Java学习

Java基礎知識

1. 基本语法：
2. 编写程序应注意：大小写敏感；类名的单词首字母大写。如MyFirst；

方法名应该以小写字母开头，若有其他单词则其他单词首字母大写。；

源文件名后缀：.java；主方法入口:所有程序由public static void main（String[] args）方法开始执行

Idea常用快捷键：

Main方法快捷键：psvm

控制台原样输出：sout+回车

删除一行：ctrl+v

复制一行：ctrl+d

Ctrl+alt+m 選中代碼部分可以自動補全，封裝成方法

Ctrl+alt+l 可以自動調整代碼格式

對象內存圖（底層邏輯）-面向對象07、

添加構造方法的快捷鍵：

//快捷鍵  
//alt+insert  
//alt + Fn +insert

未寫：抽獎的兩種實現方式，雙色球彩票系統，二維數組

92-95視頻面向對象封裝習題未寫

100-110未寫

二．運算符、表達式、流程控制語句、數組

鍵盤輸入：1. Scanner sc=new Scanner(System.in);

2.變量類型 變量名= sc.nextInt(); #使用變量接收輸入的數據

1.運算符：

（1）.賦值運算符

|  |  |  |
| --- | --- | --- |
| 符號 | 作用 | 說明 |
| = | 賦值 | int a=10，將10 賦值給變量啊 |
| -= | 減後賦值 | a-=b 將a—b的值給a |
| += | 加後賦值 | a+=b 將a+b的值給a |
| \*= | 乘後賦值 | a\*=b 將a\*b的值給a |
| /= | 除後賦值 | a/=b 將a/b的值給a |
| %= | 取餘後賦值 | a%=b 將a/b的餘數給a |

注意：擴展的賦值運算符隱含了強制類型轉換

（2）（關係運算符/比較運算符）的分類

|  |  |
| --- | --- |
| 符號 | 說明 |
| == | a==b，判斷a和b的值是否相等，成立為true，不成立為false。 |
| != | a!=b,判斷a和b的值是否相等，成立為true，不成立為false |
| > | a>b,判斷a是否大於b，成立為true，不成立為false |
| >= | a>=b,判斷a是否大於等於b，成立為true，不成立為false |
| < | a<b,判斷a是否小於b，成立為true，不成立為false |
| <= | a<=b,判斷a是否小於等於b，成立為true，不成立為false |

注意：關係運算符的結果都是boolean類型，要麼是true，要麼是false，

千萬不要把”==”誤寫成”=”

（3）.4種邏輯運算符

|  |  |  |
| --- | --- | --- |
| 符號 | 作用 | 說明 |
| & | 邏輯與（且） | 並且，兩邊都為真，結果才為真 |
| | | 邏輯或 | 或者，兩邊都為假，結果才為假 |
| ^ | 邏輯異或 | 相同為false，不同為true |
| ! | 邏輯非 | 取反 |

短路邏輯運算符

|  |  |  |
| --- | --- | --- |
| 符號 | 作用 | 說明 |
| && | 短路與 | 結果和&相同，但是具有短路效果 |
| || | 短路或 | 結果和|相同，但是具有短路效果 |

注意事項：

&|，無論左邊true false，右邊都要執行。

&& ||，如果左邊能確定整個表達式的結果，右邊不執行。

&&：左邊為false，右邊不管是真是假，整個表達式的結果一定是false，並且右邊表達式不會執行

||：左邊為true，右邊不管是真是假，整個表達式的結果一定是true，並且右邊表達式不會執行

後面兩種出現右邊不執行的情況，提高了運行效率

（4）三元運算符

作用：可以進行判斷，根據判斷的結果得到不同的內容

格式：關係表達式？表達式1：表達式2;

規則：首先確定關係表達式的結果

若結果為true，執行表達式1

若結果為false。執行表達式2

例子：a>b?a:b 若前面表達式結果為真，則選擇a，否則選擇b，一般需要一個變量進行接收

1. 流程控制語句
2. 順序結構
3. 分支結構

If 分支

//1.if分支-单分支

语法结构：  
if(布尔表达式){  
 语句块  
}

注意點：a.大括號的開頭可以另起一行書寫，但是建議寫在第一行的末尾

b.在語句體中，如果只有一行代碼，大括號可以不寫

c．如果對一個布爾類型的變量進行判斷，不要用==號，直接把變量寫在小括號中  
2.if双分支  
 语法结构：  
if(布尔表达式){  
 语句块1  
}else{  
 语句块2  
}

執行流程：1.先執行布爾表達式

2.若表達式為true則執行語句塊1，若表達式為false,則執行語句塊2

3.if多 分支  
 语法结构：  
if(布尔表达式1){  
 语句块1  
}else if（布尔表达式2）{  
 语句块2  
}else if（布尔表达式3）{  
 语句块3  
}....  
else if(布尔表达式n){  
 语句块n  
}else{  
 语句块n+1  
}

3.Switch 選擇

switch語句格式  
 switch（表達式）{  
 case 值1：  
 語句體1；  
 break；  
 case 值2：  
 語句體2;  
 break;  
 case 值3:  
 語句體3;  
 break;  
 ....  
 default:  
 語句體n+1;  
 break;   
 }  
 執行流程:  
 1.首先計算表達式的值  
 2.依次和case後面的值進行比較，如果有對應的值，就會執行相應的語句，在執行的過程中，遇到break就會結束  
 3.如果所有的case後面的值和表達式的值都不匹配，就會執行default裡面的語句體，然後結束整個switch語句。

格式說明：

1.表達式：（將要匹配的值）取值為byte、short、int、char JDK5以後可以是枚舉 JDK7以後可以是String

2.case：後面跟的是要和表達式進行比較的值（被匹配的值）

3.break:表示中斷，結束的意思，用來結束switch語句

4.default：表示所有情況都不匹配的時候，就執行該處的內容，和if語句的else相似

5.case後面的值只能是字面量，不能是變量

6.case給出的值不允許重複

補充：

default的位置和省略

1.位置：default不一定是寫在最下面，可以寫在任意位置，習慣會寫在最下面

2.省略：default的位置可以省略，語法不會有問題，但是不建議省略

case穿透：語句體當中沒有break語句，那麼程序會繼續執行下一個case的語句體，直至遇到break或者右大括號為止

使用場景：如果多個case的語句體重複了，那麼我們考慮利用case穿透去簡化代碼

switch和if語句第三種格式的各種使用場景：

if的第三種格式：一般用於對範圍的判斷

switch：把有限個數據一一列舉出來，讓我們任選其一

1. 循環結構

1.for循環  
格式：for(初始化語句;條件判斷語句;條件控制語句){  
 循環語句體  
}  
執行流程：  
（1）.執行初始化語句  
（2）.執行條件判斷語句，看結果是true還是false  
 如果是true，執行循環語句體  
 如果是false，循環結束  
（3）.執行條件控制語句  
（4）.回到2繼續執行條件判斷語句

2.while循環

格式：  
 初始化語句;  
 while(條件判斷語句){  
 循環語句體;  
 條件控制語句;  
 }  
 執行流程：  
（1）.執行初始化語句  
（2）.執行條件判斷語句，看結果是true還是false  
 如果是true，執行循環語句體  
 如果是false，循環結束  
（3）.執行條件控制語句  
（4）.回到2繼續執行條件判斷語句  
  
for和while的對比  
相同點：運行規則都是一樣的  
區別：  
for循環中：知道循環次數或者循環的範圍  
while循環：不知道循環的次數和範圍，只知道循環的結束條件

3.do...while循環  
格式：  
初始化語句;  
do{  
 循環體語句;  
 條件控制語句;  
}while (條件判斷語句);

49集至52集練習題，之後有空在寫

生成隨機數：

import java.util.Random; #1.導包

Random rc=new Random(); #創建隨機數

int num= rc.nextInt(隨機數的範圍); #實例化對象

注意：這個範圍是0開始，到給定數-1結束

所以如果定義int num= rc.nextInt(100);，範圍只是0~99，如果要得到1~100，需要整體加一int num= rc.nextInt(100)+1;

1. 數組

//數組：指的是一種容器，可以用來存儲同種數據類型的多個值。

// 注意：存儲數據時，需要結合隱式轉換考慮。例如：int類型的數組容器（還可以存byte，short），double類型的數組容器（還可以byte、short、int、long、float）

//建議：容器的類型，和存儲的數據類型保持一致

/\*數組的定義 ：

格式一：數據類型 [] 數組名

格式二：數據類型 數組名[]

數組的靜態初始化

初始化：就是在內存中，為數據容器開闢空間，並將數據存入容器中的過程

完整格式：數據類型[] 數組名=new 數據類型[]{元素1、元素2、元素3...}

簡化格式：數據類型[]數組名={元素1... }

範例： int []array=new int[]{11,22,33}

數組地址值：表示數組在整個內存中的位置。

數組元素訪問：利用索引對數組中的元素進行訪問

1.獲取數組裡面的元素

格式：數據名[索引]

2.把數據存儲到數組當中

格式：數組名[索引] = 具體元素/變量

數組的遍歷：獲取數組裡的所有元素

數組的動態初始化：初始化時只指定數組長度，由系統為數組分配初始值

格式：數據類型[] 數組名=new 數據類型[數組長度] 如int [] array=new int[3]

數組默認初始化值得規律：

整數類型：默認初始化值0

小數類型：默認初始化值0.0

字符類型：默認初始化值'/u0000'空格

布爾類型：默認初始化值false

引用類型：默認初始化值null

動態初始化和靜態初始化區別：

動態初始化：手動指定數組長度，由系統給出默認初始值。

靜態初始化：手動指定數組元素，系統會根據元素個數，計算出數組的長度

只明確元素個數，不確定具體數值，推薦使用動態初始化

需求中已經明確了要操作的具體數據，直接靜態初始化即可

數組常見操作：求最值（統計小於平均數的個數案例：for循環加if語句）、求和、交換數據（逆序操作：雙指針加臨時變量實現數組逆序）、打亂數據（與隨機索引數據交換數據）

補充：

Java內存分配

棧 ：方法運行時使用的內存，比如main方法進行。進入方法棧中執行

堆 ：存儲對象或數組，new來創建的，都存儲在堆內存

方法區：存儲可以運行的class文件

本地方法棧： JVM在使用操作系統功能的時候使用，和我們開發無關

寄存器：給cpu使用，和我們開發無關

數組的內存圖

1.只要是new出來的一定是在堆裡面開闢了一個小空間

2.如果new了多次，那麼在對林有多個小空間，每個小空間中都有各自的數據

兩個數組指向同一個數組

當兩個時更要注意指向同一個小空間是，其中一個數組對小空間中的值發生了改變，那麼其他數組再次訪問的時候都是修改之後的結果了。

方法：程序中最小的執行單元  
方法的應用場景：重複的代碼、具有獨立功能的代碼可以抽取到方法中  
方法的好處：提高代碼複用性，提高代碼的可維護性。  
  
方法的格式：  
1.定義格式： public static 返回值類型 方法名 （參數）{  
 方法體;  
 return 返回值;  
 }  
1.1最簡單的方法定義和調用  
定義：public static void 方法名（）{  
 方法體  
}  
調用：方法名（）;  
1.2帶參數的方法定義和調用  
定義：public static void 方法名（參數1，參數2，...）{ 參數定義的個數至少為一，可以多個  
 方法體  
}  
調用：方法名（參數1，參數2，...）  
2形參和實參  
形參：全稱形式參數，是指方法定義中的參數  
實參：全稱實際參數，方法調用中的參數。  
注意：方法調用中，形參和實參個數要一一對應，不然會報錯  
方法定義的小技巧？  
1.我要幹什麼？ 方法體  
2.我幹這件事需要什麼？ 參數  
  
為什麼要有返回值？為了獲取方法運行的最終結果  
返回值：方法運行的最終結果  
帶返回值方法的定義和調用  
定義：public static 返回值類型 方法名（參數）{ 參數定義的個數至少為一，可以多個  
 方法體;  
 return 返回值;  
}  
如：public static void 方法名（int a，int b）{  
 int c= a + b;  
 return c;  
}  
調用： 直接調用：方法名（實參）;  
 賦值調用:整數類型 變量名=方法名（實參）;  
 輸出調用：System.out.println(方法名（實參）);  
  
總結：  
1.什麼時候用到方法？  
在調用處要根據方法的結果，去編寫另外一段代碼  
2.有返回值方法的定義格式？  
public static 返回值類型 方法名（參數）{ 參數定義的個數至少為一，可以多個  
 方法體;  
 return 返回值;  
}  
3.有返回值得方法的調用格式  
直接調用，賦值調用，輸出調用  
4.有返回值方法中參數的運行過程  
  
方法小結：實則就是只有一種格式：有多種格式取決於是否要有返回值和參數，不要返回值則返回值類型改成void，去掉return即可  
public static 返回值類型 方法名（參數）{ 參數定義的個數至少為一，可以多個  
 方法體;  
 return 返回值;  
}  
注意事項：  
方法與方法之間是平級關係，不能互相嵌套  
方法的編寫順序和執行順序無關  
方法的返回值類型為void，表示該方法沒有返回值，  
沒有返回值得方法可以省略return語句不寫。  
如果要編寫return，後面不能跟具體的數據  
return語句代表結束方法，下面不能編寫代碼，因為執行不到，屬於無效代碼  
  
return關鍵字  
方法沒有返回值：可以省略不寫，如果書寫，表示方法結束  
方法有返回值：必須要寫。表示結束方法和返回結果。

方法的重載：

//在同一個類中，定義了多個同名的方法，這些同名的方法具有同種的功能。

//每個方法具有不同的參數類型或參數個數，這些同名的方法，構成了重載關係。

// 簡單記：同一個類中，方法名相同，參數不同的方法。與返回值無關。

// 參數不同：個數不同，類型不同，順序不同

例題：用方法遍歷數組，用方法求數組最大值、判斷某個數是否存在數組中（只有當數組中數字全部比較完了，才可以斷定不存儲）

//return break 關鍵字區別  
//return ：其實跟循環沒什麼關係，跟方法有關係，表示：1.方法結束 2.返回結果  
// 如果方法執行到了return，那麼整個方法全部借宿，裡面的循環也會隨之結束了  
//break :其實跟方法沒什麼關係，結束循環或者switch的

補充：（此段可以多看視頻73）

基本數據類型和引用數據類型

基本數據類型：數據值是存儲在自己的空間中。

特點：賦值給其他變量，也是賦的真實的值。

引用數據類型：數據值是存儲在其他空間中，自己空間中存儲的是地址值

特點：賦值給其他變量，賦的是地址值。

引用：使用了其他空間中的數據。

方法傳遞基本數據類型的內存原理：傳遞基本數據類型時，傳遞的是真實的數據，形參的改變，不影響實際參數的值

//return break 關鍵字區別

//return ：其實跟循環沒什麼關係，跟方法有關係，表示：1.方法結束 2.返回結果

// 如果方法執行到了return，那麼整個方法全部結束，裡面的循環也會隨之結束了

//break :其實跟方法沒什麼關係，結束循環或者switch的

//方法的重載：

// 在同一個類中，定義了多個同名的方法，這些同名的方法具有同種的功能。

// 每個方法具有不同的參數類型或參數個數，這些同名的方法，構成了重載關係。

// 簡單記：同一個類中，方法名相同，參數不同的方法。與返回值無關。

// 參數不同：個數不同，類型不同，順序不同

飛機票案例（方法對重複代碼的優化）、隨機生成5位數驗證碼（26個字母隨機生成處理）、複製數組、評委打分、數字加密與解密（多斟酌）、

未寫：抽獎的兩種實現方式，雙色球彩票系統，二維數組

1. 面向對象

面向對象

/\*類和對象

類（設計圖）：是對象共同特征的描述；

對象：是真實存在的具體東西

在Java中，必須先設計類才能獲得對象

3.2.如何得到對象？

public class 類名{

1.成員變量（屬性，名詞）

2.成員方法（行為，動作，動詞）

3.構造器

4.代碼塊

5.內部類

}

定義類的補充注意事項：

用來描述一類事物的類，專業叫做：Javabean類

在Javabean類中，是不寫main方法的

在以前編寫的main方法的類，叫做測試類

我們可以在測試類中創建Javabean類的對象，並調用屬性和方法

3.3定義類有哪些建議，有什麼需要注意的？

類名首字母建議大寫，需要見名知意，駝峰格式，不能用關鍵字，符合標識符規定。

一個Java文件中可以定義多個只有public修飾的類，實際一般一個文件定義一個類

成員變量的格式是什麼樣的？有什麼特點？

成員變量的完整格式： (修飾符) 數據類型 變量名稱 (=初始值),一般無需指定初始值，存在默認值

整數類型默認為0，浮點型，double型為0.0；引用類型為null

面向對象的三大特點：

1. 封裝

//1.什麼是封裝

// 封裝：正確設計對象的屬性和方法。原則：對象代表什麼，就得封裝對應的數據，並提供數據對應的行為。

//2.封裝思想的好處？

// 需要什麼方法去找即可，讓編程變得簡單

/\*private關鍵字

是一個權限修飾符

可以修飾成員（成員變量和成員方法）

被private修飾的成員只能在本類中才能訪問

若想要訪問賦值，得配合setxxx和getxxx一對方法才能使用，方法用public修飾使用

\*

成員變量和局部變量

就近原則（方法中屬性調用，誰離得近，調用誰的屬性）

當方法中，存在局部變量與成員變量相同時，this關鍵字可以在方法裡面使用成員變量的屬性

構造方法：

/\*構造方法的格式

\* public class Student{

\* 修飾符 類名（參數）{

\* 方法體;

\* }

\* }

\* 特點：

\* 1.方法名與類名相同，大小寫也要一致

\* 2.沒有返回值類型，連void都沒有

\* 3.沒有具體的的返回值（不能由return帶回結果數據）

\*

\* 執行時機：

\* 1.創建對象的時候由虛擬機調用，不能手動調用構造方法

\*2.每創建一次對象，就會調用一次構造方法

\*

\* 小結：

\* 1.構造方法的作用？

\* 創造對象的時候，有虛擬機自動調用，給成員變量進行初始化的。

\* 2.構造方法有幾種，各自的作用是什麼？

\* 無參構造方法：初始化對象時，成員變量的數據均採取默認值。

\* 有參構造方法：在初始化對象的時候，同時可以為對象進行賦值

\*

\* 構造方法注意事項

\* （1）構造方法的定義

\* 如果沒有定義構造方法，系統將給出一個默認的無參數構造方法

\* 如果定義了構造方法，系統將不再提供默認的構造方法

\* (2)構造方法的重載

\* 帶參構造方法，和無參構造方法，兩者方法名相同，但是參數不同，這叫做構造方法的重載

\* （3）推薦的使用方式

\* 無論是否使用，都手動書寫無參構造方法，和帶全部參數的構造方法

\* 標準的Javabean類

\* （1）類名需要見名知億

\* （2）成員變量使用private修飾

\* （3）提供至少兩個構造方法

\* a.無參構造方法

\* b.帶全部參數的構造方法

\* （4）成員方法

\* a.提供每一個成員變量對應的setxxx（）/getxxx()

\* b.如果還有其他行為，也需要寫上

API

API：應用程序編程接口

簡單理解：API就是別人已經寫好的東西，我們不需要自己編寫，直接使用即可

Java API：指的就是jdk中提供的各種功能的Java類

已經學過的 ：

Scanner 、Random

//字符串  
/\*字符串學習內容：  
1.String,StringBuilder,StringJonier,StringBuffer,Pattern,Matcher  
可以掌握字符串的一些常見操作了  
2.實際開發中的一些常見案例  
掌握分析問題，解決問題的能力。  
3.字符串相關的底層原理  
掌握原理更好的通過面試，處理開發中的一些複雜問題  
4.字符串的練習題  
鍛煉大家獨立解決問題的能力  
  
String字符串注意點：  
 String是Java定義好的一個類。使用的時候不需要導包。  
 java程序中的所有字符串文字（例如"abd"）都被稱為此類的對象  
 字符串的內容是不會發生改變的，他的對象在創建後不能被更改

創建方式：1.直接賦值。節省內存。

2.（4種方式）

/空參構造，可以獲取一個空白的字符串對象

String s2=new String();

System.out.println("@"+s2+"!");

//傳遞一個字符串，根據傳遞的字符串內容再創建一個新的字符串對象

String s3=new String("abc");

System.out.println(s3);

//傳遞一個字符數組，根據字符數組的內容再創建一個新的字符串對象 String（char []）

//需求：我要修改字符串的內容 abc-》Qbc 修改數組即可

char []chs={'a','b','c'};

String s4=new String(chs);

System.out.println(s4); //abc

//傳遞一個字節數組，根據字節數組的內容再創建一個新的字符串對象 String（byte []）

//應用場景：以後在網絡當中傳輸的數據其實都是字節信息

//我們一般要把字節信息進行轉換，轉成字符串,此時就要用到這個構造了

byte[] bytes={97,98,99,100};

String s5=new String(bytes);

System.out.println(s5);//abcd

//字符串的比較 （直接賦值得到的字符串，和創建對象所得到的字符串比較）

//boolean equals 方法（要比較的字符串） 完全一樣結果才是true，否則是false 格式：字符串1.equals（字符串2）

//boolean equalsIgnoreCase（要比較的字符串） 忽略大小寫的比較

格式如上

/\*集合

1.集合和數組的對比

（1）從長度：數組長度固定，集合長度可變

（2）從存儲類型：數組既可以存基本數據類型，又可以存引用數據類型。

集合可以存引用數據類型，不能直接存基本數據類型，可以使用包裝類進行基本數據類型存儲。

集合的操作：增、刪（兩種）、改、查、遍歷