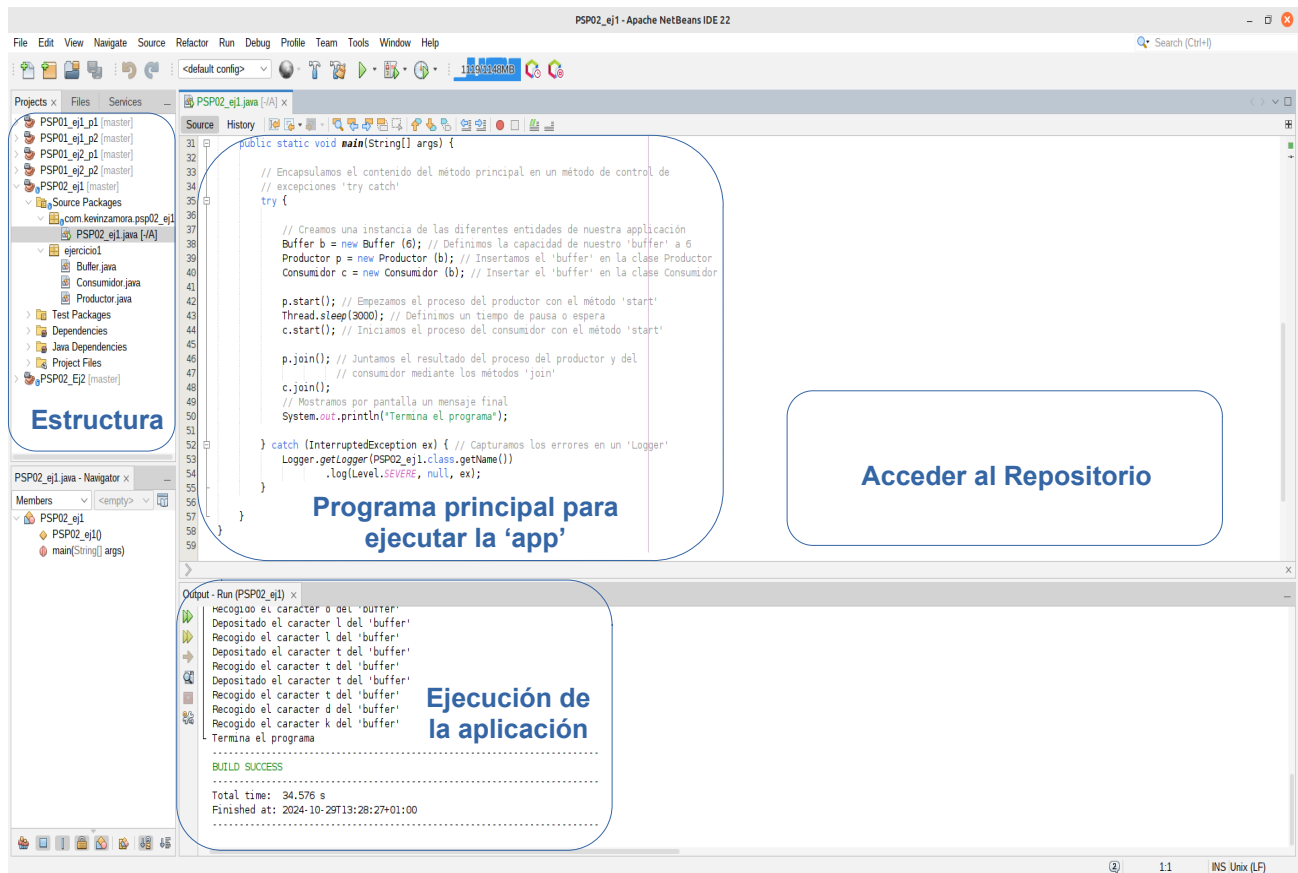
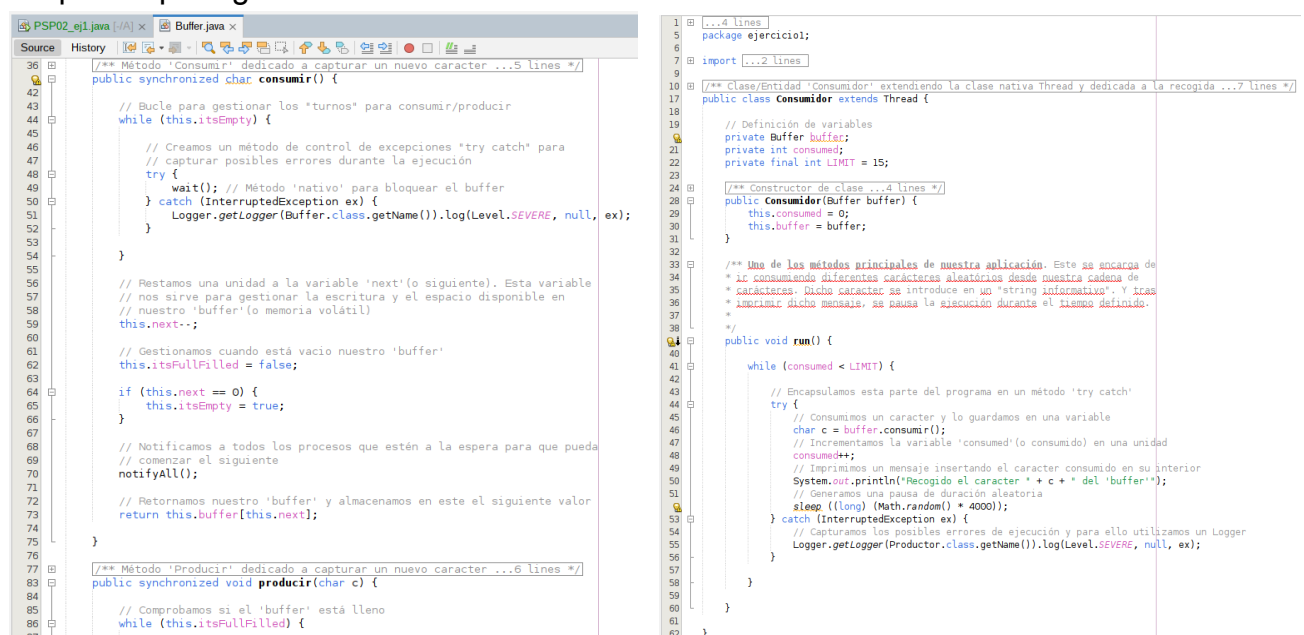


Ejercicio 1

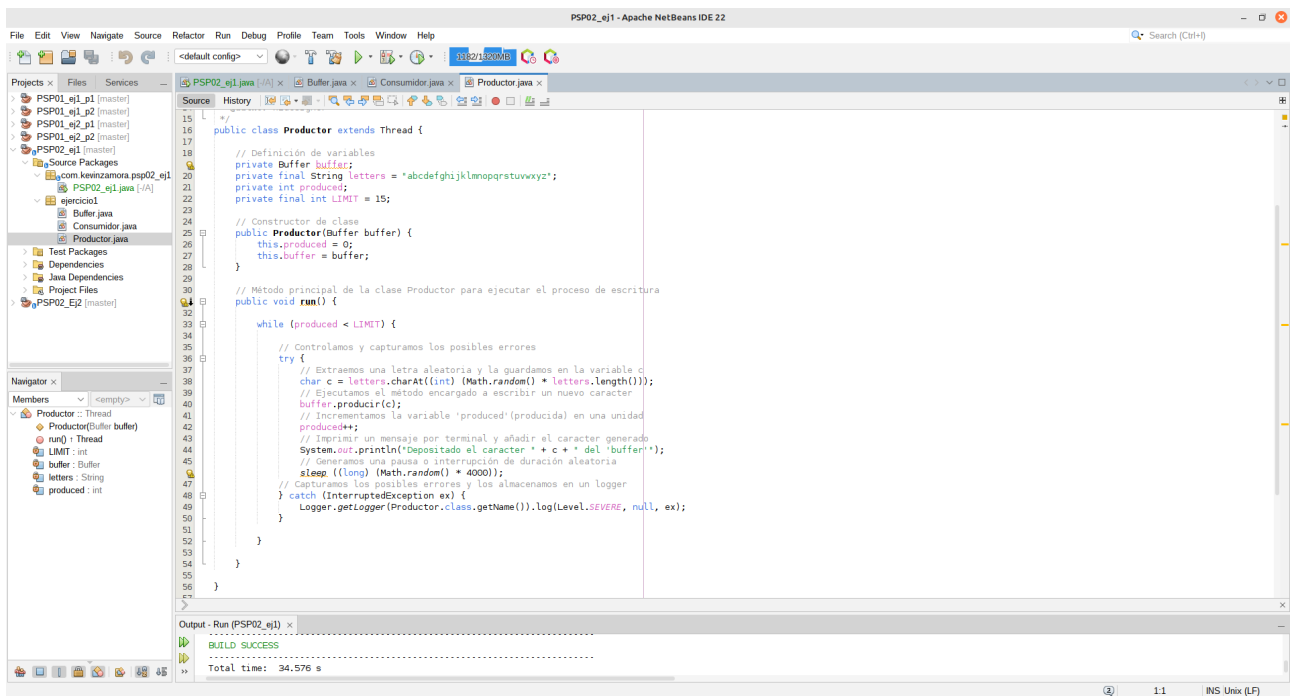
- Creamos la estructura del proyecto e inicializamos las diferentes clases/entidades necesarias ("Buffer", Consumidor, Productor y la "auxiliar 'PSP02_ej1'", para ejecutarlo).



Desarrollamos las entidades/clases 'Buffer' y 'Consumidor' y añadimos comentarios y 'etiquetas' para generar documentación mediante Java Docs.

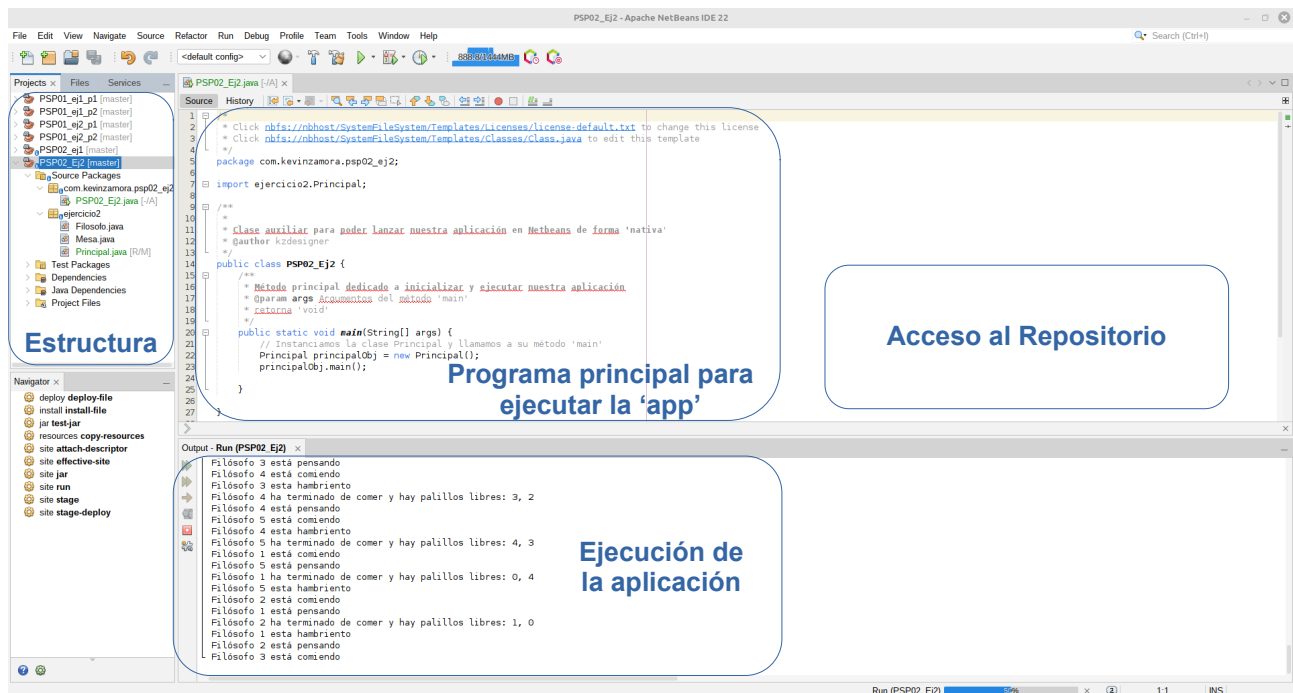


Desarrollamos la entidad/clase 'Productor' y añadimos comentarios y 'etiquetas' para generar documentación mediante Java Docs.

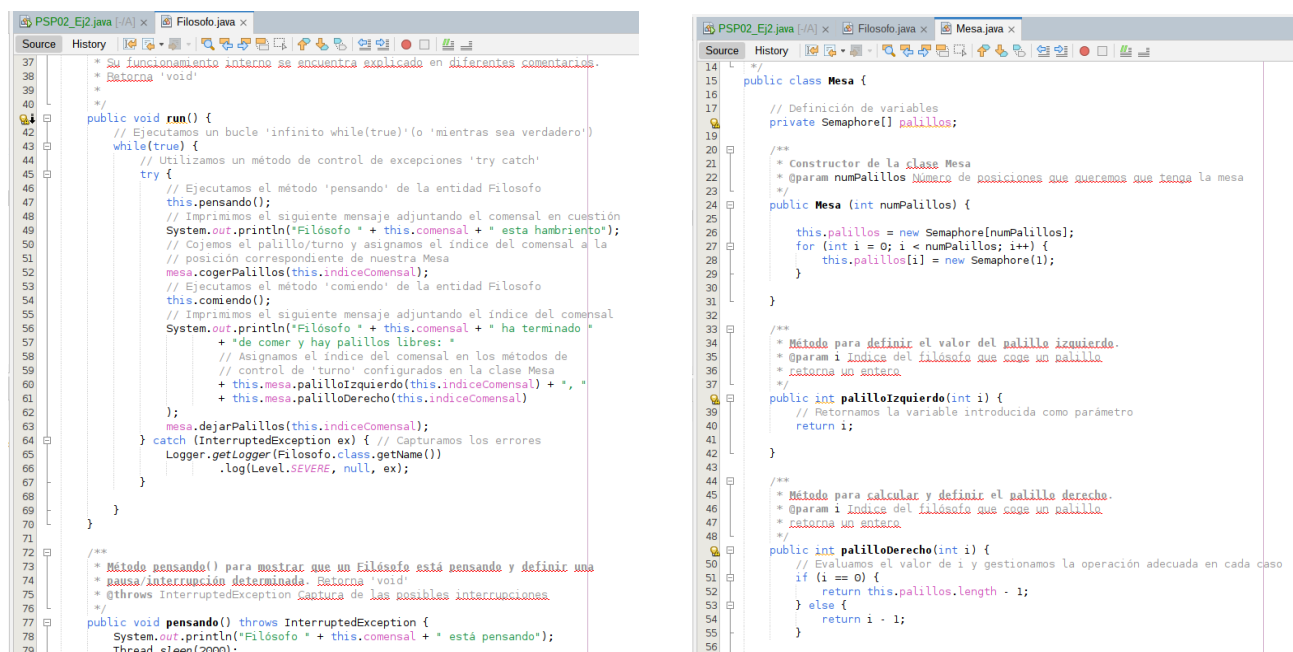


Ejercicio 2

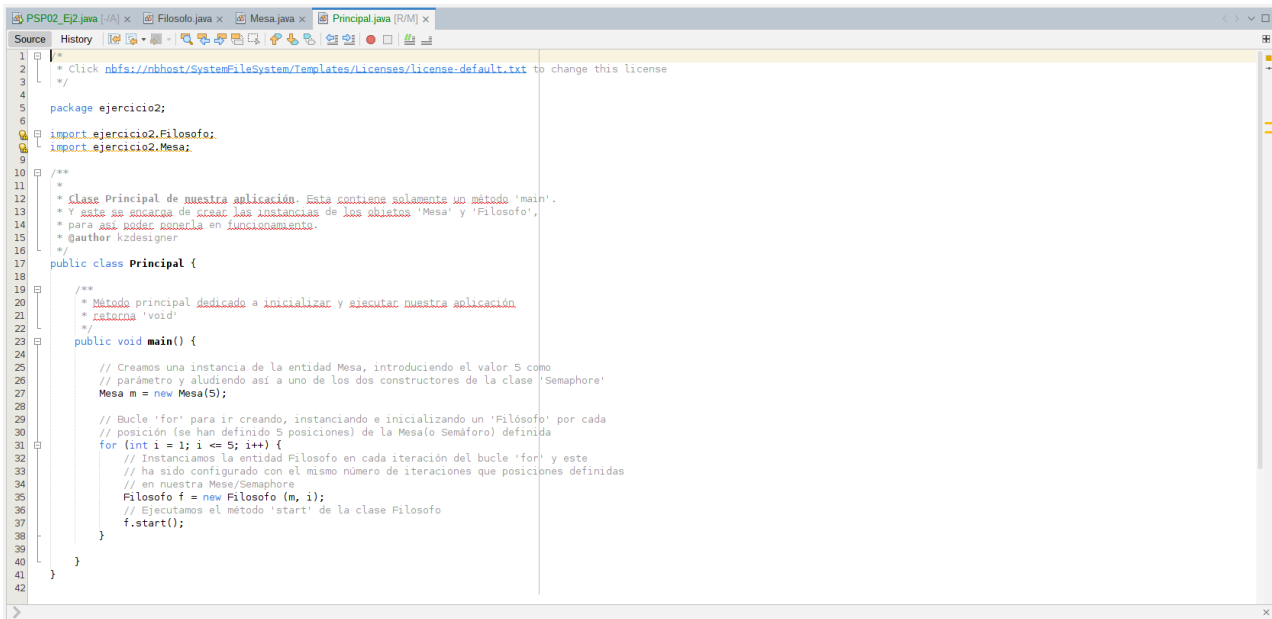
- Creamos la estructura del proyecto e inicializamos las diferentes clases/entidades necesarias (“Filosofo”, “Mesa”, “Principal” y PSP02_Ej2, para ejecutarlo).



Desarrollamos la entidad/clase ‘Filosofo’, con sus métodos ‘pensando’ y ‘comiendo’, y la entidad/clase ‘Mesa’, con los métodos ‘palilloIzquierdo’ y ‘palilloDerecho’, para gestionar los ‘turnos/semáforos’, y con los métodos ‘cogerPalillos’ y ‘dejarPalillos’, dedicados a ‘capturar’ y ‘liberar’ los diferentes palillos/turnos. Acto seguido, añadimos los comentarios pertinentes, en el código HTML, y también añadimos las ‘etiquetas Java Doc’, con las cuales poder generar la documentación de nuestra aplicación mediante Java Docs.

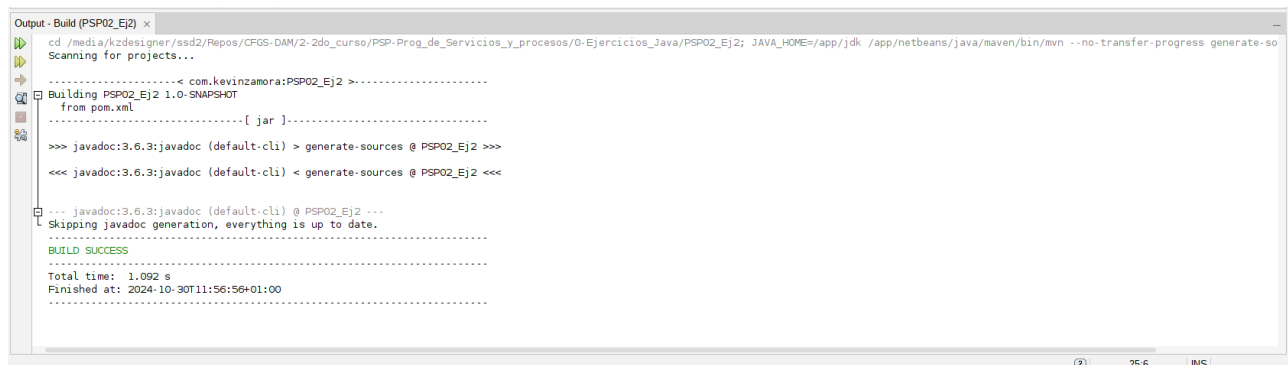


Desarrollamos la entidad/clase 'Principal' para poder inicializar la configuración básica de nuestra aplicación, a modo de 'servicio', para así 'esconder levemente'/dificultar dicha configuración y dividir dicha aplicación en 'módulos funcionales'. A su vez, también añadimos 'comentarios HTML' y 'Etiquetas Java Doc' en estas, para generar posteriormente la documentación mediante la herramienta Java Docs.



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license.default.txt to change this license
3  */
4
5  package ejercicio2;
6
7  import ejercicio2.Filosofo;
8  import ejercicio2.Mesa;
9
10 /**
11  * Clase Principal de nuestra aplicación. Esta contiene solamente un método 'main'.
12  * Y este se encarga de crear las instancias de los objetos 'Mesa' y 'Filosofo',
13  * para así poder ponerla en funcionamiento.
14  * @author kzdesigner
15  */
16
17 public class Principal {
18
19     /**
20     * Método principal dedicado a inicializar y ejecutar nuestra aplicación
21     * retorna 'void'
22     */
23     public void main() {
24
25         // Creamos una instancia de la entidad Mesa, introduciendo el valor 5 como
26         // parámetro y aludiendo así a uno de los dos constructores de la clase 'Semaphore'
27         Mesa m = new Mesa(5);
28
29         // Bucle 'for' para ir creando, instanciando e inicializando un 'Filosofo' por cada
30         // posición (se han definido 5 posiciones) de la Mesa(o Semáforo) definida
31         for (int i = 1; i <= 5; i++) {
32             // Instanciamos la entidad Filosofo en cada iteración del bucle 'for' y este
33             // ha sido configurado con el mismo número de iteraciones que posiciones definidas
34             // en nuestra Mesa/Semaphore
35             Filosofo f = new Filosofo (m, i);
36             // Ejecutamos el método 'start' de la clase Filosofo
37             f.start();
38         }
39     }
40 }
41
42
```

Para generar dicha documentación con Java Doc, seleccionamos el directorio raíz de cada una de nuestras aplicaciones, hacemos 'clic derecho' desde Netbeans, sobre este, y seleccionamos la opción 'Generar JavaDoc' o 'Generate JavaDoc'.

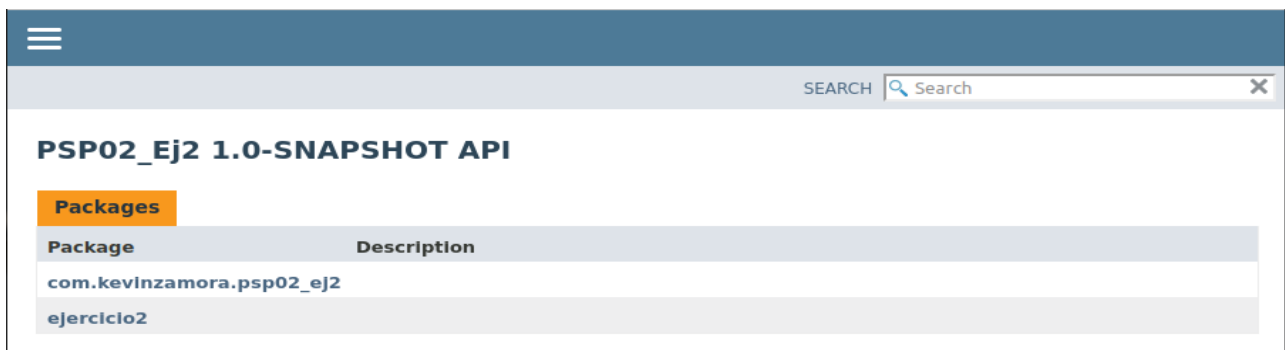


```
Output - Build (PSP02_Ej2) x
cd /media/kzdesigner/ssd2/Repos/CFG5-DAM/2-2do_curso/PSF-Prog_de_Servicios_y_procesos/0-Ejercicios_Java/PSF02_Ej2; JAVA_HOME=/app/jdk /app/netbeans/java/maven/bin/mvn --no-transfer-progress generate-so
Scanning for projects...
-----< com.kevinzamora:PSP02_Ej2 >-----
Building PSP02_Ej2 1.0-SNAPSHOT
from pom.xml
-----[ jar ]-----
>>> javadoc:3.6.3:javadoc (default-cli) > generate-sources @ PSP02_Ej2 >>>
<<< javadoc:3.6.3:javadoc (default-cli) < generate-sources @ PSP02_Ej2 <<<
... javadoc:3.6.3:javadoc (default-cli) @ PSP02_Ej2 ...
Skipping javadoc generation, everything is up to date.
-----
BUILD SUCCESS
-----
Total time: 1.092 s
Finished at: 2024-10-30T11:56:56+01:00
-----
```

Y para acceder a la documentación generada, nos dirigimos hacia el directorio llamado 'target' (u 'objetivo'), el cual está contenido dentro del repositorio raíz de nuestro proyecto. Dentro de este, nos dirigimos hacia el directorio 'site' (o 'sitio'), accediendo a su vez a la única 'carpeta' que encontraremos en su interior, la cual se llama 'apidocs'. Seguidamente y en su interior, encontraremos un archivo 'index.html', entre otros archivos y directorios. *(La presente explicación continua en la siguiente página)*

Al abrir dicho archivo 'index.html' con nuestro navegador web elegido (Mozilla Firefox, Google Chrome, Opera, Brave, etc.), aparecerá una aplicación web como la siguiente:

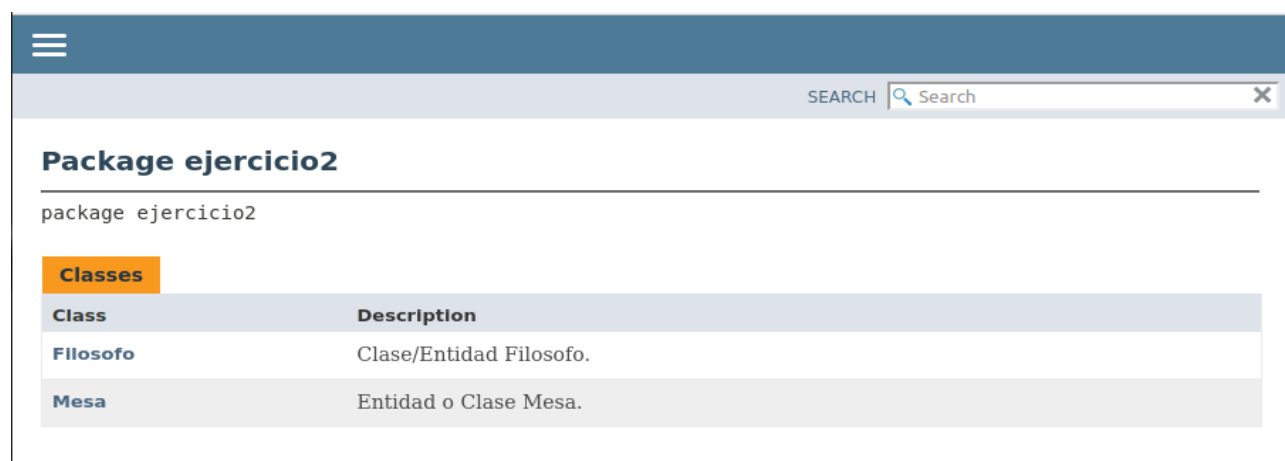
Primeramente, tenemos la página principal donde se indexan los diferentes paquetes de nuestra aplicación:



The screenshot shows a web application interface. At the top, there is a dark blue header with a hamburger menu icon on the left and a search bar on the right labeled 'SEARCH' with a magnifying glass icon and a close button. Below the header, the main content area has a title 'PSP02_Ej2 1.0-SNAPSHOT API'. Underneath the title is a tab labeled 'Packages' in an orange box. Below the tab is a table with two columns: 'Package' and 'Description'. The table contains two rows: one for 'com.kevinzamora.psp02_ej2' and another for 'ejercicio2'.

Package	Description
com.kevinzamora.psp02_ej2	
ejercicio2	

Seguidamente, nos aparece una página donde se nos muestran las diferentes clases que están contenidas en cada paquete; a esta accedemos haciendo clic sobre el paquete correspondiente:



The screenshot shows the same web application interface as before, but now it displays the details for the 'ejercicio2' package. The title is 'Package ejercicio2'. Below the title, the text 'package ejercicio2' is shown. There is a tab labeled 'Classes' in an orange box. Below the tab is a table with two columns: 'Class' and 'Description'. The table contains two rows: one for 'Filosofo' with the description 'Clase/Entidad Filosofo.' and another for 'Mesa' with the description 'Entidad o Clase Mesa.'.

Class	Description
Filosofo	Clase/Entidad Filosofo.
Mesa	Entidad o Clase Mesa.

Desde esta 'vista', podemos inspeccionar las diferentes clases de cada paquete y, al acceder a la clase seleccionada, nos aparecerá una 'vista en detalle' de todas sus propiedades, métodos y otros aspectos de interés. En la siguiente página se muestra una captura de la sección en cuestión:

Página con los detalles principales de la clase filósofo:

OVERVIEWPACKAGECLASSUSE TREEINDEXHELP

SUMMARY: NESTED | FIELD | CONSTR | METHODDETAIL: FIELD | CONSTR | METHOD

SEARCH

Packageejercicio2

ClassFilosofo

java.lang.Object[®]
java.lang.Thread[®]
ejercicio2.Filosofo

All Implemented Interfaces:
Runnable[®]

public class Filosofo
extends Thread[®]

Clase/Entidad Filosofo. Extiende la clase "Thread"

Author:
kzdesigner

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Thread[®]

Thread.Builder[®], Thread.State[®], Thread.UncaughtExceptionHandler[®]

Field Summary

Fields inherited from class java.lang.Thread[®]

MAX_PRIORITY[®], MIN_PRIORITY[®], NORM_PRIORITY[®]

Constructor Summary

Constructors

Constructor	Description
Filosofo(Mesa mesa, int comensal)	Constructor de la clase Filósofo

Method Summary

All MethodsInstance MethodsConcrete Methods

Modifier and Type	Method	Description
void	comiendo()	Método comiendo() para mostrar que un Filósofo está comiendo y definir una pausa/interrupción determinada.
void	pensando()	Método pensando() para mostrar que un Filósofo está pensando y definir una pausa/interrupción determinada.
void	run()	Método 'run()' para ejecutar el programa principal de nuestra aplicación.

Methods inherited from class java.lang.Thread[®]

activeCount[®], checkAccess[®], clone[®], countStackFrames[®], currentThread[®], dumpStack[®], enumerate[®], getAllStackTraces[®], getContextClassLoader[®], getDefaultUncaughtExceptionHandler[®], getId[®], getName[®], getPriority[®], getStackTrace[®], getState[®], getThreadGroup[®], getUncaughtExceptionHandler[®], holdsLock[®], interrupt[®], interrupted[®], isAlive[®], isDaemon[®], isInterrupted[®], isVirtual[®], join[®], join[®], join[®], join[®], ofPlatform[®], ofVirtual[®], onSpinWait[®], resume[®], setContextClassLoader[®], setDaemon[®], setDefaultUncaughtExceptionHandler[®], setName[®], setPriority[®], setUncaughtExceptionHandler[®], sleep[®], sleep[®], sleep[®], start[®], startVirtualThread[®], stop[®], suspend[®], threadId[®], toString[®], yield[®]

Methods inherited from class java.lang.Object[®]

equals[®], finalize[®], getClass[®], hashCode[®], notify[®], notifyAll[®], wait[®], wait[®], wait[®]

Constructor Details

Filosofo

public Filosofo(Mesa mesa,
int comensal)

Constructor de la clase Filósofo

Parameters:
mesa - Mesa donde se sientan los filósofos
comensal - Comensales que se sientan/están sentado en la mesa

Method Details

run

public void run()

Método 'run()' para ejecutar el programa principal de nuestra aplicación. Su funcionamiento interno se encuentra explicado en diferentes comentarios. Retorna 'void'

Specified by:
run[®] in interface Runnable[®]

Overrides:
run[®] in class Thread[®]

pensando

public void pensando()
throws InterruptedException[®]

Método pensando() para mostrar que un Filósofo está pensando y definir una pausa/interrupción determinada. Retorna 'void'

Throws:
InterruptedException[®] - Captura de las posibles interrupciones

comiendo

public void comiendo()
throws InterruptedException[®]

Método comiendo() para mostrar que un Filósofo está comiendo y definir una pausa/interrupción determinada. Retorna 'void'

Throws:
InterruptedException[®] - Captura de los posibles errores de ejecución

Y finalmente, una vez posicionados en la página anterior, también tendremos acceso a las secciones: **‘uso’** (si está usada o no), **‘árbol/tree’** (donde se indexa la estructura de directorios/carpetas), **‘index/índice’** (donde se muestra qué es cada objeto/método)’ y finalmente **‘ayuda’** (donde se muestra la sección de ayuda).

Página ‘uso’:

OVERVIEWPACKAGECLASSUSETREEINDEXHELP

SEARCH

Uses of Class

ejercicio2.Filosofo

No usage of ejercicio2.Filosofo

Página ‘árbol’:

OVERVIEWPACKAGECLASSUSETREETREEINDEXHELP

SEARCH

Hierarchy For Package ejercicio2

Package Hierarchies:
All Packages

Class Hierarchy

- java.lang.Object
 - ejercicio2.Mesa
 - java.lang.Thread (implements java.lang.Runnable)
 - ejercicio2.Filosofo

Página ‘Índice’:

OVERVIEWPACKAGECLASSUSETREETREETREEINDEXHELP

SEARCH

Index

C D E F M P R

All Classes and Interfaces | All Packages

C

- cogerPalillos(int) - Method in class ejercicio2.Mesa
 - Método para coger palillos.
 - com.kevinzamora.psp02_ej2 - package com.kevinzamora.psp02_ej2
- comiendo() - Method in class ejercicio2.Filosofo
 - Método comiendo() para mostrar que un Filósofo está comiendo y definir una pausa/interrupción determinada.

D

- dejarPalillos(int) - Method in class ejercicio2.Mesa
 - Método para coger palillos.

E

- ejercicio2 - package ejercicio2

F

- Filosofo - Class in ejercicio2
 - Clase/Entidad Filósofo
- Filosofo(Mesa, int) - Constructor for class ejercicio2.Filosofo
 - Constructor de la clase Filósofo

M

- main(String[]) - Static method in class com.kevinzamora.psp02_ej2.Principal
 - Método principal dedicado a inicializar y ejecutar nuestra aplicación
- Mesa - Class in ejercicio2
 - Entidad o Clase Mesa.
- Mesa(int) - Constructor for class ejercicio2.Mesa
 - Constructor de la clase Mesa

P

- palilloDerecho(int) - Method in class ejercicio2.Mesa
 - Método para calcular y definir el palillo derecho.
- palilloIzquierdo(int) - Method in class ejercicio2.Mesa
 - Método para definir el valor del palillo izquierdo.
- pensando() - Method in class ejercicio2.Filosofo
 - Método pensando() para mostrar que un Filósofo está pensando y definir una pausa/interrupción determinada.
- Principal - Class in com.kevinzamora.psp02_ej2
 - Clase Principal de nuestra aplicación.
- Principal() - Constructor for class com.kevinzamora.psp02_ej2.Principal

R

- run() - Method in class ejercicio2.Filosofo
 - Método 'run()' para ejecutar el programa principal de nuestra aplicación.

C D E F M P R

All Classes and Interfaces | All Packages

Página ‘Ayuda’:

OVERVIEWPACKAGECLASSUSETREETREETREEINDEXHELP

HELP: NAVIGATION | PAGES

SEARCH

JavaDoc Help

- Navigation:
 - Search
- Kinds of Pages:
 - Overview
 - Package
 - Class or Interface
 - Other Files
 - Use
 - Tree (Class Hierarchy)
 - All Packages
 - All Classes and Interfaces
 - Index

Navigation

Starting from the Overview page, you can browse the documentation using the links in each page, and in the navigation bar at the top of each page. The Index and Search box allow you to navigate to specific declarations and summary pages, including: All Packages, All Classes and Interfaces

Search

You can search for definitions of modules, packages, types, fields, methods, system properties and other terms defined in the API. These items can be searched using part or all of the name, optionally using "camelCase" abbreviations, or multiple search terms separated by whitespace. Some examples:

- "j.l.obj" matches "java.lang.Object"
- "InpStr" matches "java.io.InputStream"
- "math exact long" matches "java.lang.Math.absExact(long)"

Refer to the Javadoc Search Specification* for a full description of search features.

Kinds of Pages

The following sections describe the different kinds of pages in this collection.

Overview

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. These pages may contain the following categories:

- Interfaces
- Classes
- Enum Classes
- Exception Classes
- Annotation Interfaces