* 作業命名範例: HWK3\_4983437897\_林小明\_01.java

1. [Equals]: P.29, EX1
2. [Equals]: P29, EX2

[程式片段]

import java.util.\*;

class Account {

String eMail; // 電子郵件

int passwd; // 密碼

String addr; // 住址

Account(String em, int p, String ad) {

eMail = em; passwd = p; addr = ad;

}

Account(Account acc) { // copy constructor

eMail = acc.eMail; passwd = acc.passwd; addr = acc.addr;

}

// -- 只用eMail, passwd欄位做為equals() 比較的依據

public boolean equals(Object o) {

// DIY here

return super.equals(o) ;

}

public String toString() {

return "(EMAIL:" + eMail + ", PASSWD" + passwd + ", ADDR:" + addr + ")";

}

}

public class HWK3\_XX\_YY\_02 {

// -- accList串列儲存合法帳號(cache)，供登入時比對之用 --

static ArrayList<Account> accList = new ArrayList<Account>();

public static void main(String[] args) {

// --- 建立20個測試帳號 ----

buildAccountList(20);

//-- 場景: 配合使用者歷次登入，查詢list中的資料，決定是否為合法帳戶 --

Account[] queryData = {

new Account("acc\_0@tku.edu.tw", 1000, "addr"), //存在

new Account("acc\_4@tku.edu.tw", 1004, "addr"), //存在

new Account("acc\_8@tku.edu.tw", 1008, "addr"), //存在

new Account("acc\_8@tku.edu.tw", 1055, "addr"), //不存在

new Account("acc8@tku.edu.tw", 1008, "addr"), //不存在

};

//-- 在accList中搜尋比對，看看登入帳號是否合法?

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

int index = accList.indexOf(queryData[i]);

if (index != -1) { // 在list中有找到queryData[i]

System.out.println(accList.get(index));

}

}

}

public static void buildAccountList(int no) {

// 建立no個測試帳號

for (int i = 0; i < no; i++) {

accList.add(new Account("acc\_" + i

+ "@tku.edu.tw", 1000 + i, "Passwed"));

}

}

}

1. [大數的質因數分解]: 完成以下程式，並產生規定輸出。

[程式片段]

import java.util.\* ;

public class HWK3\_XX\_YY\_03 {

// 使用質因數分解表示法，計算並表示大整數。

public static void main(String[] args) {

for (int i = 10; i<=100; i+=10) {

System.out.println(i+"!="+factorial(i)) ; // 10!, 20!, ..., 100!

}

// primeFactoring(Integer.MAX\_VALUE-1) ;

}

public static int factorial(int n) { // 計算 n!, n太大無法計算

int fac = 1 ;

for (int i=2; i<=n; i++) {

fac \*= i ;

}

return fac ;

}

public static void primeFactoring(int x) {

x = Math.abs(x) ;

int divisor = 2 ; // 除數

System.out.print(x+"=") ;

while (x>=2 && x>=divisor) {

int cnt = 0 ; // 記錄次方數

while (x%divisor==0) {

x /= divisor ;

cnt++ ;

}

if (cnt!=0) { // 此處的divisor湊巧都會是質數，why?

System.out.print(divisor+"^"+cnt+" ") ;

}

divisor++ ;

}

System.out.println() ;

}

}

/\* [執行結果]

10!=(2^8)(3^4)(5^2)(7^1)

20!=(2^18)(3^8)(5^4)(7^2)(11^1)(13^1)(17^1)(19^1)

30!=(2^26)(3^14)(5^7)(7^4)(11^2)(13^2)(17^1)(19^1)(23^1)(29^1)

40!=(2^38)(3^18)(5^9)(7^5)(11^3)(13^3)(17^2)(19^2)(23^1)(29^1)(31^1)(37^1)

50!=(2^47)(3^22)(5^12)(7^8)(11^4)(13^3)(17^2)(19^2)(23^2)(29^1)(31^1)(37^1)(41^1)(43^1)(47^1)

60!=(2^56)(3^28)(5^14)(7^9)(11^5)(13^4)(17^3)(19^3)(23^2)(29^2)(31^1)(37^1)(41^1)(43^1)(47^1)(53^1)(59^1)

70!=(2^67)(3^32)(5^16)(7^11)(11^6)(13^5)(17^4)(19^3)(23^3)(29^2)(31^2)(37^1)(41^1)(43^1)(47^1)(53^1)(59^1)(61^1)(67^1)

80!=(2^78)(3^36)(5^19)(7^12)(11^7)(13^6)(17^4)(19^4)(23^3)(29^2)(31^2)(37^2)(41^1)(43^1)(47^1)(53^1)(59^1)(61^1)(67^1)(71^1)(73^1)(79^1)

90!=(2^86)(3^44)(5^21)(7^13)(11^8)(13^6)(17^5)(19^4)(23^3)(29^3)(31^2)(37^2)(41^2)(43^2)(47^1)(53^1)(59^1)(61^1)(67^1)(71^1)(73^1)(79^1)(83^1)(89^1)

100!=(2^97)(3^48)(5^24)(7^16)(11^9)(13^7)(17^5)(19^5)(23^4)(29^3)(31^3)(37^2)(41^2)(43^2)(47^2)(53^1)(59^1)(61^1)(67^1)(71^1)(73^1)(79^1)(83^1)(89^1)(97^1)

\*/