1.static:靜態的

2.static可以用來修飾：主要修飾類的內部結構

屬性、方法、代碼塊、內部類

3.使用static修飾屬性：靜態變量（或類變量）

3.1 屬性，按是否使用static修飾，又分為：靜態屬性vs 非靜態屬性(實例變量)

實例變量：我們創建了類的多個對象，每個對像都獨立的擁有一套類中的非靜態屬性。當修改其中一個對像中的非靜態屬性時，不會導致其他對像中同樣的屬性值的修改。

靜態變量：我們創建了類的多個對象，多個對象共享同一個靜態變量。當通過某一個對象修改靜態變量時，會導致其他對象調用此靜態變量時，是修改過了的。

3.2 static修飾屬性的其他說明：

3.2.1 靜態變量隨著類的加載而加載。可以通過"類.靜態變量"的方式進行調用

3.2.2 靜態變量的加載要早於對象的創建。

3.2.3 由於類只會加載一次，則靜態變量在內存中也只會存在一份：存在方法區的靜態域中。

3.2.4 類變量 實例變量

類 yes no

對象 yes yes

3.3 靜態屬性舉例：System.out; Math.PI;

4.靜態變量內存解析:



4.使用static修飾方法：靜態方法、類方法

4.1 隨著類的加載而加載，可以通過"類.靜態方法"的方式進行調用

4.2 靜態方法 非靜態方法

類 yes no

對象 yes yes

4.3 靜態方法中，只能調用靜態的方法或屬性(跟生命週期有關)

非靜態方法中，既可以調用非靜態的方法或屬性，也可以調用靜態的方法或屬性

5. static注意點：

5.1 在靜態的方法內，不能使用this關鍵字、super關鍵字

5.2 關於靜態屬性和靜態方法的使用，大家都從生命週期的角度去理解。

6. 開發中，如何確定一個屬性、方法是否要聲明為static的？

6.1 關於屬性

屬性是可以被多個對象所共享的，不會隨著對象的不同而不同的。

類中的常量也常常聲明為static

6.2 關於方法

操作靜態屬性的方法，通常設置為static的

工具類中的方法，習慣上聲明為static的。比如：Math、Arrays、Collections

7.使用舉例：

舉例一：Arrays、Math、Collections等工具類

舉例二：單例模式

舉例三：

**class** Circle{

**private** **double** radius;

**private** **int** id;//自動賦值

**public** Circle(){

id = *init*++;

*total*++;

}

**public** Circle(**double** radius){

**this**();

// id = init++;

// total++;

**this**.radius = radius;

}

**private** **static** **int** *total*;//記錄創建的圓的個數

**private** **static** **int** *init* = 1001;//static聲明的屬性被所有對象所共享

**public** **double** findArea(){

**return** 3.14 \* radius \* radius;

}

**public** **double** getRadius() {

**return** radius;

}

**public** **void** setRadius(**double** radius) {

**this**.radius = radius;

}

**public** **int** getId() {

**return** id;

}

**public** **static** **int** getTotal() {

**return** *total*;

}

}