* 作業命名範例: HWK4\_4983437897\_林小明\_01.java

1. [物件模型: 物件陣列]: 依照以下的說明撰寫程式，並產生規定的輸出。

[程式片段]

import java.awt.Point ; // 內建的class Point有覆寫物件模型的各種函數

import java.util.\* ;

class Polygon implements Comparable { // 多邊形

Point[] points ; // 記錄多邊形的各頂點

Polygon() { }

Polygon(Point[] pa) { // 此處暫且使用shallow copy

points = pa ;

}

// -- copy constructor ---

Polygon(Polygon pg2) {

// DIY, 使用deep copy

}

public boolean equals(Object obj) {

// DIY, 依序比較多邊形各頂點是否相同

return true ;

}

public String toString() {

// 格式請參考程式輸出

return "Polygon" ;

}

public double boundary() {

// [功能]: 計算多邊形週長，供compareTo()呼叫

return 0.0 ;

}

public int compareTo(Object obj) {

// DIY: 以多邊形周長的大小做為比較的依據

// 若本多邊形周長大於obj的周長則回傳1，相等回傳0，否則回傳-1

return 0 ;

}

public int hashCode() {

// 初始值17, 之後逐一累加Point之hashCode()，累加前先乘上37

return 0 ;

}

}

public class HWK4\_XX\_YY\_01 {

public static void main(String[] args) {

Point[] ps1 = {new Point(0,0), new Point(4,0), new Point(6,2),

new Point(4,6), new Point(1,6)};

Point[] ps2 = { new Point(4,0), new Point(6,2), new Point(4,6),

new Point(1,6), new Point(0,0)};

Point[] ps3 = { new Point(-3,0), new Point(4,0), new Point(2,5)};

System.out.println("--- test constructor,toString(),boundary(),hashCode() ---") ;

Polygon[] pgs = {new Polygon(ps1), new Polygon(ps2),

new Polygon(ps3), null } ;

pgs[3] = new Polygon(pgs[2]) ; // test copy constructor

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

System.out.print("pgs["+i+"]="+pgs[i]);

System.out.printf(" Boundary=%.3f",pgs[i].boundary()) ;

System.out.printf(" hashCode=%d\n",pgs[i].hashCode()) ;

}

System.out.println("---- test equals()--- ") ;

Polygon key = new Polygon(new Point[]{

new Point(6,2), new Point(4,6),

new Point(1,6), new Point(0,0), new Point(4,0)}) ;

System.out.println("key="+key) ;

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

if (pgs[i].equals(key))

System.out.println("found in pgs[], index="+i) ;

}

System.out.println("---- test Comparable ---") ;

Arrays.sort(pgs) ;

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

System.out.println("pgs["+i+"]="+pgs[i]);

}

}

/\*[程式輸出]

--- test constructor,toString(),boundary(),hashCode() ---

pgs[0]={(0,0),(4,0),(6,2),(4,6),(1,6)} Boundary=20.383 hashCode=1209532433

pgs[1]={(4,0),(6,2),(4,6),(1,6),(0,0)} Boundary=20.383 hashCode=1209532433

pgs[2]={(-3,0),(4,0),(2,5)} Boundary=19.456 hashCode=-585891823

pgs[3]={(-3,0),(4,0),(2,5)} Boundary=19.456 hashCode=-585891823

---- test equals()---

key={(6,2),(4,6),(1,6),(0,0),(4,0)}

found in pgs[], index=0

found in pgs[], index=1

---- test Comparable ---

pgs[0]={(-3,0),(4,0),(2,5)}

pgs[1]={(-3,0),(4,0),(2,5)}

pgs[2]={(0,0),(4,0),(6,2),(4,6),(1,6)}

pgs[3]={(4,0),(6,2),(4,6),(1,6),(0,0)}

\*/

1. [物件模型: 繼承與equals()]: 根據程式中的說明與[程式輸出]，分別為class A, B加上合適的函數，以產生規定的輸出。

[程式片段]

class A {

private int[] a = new int[5];

A(){}

A(int[] x) {

for (int i = 0 ; i<a.length; i++) { a[i] = x[i] ; }

}

int getLen() { return a.length; }

// --- DIY: 自行加上equals()與toString()

// [注意]: 本類別中不可加入getXXX(), setXXX()等accessor functions

}

class B extends A {

private int[] b = new int[3] ;

B() {}

B(int[] ab) {

// DIY: ab[]的前5個元素初始化a[], 其餘的初始化b[]

}

// -- 自行加上equals()與toString()

}

public class HWK4\_XX\_YY\_02 {

public static void main(String[] args) {

int[] ab = {1,2,3,4,5,6,7,8} ;

B b1 = new B(ab) ; // 將{1,2,3,4,5},{6,7,8}分別填入a[],b[]

B b2 = new B(new int[]{1,2,3,4,5,6,7,9}) ;

System.out.println("b1="+b1) ;

System.out.println("b2="+b2) ;

A a = new A(ab) ;

System.out.println("a="+a) ;

if (b1.equals(b1)) System.out.println("b1 equals b1") ;

if (!b1.equals(b2)) System.out.println("b1 not equals b2") ;

if (!b1.equals(a)) System.out.println("b1 not equals to a") ;

if (a.equals(b2)) System.out.println("a equals to b2") ;

}

}

/\* [程式輸出]

b1=1 2 3 4 5 6 7 8

b2=1 2 3 4 5 6 7 9

a=1 2 3 4 5

b1 equals b1

b1 not equals b2

b1 not equals to a

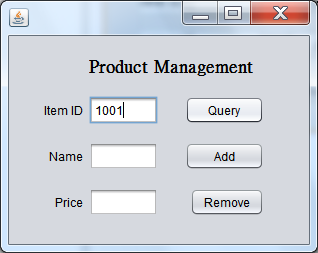
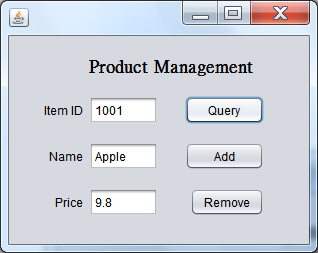
a equals to b2

\*/

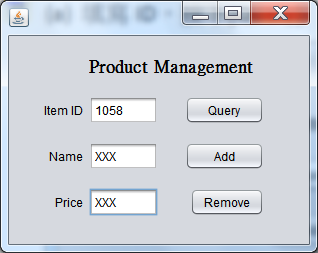
1. [HashMap]: 將以下資料表存入一HashMap，其中的ID為鍵值，再建立如下畫面，供使用者查詢、加入與刪除資料。

|  |  |  |
| --- | --- | --- |
| ID | Name | Price |
| 1001 | Apple | 9.8 |
| 1002 | Banana | 4.5 |
| 1004 | Peach | 12.3 |
| 1006 | Water Melon | 8.5 |

1. [Query]鍵
2. 填寫ID，按下[Query] (b) 由HashMap中找出資料

(c) 查無此ID



1. [Add]: 使用者自行填寫ID, Name, Price後，按下[Add]，若此ID不存在，則將其加入HashMap中
2. [Remove]: 使用者可在查詢(Query)成功後，按下[Remove]，將可該筆記錄由HashMap中刪除。

[程式片段]

// -- 畫面可用Netbeans工具製作，不一定要自行撰寫 ---

public class HWK4\_XX\_YY\_03 extends JFrame implements ActionListener {

// 請將以下三個陣列加入程式中，並利用HashMap<Prodcut>….來完成

static int[] id = {1001, 1002, 1004, 1006} ;

static String[] name = {"Apple", "Banana", "Peach", "Water Melon"} ;

static double[] price = {9.8, 4.5, 12.3, 8.5} ;

// …

}

<Product.java>: 自行加入project中

import java.util.\* ;

class Product {

int id ; String name ; double price ;

Product () { }

Product(int i, String n, double p) {

id=i; name=n; price=p ;

}

// 如有需要可自行加入其他函數

}