MODULARITAT

Documentació de classes en JAVA.

Un dels principals problemes que es troben els programadors a l'hora d'implementar programes reutilitzant classes de tercers, està centrat en l'aprenentatge. Per poder usar adequadament una classe cal saber com es diuen els seus mètodes, quins paràmetres necessiten, què fan, què retornen, quins errors es poden produir en cridar-los i sota quines condicions es poden evitar, etc.

La documentació de les classes es torna imprescindible quan modularitzem les aplicacions i reutilitzem biblioteques de classes de caràcter genèric en múltiples ocasions. Fins i tot, malgrat que les biblioteques siguin pròpies, és important documentar-les adequadament, per evitar lapsus de memòria i facilitar el seu ús a d'altres programadors.

D'entre tots els comentaris que generalment s'insereixen dins el codi quan s'implementa una classe, distingirem aquells que tenen per objectiu informar al programador d'aspectes tècnics del codi dels que s'escriuen per tal d'informar a l'usuari de com cam usar els mètodes d'una classe i en quines situacions poden fer-se servir.

Mentre els primers són necessaris només si cal fer alguna modificació en el codi de la pròpia classe, els segons es fan indispensables per poder fer servir les classes com biblioteques externes. Sovint en usar biblioteques de tercers no disposem d'accés al codi i l'única informació de la que disposem serà la documentació de l'API generada amb un javadoc. És més, a vegades l'accés al codi en biblioteques complexes podria dificultar l'aprenentatge, més que facilitar-lo, doncs com ja s'ha vis en el disseny descendent, moltes vegades, els mètodes tenen la capacitat d'amagar la complexitat i facilitar-ne el seu ús.

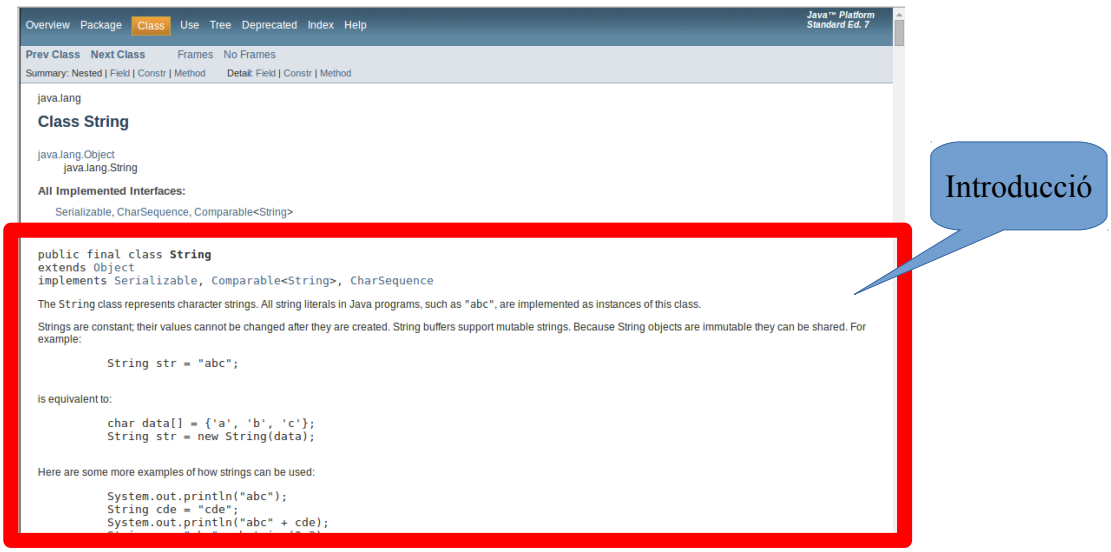
Es important separar adequadament les informacions destinades a explicar el codi de les destinades a explicar la seva funcionalitat i ús. Les primeres mai formaran par de la documentació de cap API sinó que restringiran la seva localització, exclusivament dins el codi. Les segones en canvi serviran per generar la documentació de l'API fent servir l'eina específica de JAVA anomenada JAVADOC.

L'eina JAVADOC distingirà els comentaris a exportar en format HTML perquè aquest sempre començaran per la seqüència /** (doble asterisc després de la barra). És a dir, si volem escriure un comentari detallant el codi haurem de fer servir la doble barra // o la seqüència /* (un únic asterisc després de la barra). Així per exemple, observeu que el primer comentari conté informació funcional de què fa i com cal usar el mètode *entrarEnter*, per això s'encapçala amb una barra i dos asteriscs, mentre que la resta de comentaris són detalls que clarifiquen el codi, no han d'interessar-nos si només volem usar el mètode. Aquest comentaris mai podran començar amb una barra i dos asteriscs.

```java
/**
* Mostra per pantalla el missatge del paràmetre i retorna el valor numèric
* entrat per l'usuari. En cas que l'usuari premi ENTRAR sense
* haver escrit cap número, s'informarà a l'usuari que s'està esperant un
* valor numèric.
* @param missatge és el missatge a mostra per pantalla abans de demanar
* l'entrada.
* @return valor enter introduït per l'usuari.
*/
int entrarEnter(String missatge){
	Scanner scanner = new Scanner(System.in);
	int enter = 0;
	boolean correcte=false;
	do{
		System.out.println();
		System.out.print(missatge);
		/*
		 * comprova si hi ha un enter al buffer del teclat
		 */
		correcte=scanner.hasNextInt();
		if(correcte){ // Si hi ha un enter l'extreu
			enter=scanner.nextInt();
			scanner.nextLine();
		}else{ // Si no hi ha un enter, neteja el buffer i avisa a l'usuari
			scanner.nextLine();
			System.out.print("Cal que introduïu un valor enter");
		}
	}while(!correcte); /*Surt si s'ha aconseguit extreure el valor o es
			repeteix la demanda en cas contrari*/
	return enter;
}
```

És important doncs, plantejar-se quin tipus de comentari estem fent per valorar si caldrà fer una explicació més funcional

El format de documentació especificat pel llenguatge JAVA, indica que a la documentació d'una classe cal trobar-hi una introducció que expliqui la funcionalitat global de la classe i exemplifiqui si cal, com s'ha de fer servir.



El lloc on haurem d'escriure aquesta informació quan implementem la classe, haurà de situar-se immediatament a sobre de la declaració de la classe i com ja s'ha dit caldrà iniciar el comentari amb una barra i un doble asterisc. Si no ho fem així, Javadoc no ho reconeixerà com a documentació. Veiem un exemple:

```
/*
 * Aquest comentari no és documentació. Javadoc no el reconeixerà com a tal perquè no s'ha iniciat
 * amb un doble asterisc.
 */
package exemplesdissenydescendent;

/**
 * Descripció que indica què fa la classe i si cal com es fa servir. També es poden posar
 * exemples d'ús o qualsevol altre cosa que ajudi a un programador a saber com cal usar la
 * classe. Aquest comentari serà reconegut per Javadoc com la introducció i es posarà a
 * l'inici del document.
 * @author És habitual signar l'autoria de les classes amb el nom i un correu electrònic.
 */
public class ExempleDUnaClasseQualsevol {

    ...
}
```

A més de la introducció, cada mètode de la classe hauria de tenir un comentari, iniciat per la barra i dos asteriscs, explicant la funcionalitat específica del mètode, quins paràmetres necessita, quins valors poden prendre els paràmetres i quins valors no haurien de prendre. Si ho fa, caldria indicar també quin valor retorna. No es tracta d'explicar el codi, sinó la utilitat del mètode. Observeu el comentari del mètode *entrarEnter*.

L'IDE disposa de moletes ajudes per facilitar l'escriptura i la generació de la documentació JAVADOC. A l'hora d'escriure, si situem el cursor damunt el nom de la classe i d'un dels mètodes a documentar i escrivim la barra i doble asterisc i premem ENTRAR, Netbeans acabarà completant l'estructura bàsica del comentari amb els delimitadors de comentaris i amb els camps JAVADOC necessaris (paràmetres, retorn, etc.).

Malgrat tot, aquesta ajuda només ens pot servir quan comencem a documentar, però si modifiquem un mètode, afegint paràmetres o un retorn, segur que no ens interessa canviar tot el comentari. En aquest casos disposem d'una eina interna de Netbeans capaç d'analitzar els comentaris, detectar si falta alguna cosa i proposar canvis. Per activar-la poseu el ratolí damunt la classe que volem analitzar en la finestra projects, premeu el botó dret, seleccioneu la opció *Tools* i en obrir-se el desplegable escolliu *Analize javadoc*. Això obrirà una finestra on podreu visualitzar tots els problemes detectats. Seleccioneu els que cregueu oportú, cliqueu *Fix Selected* i l'editor intentarà fer una proposta per arreglar-ho.

La generació de la documentació usant l'eina javadoc és també molt senzilla. Només cal situar el punter del ratolí sobre el projecte (a la finestra *Projects*), clicar el botó dret i seleccionat *Generate Javadoc.* Si els comentaris estan ben escrits, això generarà una carpeta dins del projecte amb els documents HTML necessaris per visualitzar la documentació en un navegador. El contingut es generarà dins el directori *dist/javadoc* situat a la carpeta on es guarda el projecte.