物件的位置，也可以當成是物件的名稱，當系統要用到物件的內容時，必須以物件參考變數的值當做物件的位置。

(2)指標：Java的指標類似C/C++，主要功能就是保存變數的位置，以供程式直接對記憶體位置進行讀寫。因為直接操作記憶體，所以使用指標最大的優點就是效能。

**4. 何謂執行緒(thread)？使用執行緒有何優點？執行緒大致 上可分為哪兩大類？(20 分)**

（1）電腦系統常利用多工的方式來提升效率，而程序(process)和執行緒(thread)則是「多工」的基本觀念。程序是執行中的程式，程式(program)儲存在媒體上，執行時才會載入電腦裡，執行緒可以看成是輕量級的執行程式，通常都會比較簡短。

（2）從多工的來看，執行緒的效能比程序來的更好，因為簡短的執行緒在安排執行時彈性比較大，除此之外，執行緒之間共享位置空間、執行緒執行時的切換較為省事。

（3）執行緒分為以下兩大類：

**常駐執行緒（deamon thread）**

屬於系統產生的執行緒，常駐行在應用程式執行的過程中一直都存在，等所有的使用者執行緒都結束後，常駐執行緒才會停止執行。

**使用者執行緒（user thread）**

執行Java應用程式的時候，系統會產生一個使用者執行緒，負責執行應用城市中的main()方法。

**5. 請寫出一個計算 2 個正整數的最大公因數的程式。(20 分，請將程式碼與執行結果印出來。**

**（下一頁）**

package flowchart;

import java.util.ArrayList;

public class FirstProgram {

public static void main(String[] args) {

int [] num = {16,24};

ArrayList<Integer> arr1 = new ArrayList<Integer>();

for (int i = 0; i < num.length; i++) {

arr1.add(num[i]);

}

int jg = *gcdd*(arr1);

System.***out***.println(jg);

}

static int gcd(int m,int n)

{ if(n == 0){

return m;

}

int r = m%n;

return *gcd*(n,r);

}

static int gcdd(ArrayList<Integer> number) {

while(number.size()>1) {

int temp = *gcd*(number.get(0), number.get(1));

number.remove(0);

number.remove(0);

number.add(0,temp);

}

return number.get(0);

}

}

8