

Unitat 05. Documentació

Entorns de desenvolupament



Contingut

Introducció.....	3
Comentaris interns	3
JavaDoc.	3
Etiquetes JavaDoc.....	5
Generació de la documentació	6
Des de la línia d'ordres	6
Des de NetBeans	6
Documentació generada	8
Exemple: classe Cercle	9

Introducció

Documentar el codi d'un programa és dotar el programari de tota la informació que siga necessària per explicar el que fa. Els desenvolupadors que duen a terme el programari (i la resta de l'equip de treball) han d'entendre què està fent el codi i el perquè.

Comentaris interns

Una forma molt senzilla de documentar el programa és utilitzant els comentaris interns al codi font. Cada llenguatge té la seua forma d'implementar comentaris, però en general s'utilitzen símbols com ara `/* */` i `//` i altres similars.

En Java les formes de mostrar els comentaris intercalats amb el codi font són:

- afegint els caràcters `//` davant del comentari. En eixe cas, el compilador considerarà com a comentari tot el text des dels caràcters `//` fins al final de la línia
- afegint els caràcters `/*` al principi d'un bloc de text, i `*/` al final. D'esta manera el compilador considerarà com a comentari tot el text inclòs entre els símbols `/*` i `*/`

Cal tindre en compte que este tipus de comentaris no tenen cap altra utilitat només que servir al programador de guia, de recordatori del que fa cada funció, línia de codi, etc. No se genera automàticament cap documentació utilitzant els comentaris d'este tipus.

JavaDoc.

La utilitat JavaDoc permet la creació de documentació d'APIs en format HTML. Actualment és un estàndard per crear comentaris de classes en Java.

La sintaxi utilitzada per a crear comentaris assimilables per JavaDoc és la següent:

```
/** Inici dels comentaris de JavaDoc
 * línia de comentari
 * línia de comentari
 ...
 */ final del comentaris
```

Cada línia pot incloure certes etiquetes, tant pròpies de JavaDoc com del llenguatge HTML. Les etiquetes pròpies de JavaDoc comencen amb el caràcter `@`.

```
/** Comentaris de JavaDoc
 * De forma automàtica es generarà una pàgina HTML
 * @etiqueta1 text específic de l'etiqueta1
 * @etiqueta2 text específic de l'etiqueta2
 */
```

Els comentaris Javadoc estan pensats per ser utilitzats al principi de cada classe, mètode, interfície, atribut, etc. Només en eixe cas són reconeguts per Javadoc. Els comentaris que s'escriuen en el cos d'un mètode no serveixen per generar documentació. L'eina Javadoc només reconeix un comentari per documentació per a cada declaració.

Per exemple, un error habitual és escriure una sentència import entre el comentari de la classe i la seua declaració. Els comentaris, en eixe cas, se reconeixen com a comentaris però Javadoc no els podrà utilitzar per generar documentació.

MAL

```
/ **
* Comentari per a la documentació per a la classe
* /
import java.util.*; // ERROR
public class nom_de_la_classe
{...
```

BÉ

```
import java.util.*;
/ **
* Comentari per a la documentació per a la classe
* /
public class nom_de_la_classe
{...
```

Un comentari Javadoc es compon d'una descripció principal seguida d'una secció d'etiquetes (tags). La descripció principal comença després del marcador de principi de comentari / ** i segueix fins a la secció d'etiquetes. El principi d'esta secció està marcat pel primer caràcter @. La primera frase hauria de descriure l'entitat o element declarat. Se considera que la primera frase acaba quan troba un punt seguit d'un espai en blanc, un tabulador o una marca de final de línia, o també quan troba la primera etiqueta @. La eina Javadoc còpia aquesta primera frase en el resum de l'entitat, a la part superior de la pàgina HTML de la documentació que genera.

La descripció principal no pot continuar un cop iniciada la secció d'etiquetes. Ara bé, és possible tindre un comentari només amb secció d'etiquetes, sense descripció principal.

```
/ **
* Esta frase seria la descripció principal de l'element la
* declaració s'iniciaria just després d'este comentari
* @author Laboratori de LP
* /
```

Com s'ha comentat, els comentaris Javadoc permeten afegir codi HTML en la seua descripció. Així podem donar el format desitjat a la documentació generada per Javadoc i també afegir enllaços, tal com veurem més endavant.

Etiquetes JavaDoc.

Hi ha dos tipus d'etiquetes:

- Etiquetes de bloc: només es poden utilitzar en la secció d'etiquetes que segueix a la descripció principal. Són de la forma: `@etiqueta`
- Etiquetes inline: es poden utilitzar tant en la descripció principal com en la secció d'etiquetes. Són de la forma: `{@tag}`, és a dir, s'escriuen entre els símbols de claus.

Algunes etiquetes:

<code>@author nom_autor</code>	Especifica l'autor de l'entitat. Per a que aparega en la documentació generada per Javadoc s'ha d'utilitzar l'opció <code>-author</code> . És vàlida per a documentar paquets i classes.
<code>@version</code>	Especifica la versió. Per a paquets i classes.
<code>@param nom descripció</code>	Afegeix un paràmetre a la secció "Paràmetres". La descripció es pot escriure en més d'una línia. És vàlid en els comentaris per a documentació de mètodes i constructors.
<code>@return descripció</code>	Afegeix una secció "Returns" amb el text "descripció". Descriu el tipus retornat i el rang de possibles valors. Vàlid en els comentaris de mètodes.
<code>@see referencia</code>	Afegeix una capçalera "See Also" amb un enllaç que apunta a una referència. Un comentari per documentació pot contenir més d'una d'estes etiquetes: tots els enllaços d'este tipus s'agruparan sota la mateixa capçalera.
<code>{@link package.class #member etiqueta}</code>	Crea un enllaç amb el text "etiqueta" que apunta a la documentació del paquet, classe o membre especificat. El nom de la classe ("package.class") se pot ometre si el mètode referenciat està en la mateixa classe on està el comentari. Similar a <code>@see</code> però genera el link en el mateix lloc on s'ha posat, no en la capçalera "See also".
<code>@since text</code>	Afegeix a la documentació una capçalera "Since" amb el "text" especificat. Serveix per dir a partir de quina versió del programa està disponible l'entitat declarada.
<code>@throws (ó @exception) nom_classe descripció</code>	Nom (i opcionalment descripció) de la classe llançada quan hi ha una excepció. Vàlid en la documentació de mètodes i constructors.
<code>@deprecated text</code>	Adverteix que una determinada funcionalitat no s'hauria d'utilitzar perquè ha quedat obsoleta.

Hi ha més etiquetes que pots veure en la documentació oficial de Javadoc.

<https://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html>

Generació de la documentació

Se pot generar la documentació des de la línia d'ordres amb la instrucció javadoc, o també des de qualsevol IDE que la porte integrada.

Des de la línia d'ordres

La sintaxi de la instrucció javadoc inclou moltes opcions que seria molt llarg enumerar. Podeu veure totes les possibilitats en el següent enllaç:

<https://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html#options>

Les opcions bàsiques serien:

-author	per incloure l'autor (no ho fa per defecte)
-version	per incloure la versió (no ho fa per defecte)
-d	per especificar en quina carpeta volem que guardi la documentació (per defecte ho farà en la carpeta actual)
-sourcepath	per dir-li en quina carpeta està el paquet

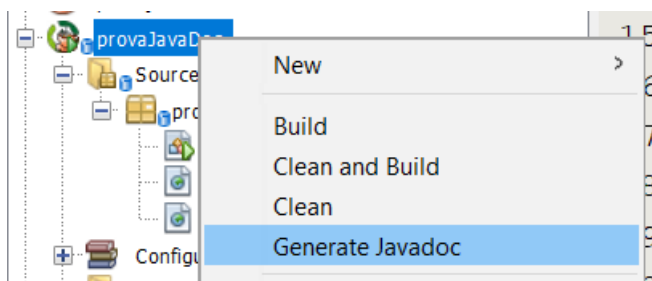
Per exemple, si tenim un projecte anomenat `provaJavaDoc` en la nostra carpeta `NetBeansProjects`, amb una estructura `src\main\java\provaJavaDoc` i el paquet s'anomena `provaJavaDoc`, la generació de la documentació des de la línia d'ordres la podríem fer, des de l'arrel del projecte, així:

```
javadoc -author -version -d docs -sourcepath src\main\java provaJavaDoc
```

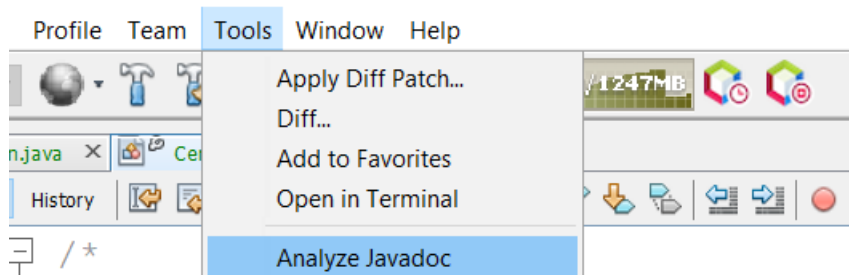
Des de NetBeans

Des d'un IDE és prou més senzill, encara que no he trobat com fer que mostri l'autor i la versió.

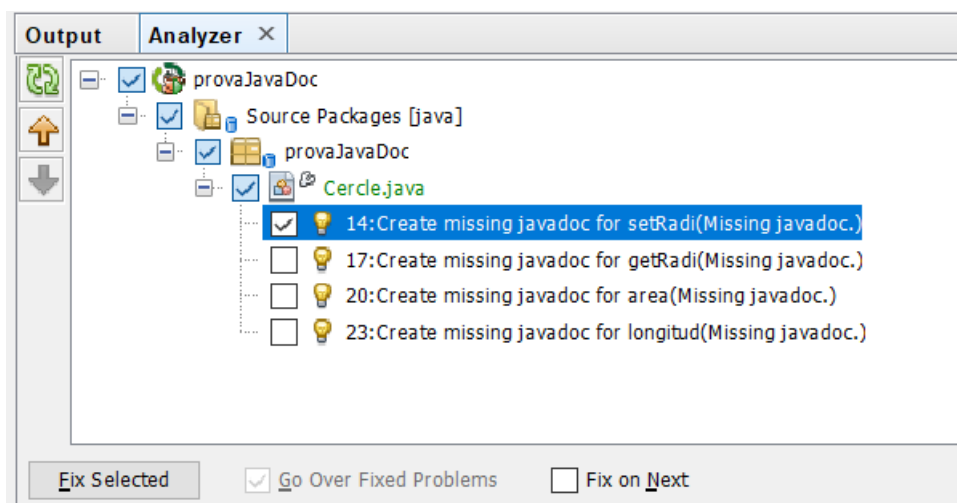
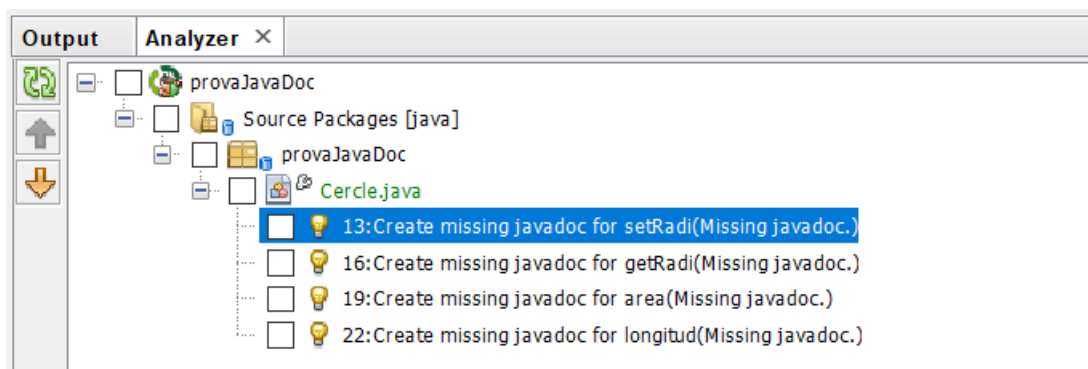
Simplement hi ha que posar-se sobre el projecte, i amb el botó de la dreta seleccionar l'opció `Generate Javadoc`.



En NetBeans també tenim l'opció de fer una anàlisi prèvia a la generació de la documentació, veure quins elements no tenim documentats i que els documente automàticament. Ho podem fer situant-nos sobre el codi (o el paquet) que volem analitzar, i anant a **Tools -> Analyze Javadoc**



Ens mostrarà una finestra on podem veure tots els elements que no tenim documentats, i ens donarà l'opció de fer-ho, seleccionant aquells que volem documentar i polsant el botó "Fix Selected".

























I ens generarà automàticament la documentació mínima necessària, que normalment haurem de completar.

Documentació generada

La documentació serà generada en format HTML i estarà disponible per defecte, si ho fem des de NetBeans, en la carpeta `build\docs\javadoc` dins del nostre projecte. Si ho fem des de la línia d'ordres, en la carpeta especificada amb l'opció `-d`

Si entrem a la carpeta javadoc veurem una llista d'arxius més o menys així:

 jquery	28/02/2021 22:25	Carpeta de archivos	
 provaJavaDoc	28/02/2021 22:40	Carpeta de archivos	
 resources	28/02/2021 22:25	Carpeta de archivos	
 allclasses	28/02/2021 22:40	Chrome HTML Do...	2 KB
 allclasses-index	28/02/2021 22:40	Chrome HTML Do...	6 KB
 allpackages-index	28/02/2021 22:40	Chrome HTML Do...	5 KB
 constant-values	28/02/2021 22:40	Chrome HTML Do...	5 KB
 deprecated-list	28/02/2021 22:40	Chrome HTML Do...	5 KB
 element-list	28/02/2021 22:40	Archivo	1 KB
 help-doc	28/02/2021 22:40	Chrome HTML Do...	10 KB
 index	28/02/2021 22:40	Chrome HTML Do...	1 KB
 index-all	28/02/2021 22:40	Chrome HTML Do...	8 KB
 member-search-index	28/02/2021 22:40	Archivo JavaScript	1 KB
 member-search-index	28/02/2021 22:40	Archivo WinRAR ZIP	1 KB
 overview-tree	28/02/2021 22:40	Chrome HTML Do...	5 KB
 package-search-index	28/02/2021 22:40	Archivo JavaScript	1 KB
 package-search-index	28/02/2021 22:40	Archivo WinRAR ZIP	1 KB
 script	28/02/2021 22:40	Archivo JavaScript	6 KB
 search	28/02/2021 22:40	Archivo JavaScript	13 KB
 stylesheet	28/02/2021 22:40	Documento de hoj...	23 KB
 type-search-index	28/02/2021 22:40	Archivo JavaScript	1 KB
 type-search-index	28/02/2021 22:40	Archivo WinRAR ZIP	1 KB

Si obrim l'arxiu index amb un navegador que tinga JavaScript activat, veurem la pantalla principal de la nostra documentació.

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

Package provaJavaDoc

Class Summary

Class	Description
Cercle	
Main	

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

Des del menú principal podem veure l'índex amb tots els elements documentats, les classes, cadascuna de les classes amb els seus mètodes, un arbre amb tota la informació, etc.

Exemple: classe Cercle

Anem a fer un programa que permeti crear un cercle, amb un radi, i després mostri la seua superfície i la longitud de la seua circumferència.

Primer anem a fer la classe Cercle en un arxiu Cercle.java:

```
public class Cercle {
    private float radi;

    public void setRadi(float radi) {
        this.radi=radi;
    }
    public float getRadi() {
        return this.radi;
    }
    public double area() {
        return Math.pow(this.radi,2)*Math.PI;
    }
    public double longitud() {
        return 2*Math.PI*this.radi;
    }
}
```

I ara el main en un arxiu Main.java:

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Cercle c=new Cercle();
        c.setRadi(5);
        System.out.println("La superfície del cercle amb radi
"+c.getRadi()+" és: "+c.area());
        System.out.println("La longitud del cercle amb radi
"+c.getRadi()+" és: "+c.longitud());
    }

}
```

Podem analitzar els arxius i que genere les etiquetes automàticament, o fer tot el procés manualment des de l'editor. En qualsevol cas, el resultat de documentar la classe Cercle hauria de ser aproximadament el següent:

```

package provaJavaDoc;

/**
 *
 * @author fidel
 * @version 1.0
 */
public class Cercle {
    private float radi;

    /**
     * Mètode set per assignar-li valor al radi del cercle
     * @param radi és el radi del cercle
     */
    public void setRadi(float radi) {
        this.radi=radi;
    }

    /**
     * Mètode per a recuperar el valor del radi del cercle
     * @return torna el radi del cercle
     */
    public float getRadi() {
        return this.radi;
    }

    /**
     * Mètode per a tornar la superfície del cercle
     * Ho fa amb la fórmula  $\text{radi}^2 \cdot \pi$ 
     * Necessitem:
     * <ul><li>El valor de PI, determinat per la constant Math.PI</li>
     * <li>El valor del radi del cercle, que el tenim en l'atribut radi</li>
     * <li>La funció Math.pow per elevar el radi al quadrat</li></ul>
     * @return torna la superfície
     */
    public double area() {
        return Math.pow(this.radi,2)*Math.PI;
    }

    /**
     * Mètode per a tornar la longitud de la circumferència
     * La calcula amb la fórmula  $2 \cdot \pi \cdot \text{radi}$ 
     * Necessitem:
     * <ul><li>El valor de PI, determinat per la constant Math.PI</li>
     * <li>El valor del radi del cercle, que el tenim en l'atribut
     * radi</li></ul>
     * @return torna la longitud de la circumferència
     */
    public double longitud() {
        return 2*Math.PI*this.radi;
    }
}

```

I quan generem la documentació amb JavaDoc, veuríem el següent:

PACKAGE

CLASS

TREE

DEPRECATED

INDEX

HELP

ALL CLASSES

Package provaJavaDoc

Class Summary	
Class	Description
Cercle	
Main	

PACKAGE

CLASS

TREE

DEPRECATED

INDEX

HELP

ALL CLASSES

Com abans. Però si fem clic en la classe Cercle veurem:

PACKAGE

CLASS

TREE

DEPRECATED

INDEX

HELP

ALL CLASSES

SEARCH:

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
double	area()	Mètode per a tornar la superfície del cercle Ho fa amb la fórmula radi*pi² Necessitem: El valor de PI, determinat per la constant Math.PI El valor del radi del cercle, que el tenim en l'atribut radi La funció Math.pow per elevar el radi al quadrat
float	getRadi()	Mètode per a recuperar el valor del radi del cercle
double	longitud()	Mètode per a tornar la longitud de la circumferència La calcula amb la fórmula 2*pi*radi Necessitem: El valor de PI, determinat per la constant Math.PI El valor del radi del cercle, que el tenim en l'atribut radi
void	setRadi (float radi)	Mètode set per assignar-li valor al radi del cercle

I si baixem a un mètode concret, per exemple area:

area

```
public double area()
```

Mètode per a tornar la superfície del cercle Ho fa amb la fórmula **radi*pi²** Necessitem:

- El valor de PI, determinat per la constant Math.PI
- El valor del radi del cercle, que el tenim en l'atribut radi
- La funció Math.pow per elevar el radi al quadrat

Returns:

torna la superfície

I si anem a l'opció "Index" del menú de la documentació, trobarem un índex amb tots els elements:

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)[ALL CLASSES](#)[A](#) [C](#) [G](#) [L](#) [M](#) [P](#) [S](#)[All Classes](#) [All Packages](#)

A

area() - Method in class [provaJavaDoc.Cercle](#)

Mètode per a tornar la superfície del cercle Ho fa amb la fórmula **radi*pi²** Necessa: funció Math.pow per elevar el radi al quadrat

C

Cercle - Class in [provaJavaDoc](#)

Cercle() - Constructor for class [provaJavaDoc.Cercle](#)

G

getRadi() - Method in class [provaJavaDoc.Cercle](#)

Mètode per a recuperar el valor del radi del cercle

L

longitud() - Method in class [provaJavaDoc.Cercle](#)

Mètode per a tornar la longitud de la circumferència La calcula amb la fórmula : l'atribut radi