**Programación de servicios y procesos**

**Tarea Unidad 1**

Contenido

[Actividad 1. 2](#_Toc180498213)

[Actividad 2. 4](#_Toc180498214)

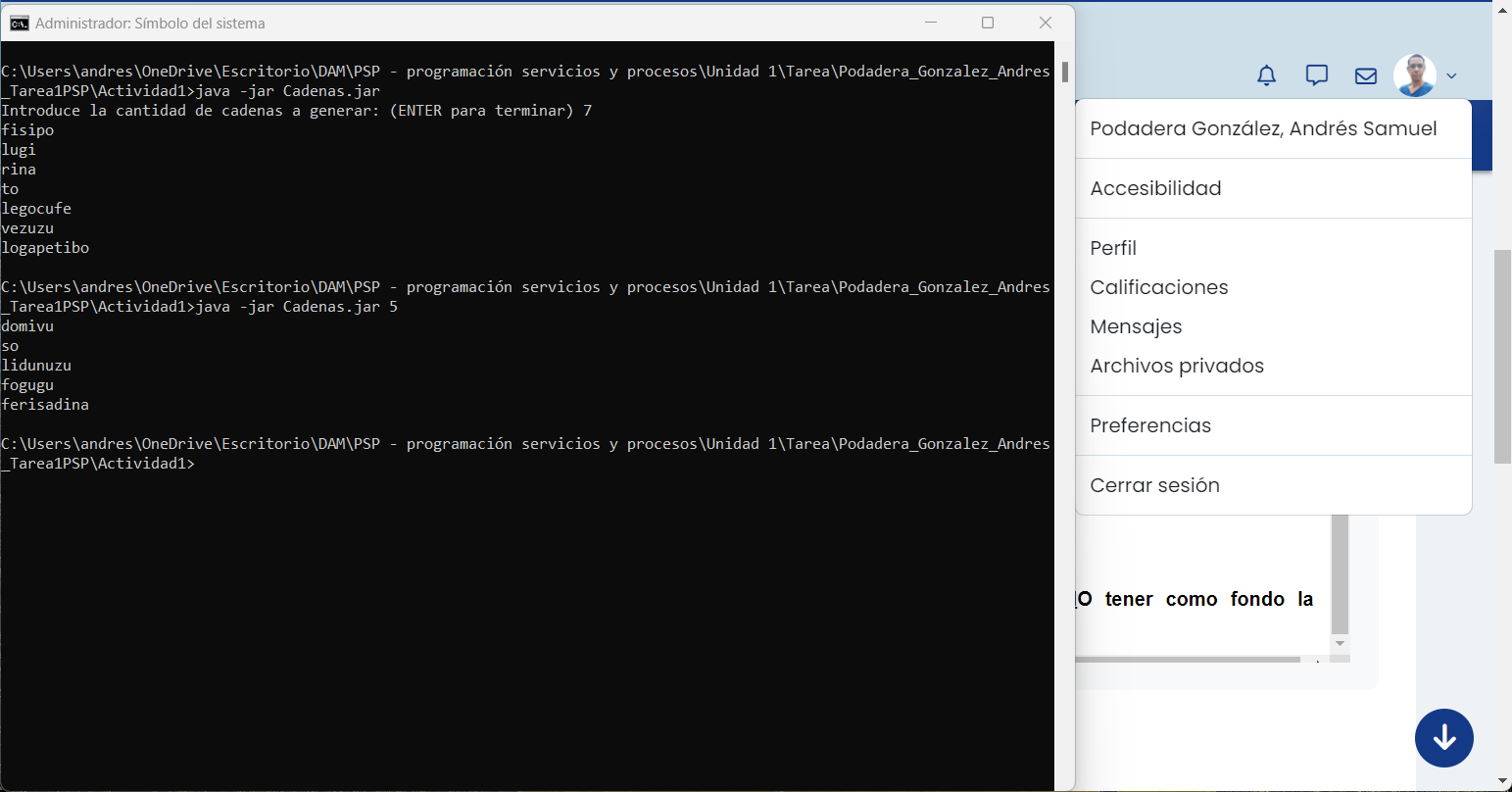
# Actividad 1.

En primer lugar, quiero aclarar que hay varias formas de ejecutar las pruebas sobre nuestros programas, una sería usar los archivos .java, una vez compilados y otra la opción de los jars que será la que usaré en este caso. Uso esta opción porque me parece la opción más portable y la que evitaría tener que tocar el programa según desde donde los estemos utilizando. Para la realización de las pruebas obtengo los archivos .jar de cada programa, pulsando la opción “Clean and Build” de netbeans y moviendo los archivos generados a una carpeta diferente desde donde realizaremos las pruebas.

Las pruebas van a consistir en varias ejecuciones de los diferentes programas para terminar con la ejecución de los dos uniendo la salida de uno con la entrada del otro. Empezamos con la ejecución de los comandos correspondientes en un CMD de Windows abierta en el directorio donde se encuentran los archivos .jar de nuestros programas:

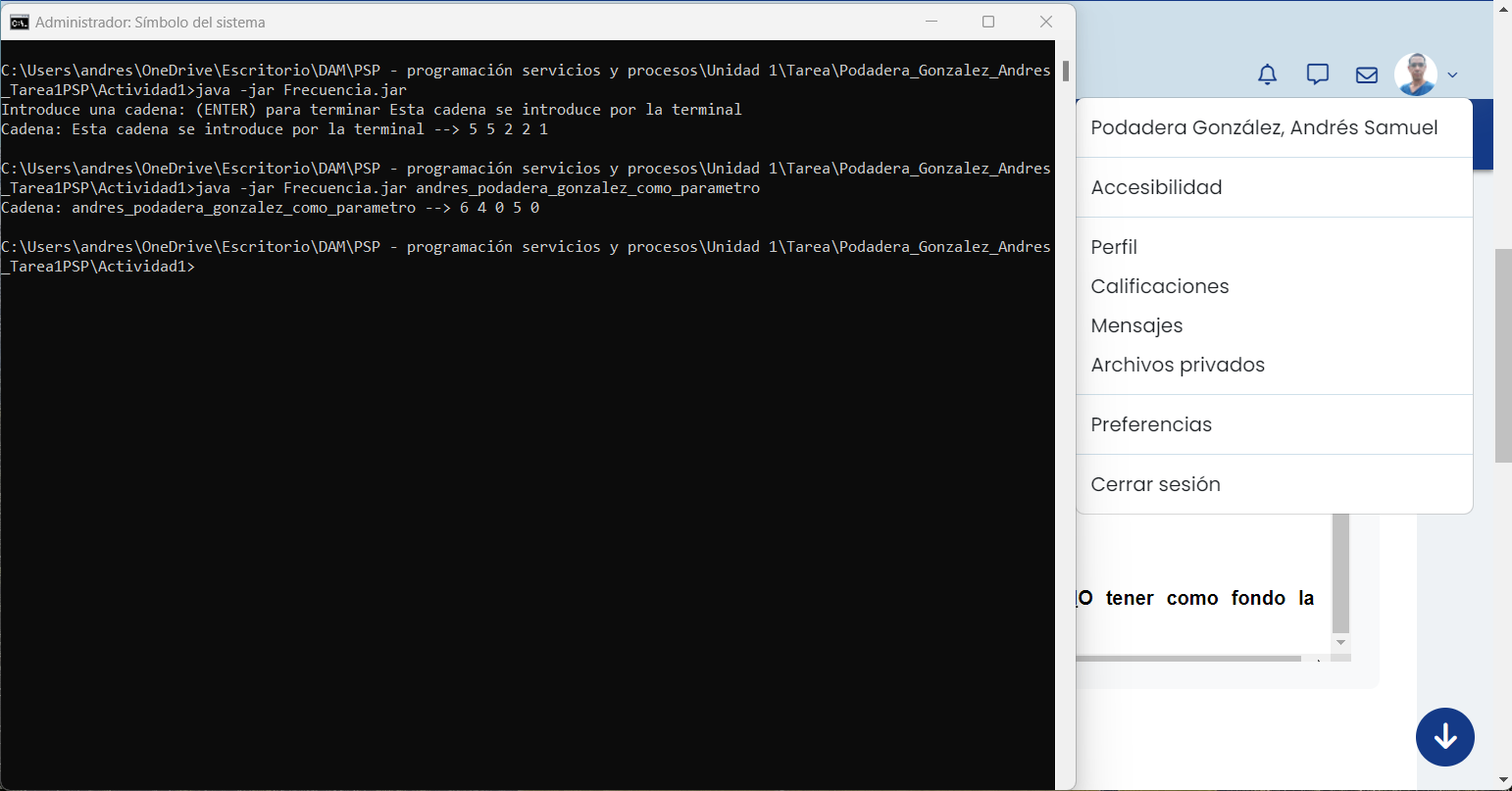
**java -jar Cadenas.jar**

**java -jar Cadenas.jar 5**

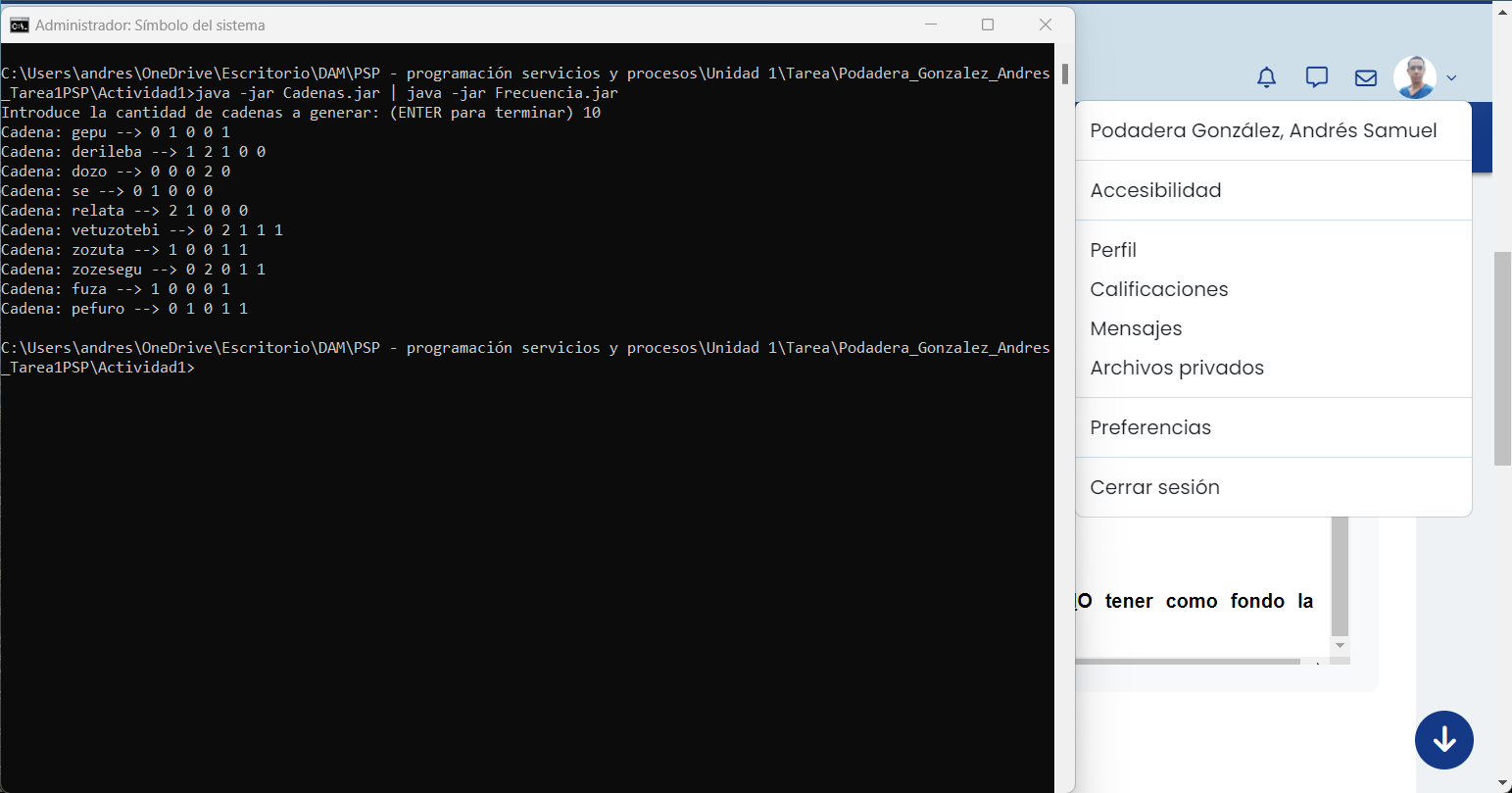
****

**java -jar Frecuencia.jar**

**java -jar Frecuencia.jar esta es una cadena de prueba**

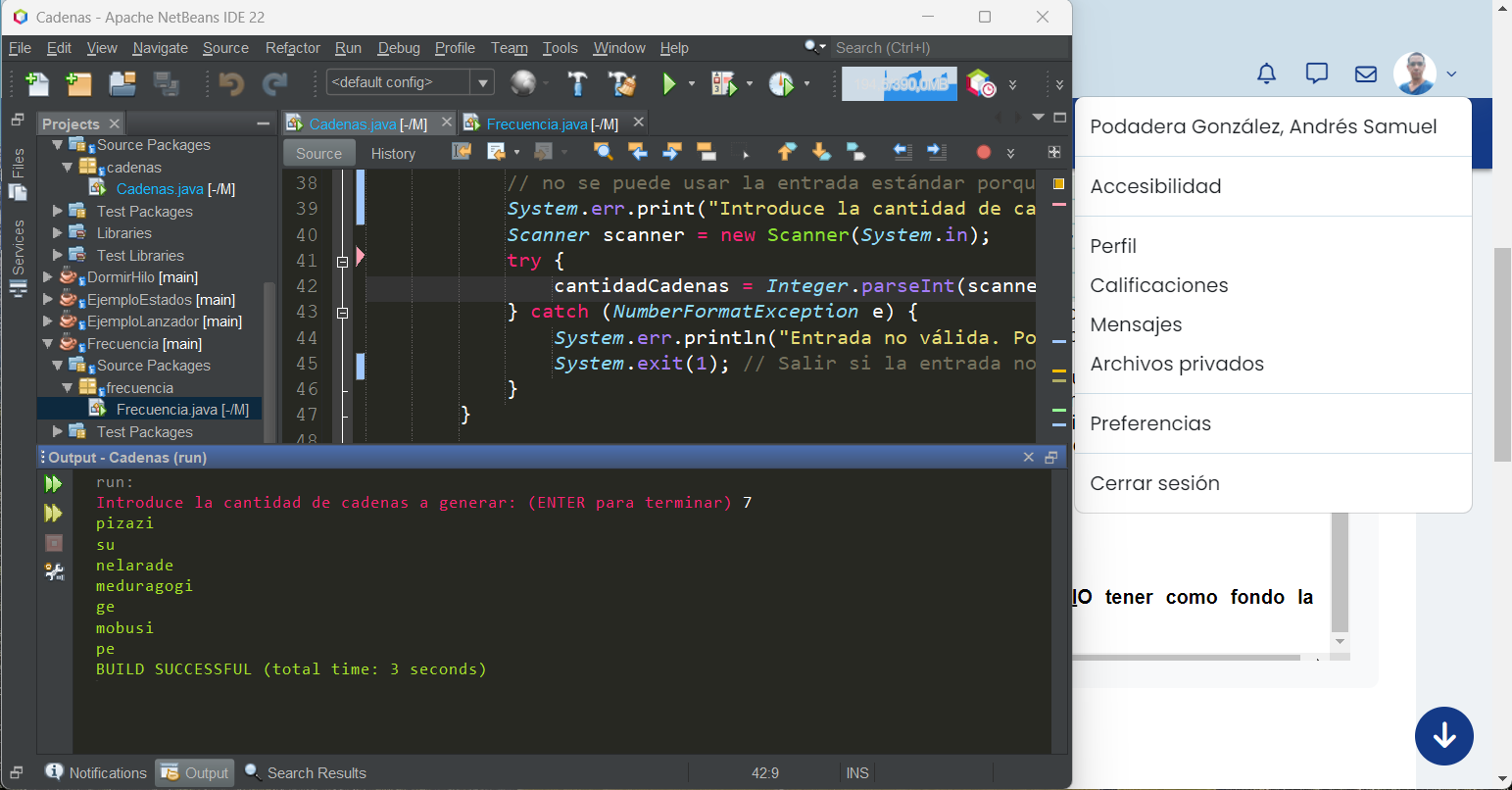
****

**java -jar Cadenas.jar | java -jar Frecuencia.jar**

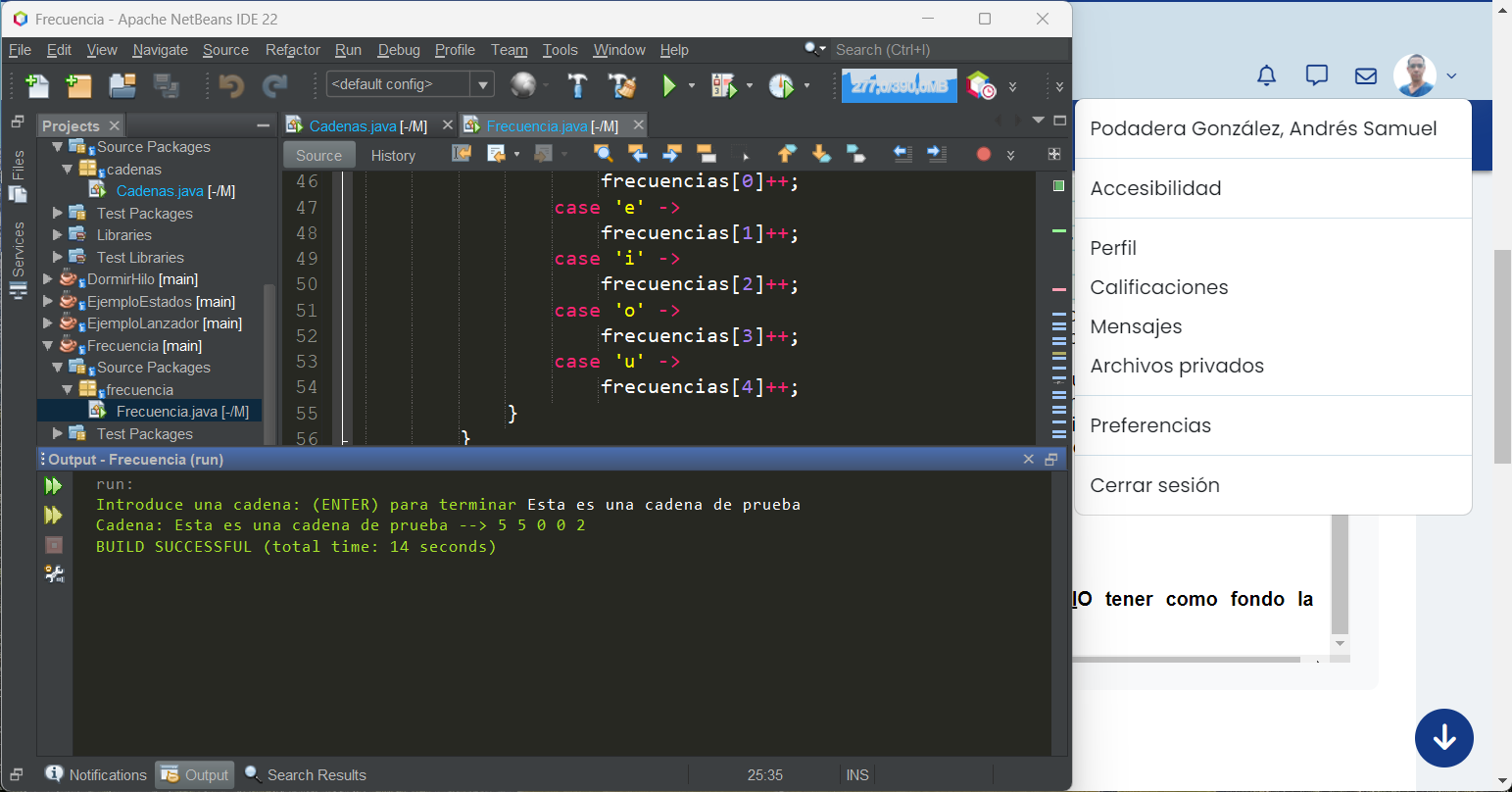
****

**Desde Netbeans:**

* **Cadenas**

****

* **Frecuencia**



Las conclusiones a las que podemos llegar es que nuestros programas se adaptan dependiendo desde donde sean llamados o ejecutados y que si usamos un pipe | podemos conectar la salida del primero con la entrada del segundo.

# Actividad 2.

En esta actividad dos se nos pedía que implementáramos un programa que usara hilos para gestionar un array de números. Nuestro elemento compartido será el array de números y para ello he creado una clase llamada Buffer, simulando el comportamiento de un programa real en el que la entrada de datos es un buffer compartido. En esta clase tendremos principalmente el array de números, variables para controlar cuando el array esta lleno y cuando este vacío, además de la sumatoria que acumula todos los números que se van consumiendo. También tendremos toda la lógica pertinente para consumir y producir números en/del array.

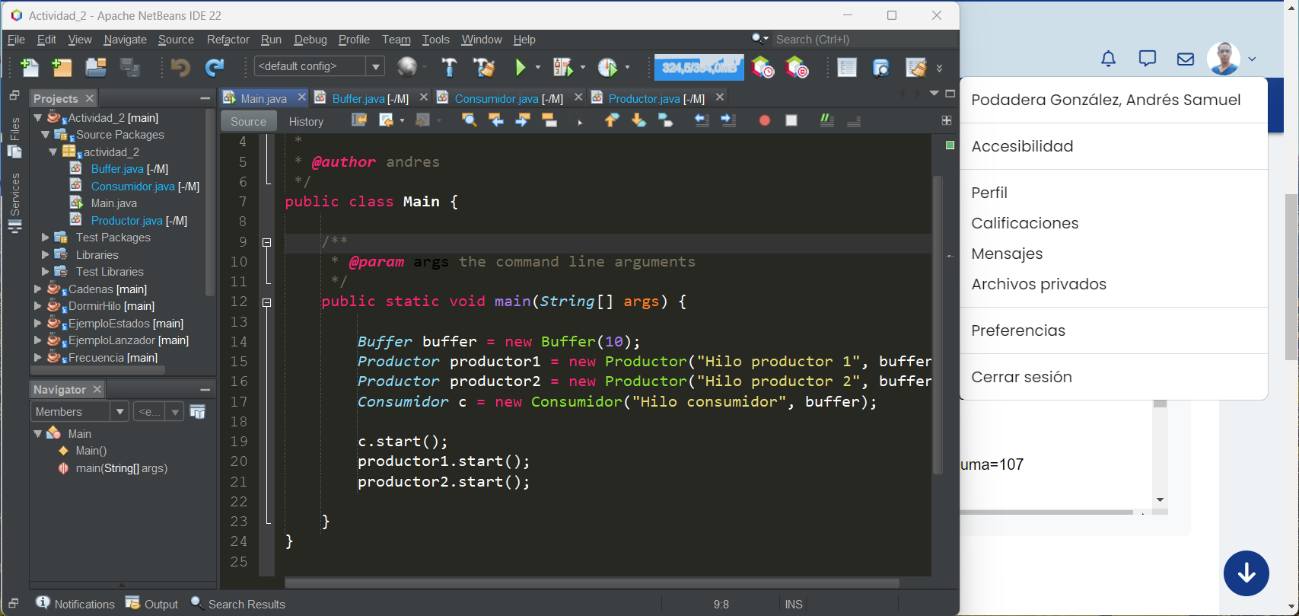
Además, esta implementación usa una clase Consumidor y otra Productor. Ambas clases almacenan un objeto Buffer común, mediante el método constructor al momento de crearse los respectivos objetos.

La clase Consumidor será la encargada de crear el hilo consumidor y la clase Productor será la encargada de crear nuestros dos hilos productores.

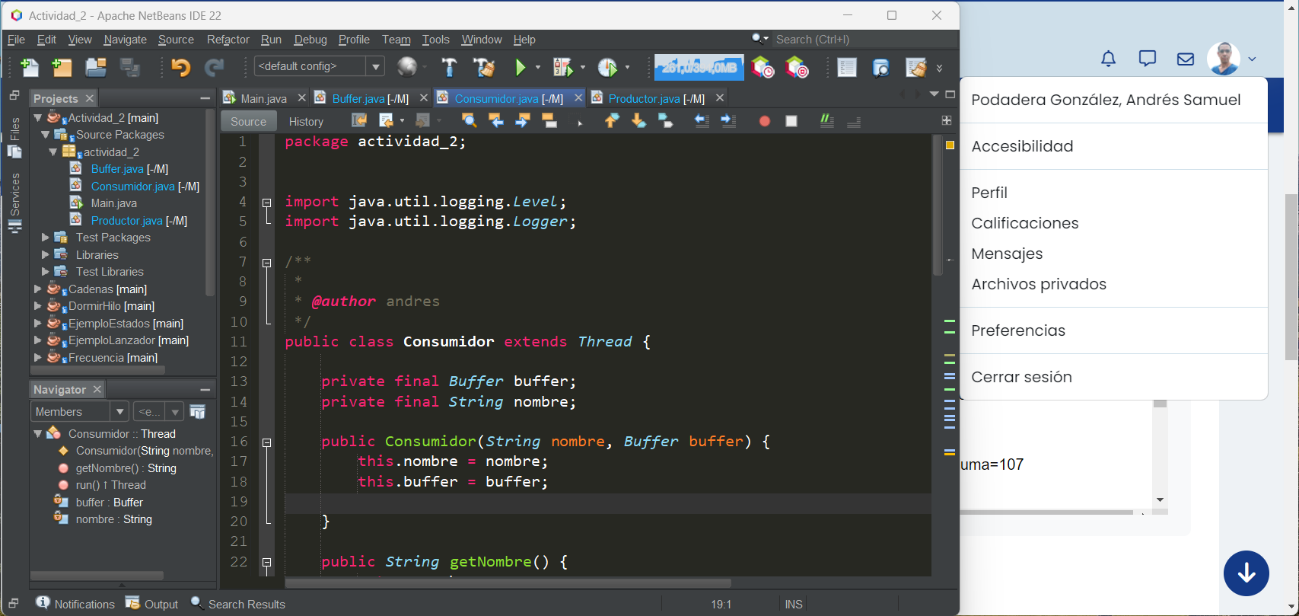
Todo esto se materializa en la clase Main, una clase principal desde la que iniciará nuestro programa y donde se crearán objetos e hilos que usaremos en la tarea.

Adjunto varias capturas del programa para demostrar mi autoría y añado comentarios al código lo suficientemente precisos y extensos para que se entienda lo que hace.

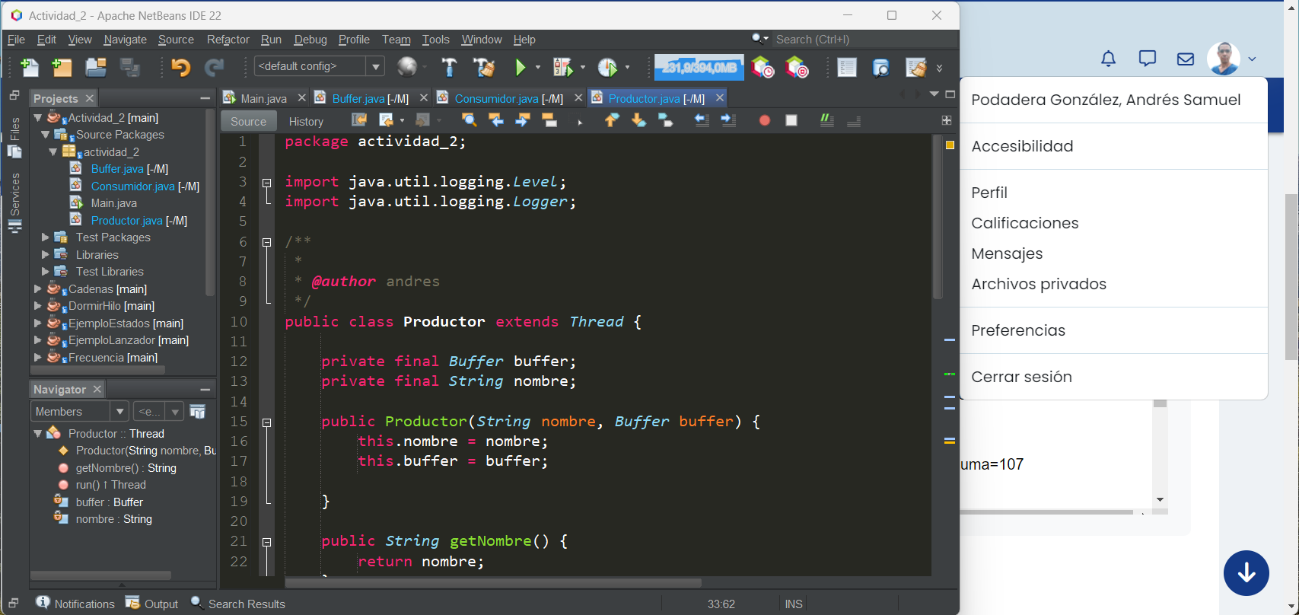
* **Main:**

****

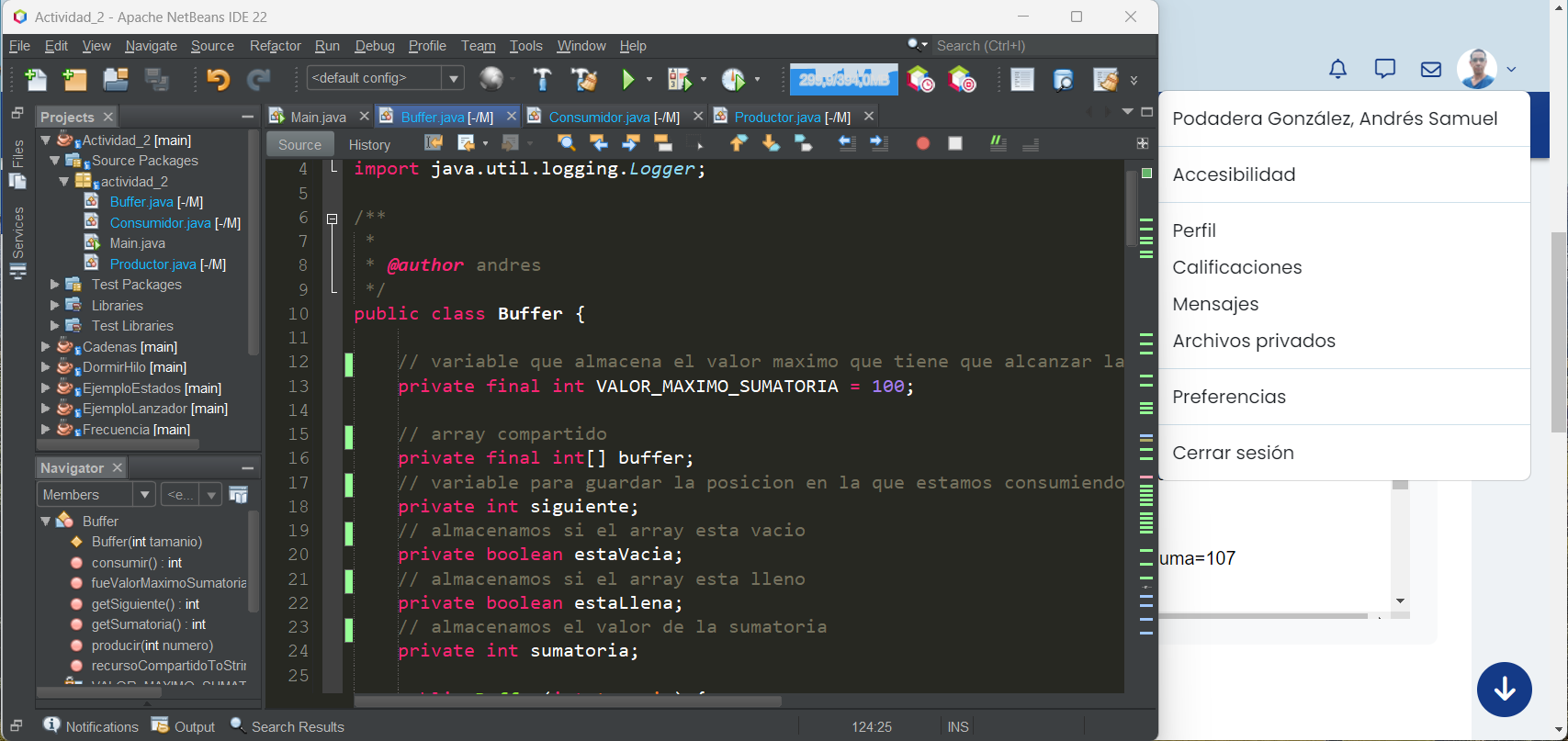
* **Consumidor**

****

* **Productor**

****

* **Buffer**

****

La información de salida del programa se almacena en un archivo llamado **logs.txt** dentro del proyecto.

