Diferencias entre static y final

David Corzo, Anesveth Maatens, Ian Jenatz & Adriana Mundo

2019-Nov-14 23:15:56

1. Static & Final

Las palabras "static" & "final" en un programa son keywords que pueden ser aplicadas a una clase, variable o método. La gramática para implementar una variable, clase o método en Java es simplemente poner el keyword "final" o "static" al iniciar la declaración. Uno de los usos más familiares y recurrentes que tiene en particular las estáticas son que se utilizan para el método "main", dicho método como ya se sabe es el punto de entrada al programa; En este ensayo se tratará de explicar cosas como por qué se declara "main" como "static" y para que nos funciona el keyword "final".

2. Definición de "static"

Definición de "Static": *Static en Java es una keyword que es aplicable a clases, variables y métodos; cuando se declara el keyword static antes de un método, variable, etcétera se convierte en global a todos los demás miembros de la clase.* Ahorita cobra cierto valor que el método de entrada al programa (main) por que recordar que se declara como "`public` **`static`** `void main(String[] args) { <<acciones>>; }`".

2.1. Tipos relevantes de "static"

Las variables static tienen ciertas peculiaridades entre ellas, por ejemplo:

- Las variables estáticas tiene la validez de una variable global para cualquier otro miembro de la misma clase.
- No se requiere instanciar un objeto de la clase antes de poder instanciar la variable estática.
- ***Ejemplo:*** `nombre_de_la_clase.miembro_static`; // esto puede accesar a esta variable sin instanciarla previamente.

Los métodos estáticos por otro lado tienen otras características notables que los hacen únicos, y algunas peculiaridades que uno tiene que tener en cuenta si un quiere evadir errores, por ejemplo:

- Los métodos estáticos solo pueden llamar a otros métodos estáticos, sólo pueden accesar información declarada como estática, puede ser accesada de la misma manera que las variables estáticas.

Las clases estáticas por otro lado sólo pueden ser ejecutadas si se instancian primero.

2.2. Implicaciones de "static"

- Los elementos estáticos no pueden usarse con un elemento no estático o llamar a un método no estático directamente.
- Keywords de Java como "this." o "super" no pueden se utilizadas en este contexto.
- Solo pueden acceder a información tipo static.

- Los métodos no pueden ser "overriden".
- Las variables static van como parte de la definición de la clase no en el heap.
- Comprometen el concepto de encapsulación ya que la implementación técnica de las mismas permiten que el estado se mantenga a lo largo de todas las instancias de la clase, esto compromete las encapsulaciones que sean de interés.
- El proceso de garbage collecting es diferente ya que se collecionan como basura cuando la clase en sí se collecta, esto se debe a que los elementos estáticos son parte de la definición de la clase.
- Static se usa primordialmente para la memoria, significa que solo hay una copia de el elemento en memoria que es compartida por la instacia de la clase.

2.2.1. Ejemplo del uso de "static"

```
public class Static {
    static int x1 = 10;
    static int x2 = 10;

    public static void main(String[] args) {
        int sum = x1 + x2;
        System.out.println(sum);
    }
}
```

Recordar lo siguiente: *En el ejemplo anterior si yo fuese a no declarado x1 & x2 como static no las pudiera haber sumado posteriormente en el método main.*

3. Definición de "Final"

Definición de "Final": *En Java se define con el keyword "final" previo a las variables, clases, métodos que se quiere que tengan las características de final, básicamente la función que desempeñan es darle a dicho elemento del programa la característica de que no se puede modificar después en el programa, estas deben de ser inicializadas al momento de su declaración.* En otras palabras cuando se declara un elemento previamente mencionado como final en un programa de Java se le trata como que si fuese constante ya que no se puede modificar.

3.1. Usos de "final"

Entre todos los posibles usos que se le pueden dar a la declaración final en una clase, variable o método se usa y es muy util en algunos casos concretos por ejemplo, declara una variable final sirve para dar la característica que no se modifique dicha variable osea se deja como una constante desde la declaración hasta el final del programa.

Otro ejemplo del uso de final en métodos es que no se permita sobre escribir un método, hay casos que se hereda un método de una clase padre y al pasar esto puede uno modificar el método que se heredo, hay veces que por razones varias se quiere que se use el método pero que no se modifique por lo que se usa final para asegurar que no se pueda modificar dicho método.

Otro ejemplo es cuando se declara una clase como final se previene evitar herencia, este es otro uso de el keyword final en Java.

3.1.1. Implicaciones del uso de "final"

- Se debe de tener en cuenta que siempre que se declare un elemento de tipo final se debe inicializar en algún momento en el programa, de lo contrario Java tirará un error de compilación.
- Final no es modificable, irónicamente una variable final es una constante.
- Se puede agregar o remover objetos de un array con declaración final.
- Se tiene la convención entre programadores de siempre declarar las variables de tipo final en mayúsculas.
- Si no se inicializa un elemento estático en el momento de su declaración, se debe de declarar dentro de un "instance-initializer block" o dentro de un constructor, si se tiene múltiples constructores en la clase se debe inicializar en todos, otra manera es dentro de un "static block".

3.1.2. Ejemplo de declaración de "final"

```
class Final {
    public static void main(String[] args) {
        final double pi = 3.14; // aproximado
        System.out.println(pi);
    }
}
```

4. Best of both worlds, final static together

Usar static o final no es un asunto mutuamente excluyente, se pueden declarar variables que son static y final a la vez, por ejemplo:

```
final static int x = 12;
```

4.1. Implicación de usar las dos

- Se debe inicializar la variable en algún momento del programa.
- Los métodos de inicialización son los mismos que se hablaron en las implicaciones de final. (static blocks por ejemplo).

```
class Algo {
    final static int x;
    {
        x = 12;
    }
}
```

- No son modificables media vez estén inicializadas con un valor.
- Declarar una variable como final static es lo mismo que crear una constante.

5. Fuentes de investigación

[1]: http://gopaldas.org/core-java/advantages-and-disadvantages-of-using-static-in-java/

[2]: https://javarevisited.blogspot.com/2013/07/when-to-make-method-static-in-java.html

[3]: https://www.javatpoint.com/static-keyword-in-java

[4]: https://www.geeksforgeeks.org/final-keyword-java/

[5]: https://www.geeksforgeeks.org/final-static-variable-java/