

I.E.S Aguadulce

Titulación: Desarrollo de Aplicaciones Multiplataforma

Asignatura: Programación de servicios y procesos



En



Tarea número 2

– Programación multihilo –

Autor:	David Jiménez Riscardo
Teléfono:	618882196
E-Mail:	david.jimenez.riscardo@gmail.com

Dos Hermanas, 5 de Diciembre del año 2021

Table of Contents

1	COLA DE ELEMENTOS.....	III
2	PRODUCTORES DE ELEMENTOS.....	IV
3	CONSUMIDORES DE ELEMENTOS.....	V
4	PROGRAMA PRINCIPAL.....	VI
5	CASOS DE PRUEBA.....	VII
5.1	CASO DE PRUEBA 1: UN PRODUCTOR Y DOS CONSUMIDORES.....	VII
5.2	CASO DE PRUEBA 2: UN PRODUCTOR Y TRES CONSUMIDORES.....	VIII
5.3	CASO DE PRUEBA 3: DOS PRODUCTORES Y DOS CONSUMIDORES.....	IX
5.4	CASO DE PRUEBA 4: DOS PRODUCTORES.....	X
5.5	CASO DE PRUEBA 5: UN PRODUCTOR Y UN CONSUMIDOR.....	XI
5.6	CASO DE PRUEBA 6: DOS CONSUMIDORES.....	XII
6	MODO DEPURACIÓN Y MODO SILENCIOSO.....	XII
7	EVALUACIÓN DE LA TAREA.....	XV

Se tendrá que llevar a cabo la implementación del modelo productor-consumidor utilizando hilos en Java. Para ello se tendrá que implementar y hacer funcionar las cuatro clases que se describen en los siguientes apartados así como rellenar la Ficha de la tarea donde explicar el funcionamiento de cada componente y analizar algunos escenarios específicos de prueba. También se proporciona el proyecto Netbeans sobre el que se tiene que empezar a trabajar para implementar las clases Java que se piden:

1 Cola de elementos

En primer lugar se deberá desarrollar una estructura de datos que pueda almacenar los elementos que van creando los productores y de donde podrán ir tomando los consumidores.

Será una clase llamada **ColaElementos** que contendrá en su interior un objeto de tipo **Queue** (que implemente la interfaz **Queue**). Para simplificar, el contenido de esa cola serán objetos de tipo carácter (objetos **Character**), es decir será una **Queue<Character>**. Los elementos van a ser representados por caracteres, normalmente letras mayúsculas.

Esta clase dispondrá de un único constructor, sin parámetros, que instanciará su único atributo (tal como se ha indicado, será un objeto de tipo cola de caracteres).

Esta clase dispondrá de los siguientes métodos:

1. Método para añadir un nuevo elemento al final de la cola: **void addElemento(char elemento)**.
2. Método para tomar un elemento del principio de la cola: **char getElemento()**.
3. Método para obtener el tamaño de la cola: **int size()**.
4. Método para obtener una representación textual de la cola: **String toString()**.

Se debe tener en cuenta que un objeto de esta clase es un componente muy delicado de la aplicación. Se trata de un elemento compartido tanto por los productores como por los consumidores. Cuando un hilo esté ejecutando alguno de estos métodos para añadir, eliminar o consultar elementos, no debería poder ejecutarse al mismo tiempo que otro, pues se podrían obtener resultados que no tuvieran sentido, por ejemplo dos consumidores que accedan y consuman el mismo elemento. ¿Cómo lo harías?

2 Productores de elementos

Se va a necesitar disponer de hilos productores de elementos que se dediquen a producir elementos (caracteres) y los vaya añadiendo a la cola de elementos disponibles para que los consumidores puedan usarlos. Para ello se tendrá que implementar una clase **ProductorElementos** con un método **run** que vaya generando letras mayúsculas a partir de una letra inicial y las vaya añadiendo a la cola de elementos disponibles.

Este productor de elementos dispondrá de un único constructor con los siguientes parámetros:

- nombre del productor ("P1", "P2", "P3", etc.), de tipo **String**.
- cola de elementos donde ir colocando cada elemento que se genere, de tipo **ColaElementos** (será el recurso compartido usado por todos los productores y todos los consumidores);
- tiempo que se tarda en producir cada elemento (en milisegundos), de tipo **int**. Es el tiempo que hay que esperar desde que se genera un elemento hasta que se genera otro. Nos sirve para simular el proceso de producción real;
- cantidad de elementos, también de tipo entero, que va a producir ese hilo. Cuando se generen todos sus elementos, finalizará su ejecución;
- modo de funcionamiento (0 para depurar y otro número para modo "silencioso"). Se verá más adelante para qué sirve este parámetro.

El bloque de ejecución de este hilo (método **run**) irá produciendo elementos (letras mayúsculas) y los irá colocando en la cola de elementos común que se le proporcionó a través del constructor. El primer elemento que generará será una 'A' y si tiene que generar más elementos más allá de la 'Z' comenzaría de nuevo por la 'A'. Entre la producción de un elemento y el siguiente debe transcurrir el tiempo de espera que se especificó en el constructor.

Además, cada productor debe disponer de alguna estructura de datos interna que almacene un registro de todos los elementos que ha producido.

Cuando finalice su ejecución se mostrará el siguiente mensaje en pantalla, siendo los elementos disponibles los que no se hayan consumido aún:

```
Fin del productor <nombre>. Elementos producidos: <lista de elementos producidos por este productor>
```

```
Elementos disponibles en la cola: <contenido de la cola de elementos disponibles en ese momento>
```

Por ejemplo:

```
Fin del productor P1. Elementos producidos: [A, B, C, D, E].  
Elementos disponibles en la cola: [C, D, E]
```

3 Consumidores de elementos

Adicionalmente se necesita disponer de hilos consumidores de elementos que se dediquen en este caso a recoger los elementos (caracteres) de una cola de elementos disponibles y los vayan consumiendo o usando. Para ello habrá que implementar una clase **ConsumidorElementos** con un método **run** que vaya obteniendo los elementos de la cola (caracteres) y los "gaste", añadiéndolos a una estructura de datos propia donde se registran los productos usados por este consumidor.

Este consumidor de elementos dispondrá de un constructor con los siguientes parámetros:

- nombre del consumidor ("C1", "C2", "C3", etc.), de tipo **String**;
- cola de elementos de donde ir obteniendo los elementos para consumir, de tipo **ColaElementos**;
- tiempo que se tarda en consumir cada elemento (en milisegundos), de tipo **int**. Es el tiempo que hay que esperar entre que se consume un elemento y otro. Nos sirve para simular el proceso de consumo real;
- cantidad de elementos, también de tipo entero, que necesita a consumir este hilo. Cuando consuma todos elementos que necesite, finalizará su ejecución;
- modo de funcionamiento (0 para depurar y otro número para modo "silencioso").

El bloque de ejecución de este hilo (método **run**) irá obteniendo elementos (letras) de la cola y los "consumirá". Debe ir almacenando en alguna estructura de datos interna los elementos que va consumiendo para poder consultar ese registro en cualquier momento. Entre el consumo de un elemento y otro debe transcurrir el tiempo de espera que se especificó en el constructor. Es la manera de simular el proceso de "consumo".

Cuando finalice su ejecución mostrará el siguiente mensaje en pantalla:

```
Elementos consumidos: <lista de elementos consumidos por este consumidor>  
Elementos disponibles en la cola: <contenido de la cola de elementos disponibles en  
ese momento>
```

Por ejemplo:

```
Fin del consumidor C1. Elementos consumidos: [A, B, C, D, E]
Elementos disponibles en la cola: [F, G, H, I]
```

4 Programa principal

Por último, tendrás que implementar un programa principal (clase con método `main`, que será el hilo principal del proceso) desde donde se lancen los hilos productores y consumidores. Este programa tomará la información sobre los hilos que debe lanzar desde un archivo de configuración de hilos llamado `hilos.cfg`, que estará en la misma ubicación del programa.

El archivo `hilos.cfg` será un archivo de texto con la siguiente estructura:

1. una línea por cada hilo que se vaya a lanzar;
2. cada línea tendrá la información de creación del hilo siguiendo la siguiente estructura:
 - tipo de hilo (productor o consumidor);
 - nombre del hilo;
 - tiempo de producción/consumo (dependiendo del tipo de hilo), en milisegundos;
 - cantidad de elementos a producir/consumir (dependiendo del tipo de hilo);
3. cada uno de esos elementos estará separado por la cadena ":" (dos puntos dos puntos).

Por ejemplo, un posible archivo de configuración de hilos podría ser:

```
Productor::P1::20::10
Consumidor::C1::100::5
Consumidor::C2::120::5
```

En este caso se estarían definiendo tres hilos para que la aplicación los lanzara:

1. un primer hilo, de tipo productor y nombre "P1", con un tiempo de producción de 20 milisegundos, que va a generar diez elementos;
2. un segundo hilo, de tipo consumidor y nombre "C1", con un tiempo de consumo de 100 milisegundos, que va a consumir cinco elementos;
3. un tercer hilo, de tipo también consumidor y nombre "C2", con tiempo de consumo de 120 milisegundos, que va a consumir otros cinco elementos.

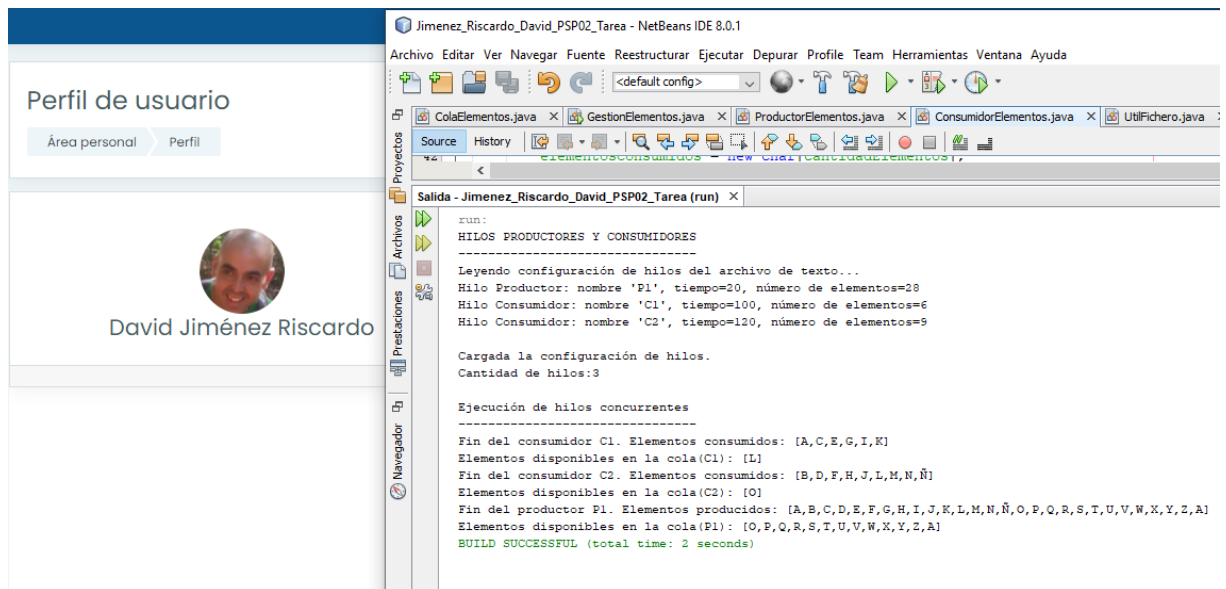
5 Casos de prueba

Una vez que tengamos nuestra aplicación de productores-consumidores funcionando correctamente, es el momento de hacerla trabajar bajo diferentes escenarios. Eso podemos llevarlo a cabo fácilmente modificando el archivo de configuración de hilos.

Parte del trabajo en esta tarea, una vez implementado el programa, será llevar a cabo estas seis pruebas obligatoriamente con los escenarios planteados en el enunciado de la tarea (no pueden inventarse).

1. Una captura de una ejecución de la prueba. Obligatorio tener como fondo la plataforma con el perfil del alumno/a.
2. Una breve descripción y análisis de lo que sucede en cada ejecución: si se consumen todos los elementos producidos, si quedan elementos por consumir y sobran en la cola de disponibles, si faltan elementos para ser consumidos y algún consumidor queda bloqueado, etc.
3. En caso de que esa configuración tenga algún problema, indicar cómo se podría resolver.

5.1 Caso de prueba 1: un productor y dos consumidores.




```
Productor::P1::20::28
Consumidor::C1::100::6
Consumidor::C2::120::9
```

En este caso de prueba podemos apreciar que quedan 13 elementos por consumir ya que se consumen en total 15 de 28 elementos producidos. Esta configuración no presenta ningún problema, el programa se ejecuta correctamente.

5.2 Caso de prueba 2: un productor y tres consumidores

Perfil de usuario

Área personal Perfil



David Jiménez Riscardo

Proyectos

Archivos

Prestaciones

Navegador

Jimenez_Riscardo_David_PSP02_Tarea - NetBeans IDE 8.0.1

Archiv Editi Ve Naveg Fuen Reestruct Ejecut Depur Profi Tear Herramie Venta Ayuc

...ava ProductorElementos.java x ConsumidorElementos.java x UtilFichero.java...

Source History

Salida - Jimenez_Riscardo_David_PSP02_Tarea (run) x

run:
HILOS PRODUCTORES Y CONSUMIDORES

Leyendo configuración de hilos del archivo de texto...
Hilo Productor: nombre 'P1', tiempo=20, número de elementos=14
Hilo Consumidor: nombre 'C1', tiempo=100, número de elementos=5
Hilo Consumidor: nombre 'C2', tiempo=120, número de elementos=7
Hilo Consumidor: nombre 'C3', tiempo=100, número de elementos=5

Cargada la configuración de hilos.
Cantidad de hilos:4

Ejecución de hilos concurrentes

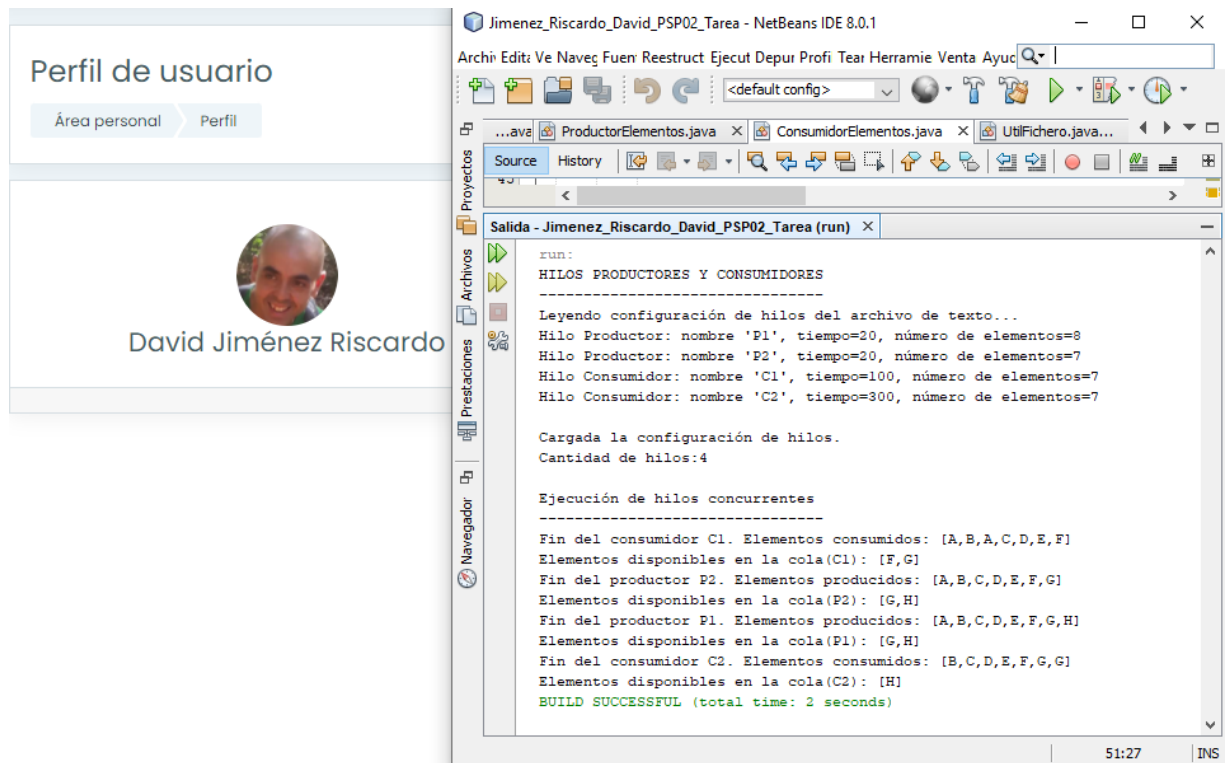
Fin del consumidor C1. Elementos consumidos: [A,D,G,J,M]
Elementos disponibles en la cola(C1): [N]
Fin del productor P1. Elementos producidos: [A,B,C,D,E,F,G,H,I,J,K,L,M,N]
Elementos disponibles en la cola(P1): [N]
Fin del consumidor C3. Elementos consumidos: [B,E,H,K,N]
Elementos disponibles en la cola(C3): [COLA VACIA]

Jimenez_Riscardo_David_PSP02_Tarea (run) running... 51:27 INS

```
Productor::P1::20::14  
Consumidor::C1::100::5  
Consumidor::C2::120::7  
Consumidor::C3::100::5
```

En este caso de prueba observamos que se producen menos elementos de los que van a ser consumidos quedando un consumidor en espera ante la falta de elementos por consumir, por lo tanto el programa no consigue finalizar. Los consumidores C1 y C3 han conseguido consumir sus 5 elementos asignados sin embargo el consumidor C2 sólo ha podido consumir 4 de sus 7 elementos por lo que se queda en espera. Para solventar este problema podríamos llevar un **conteo del número de elementos que se van consumiendo en la cola** de forma que si llegamos a consumir el número total de elementos producidos no dejaremos a ningún hilo en espera.

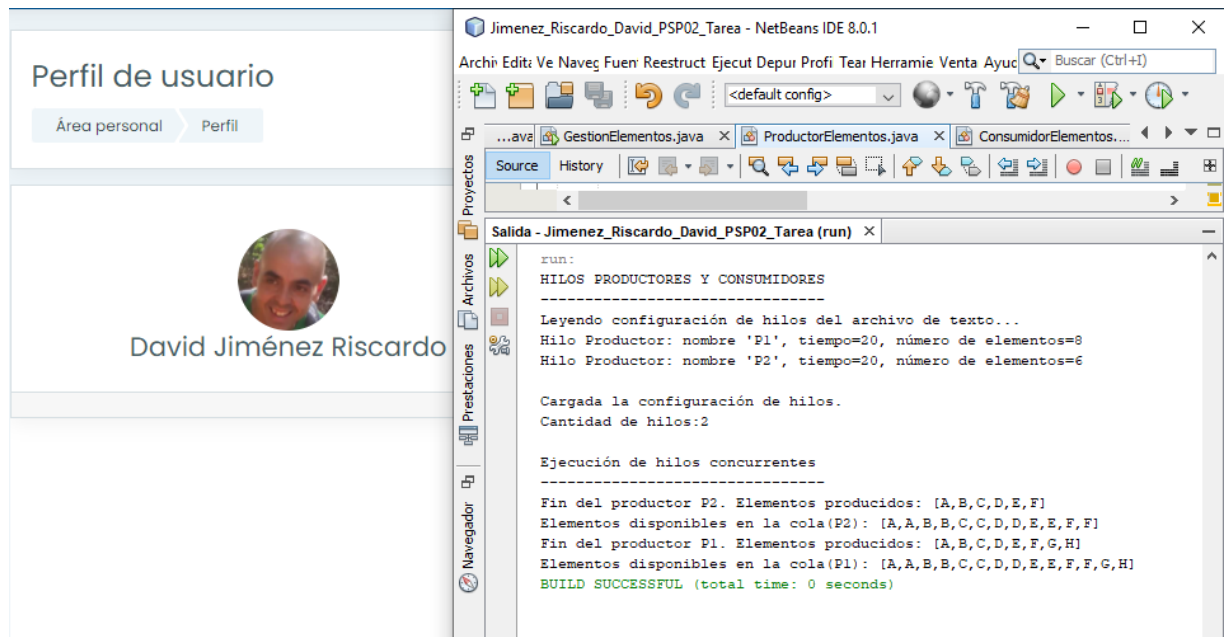
5.3 Caso de prueba 3: dos productores y dos consumidores



```
Productor::P1::20::8
Productor::P2::20::7
Consumidor::C1::100::7
Consumidor::C2::300::7
```

Este caso se ejecuta correctamente, se consumen todos los elementos salvo uno, pues el número total de elementos producidos es mayor al de consumidos en una unidad. Es un poco confuso pues los productores producen elementos iguales. Quizás podríamos hacer una mejora que consistiría en **comprobar antes de agregar el elemento producido a la cola si ya existe** en la misma.

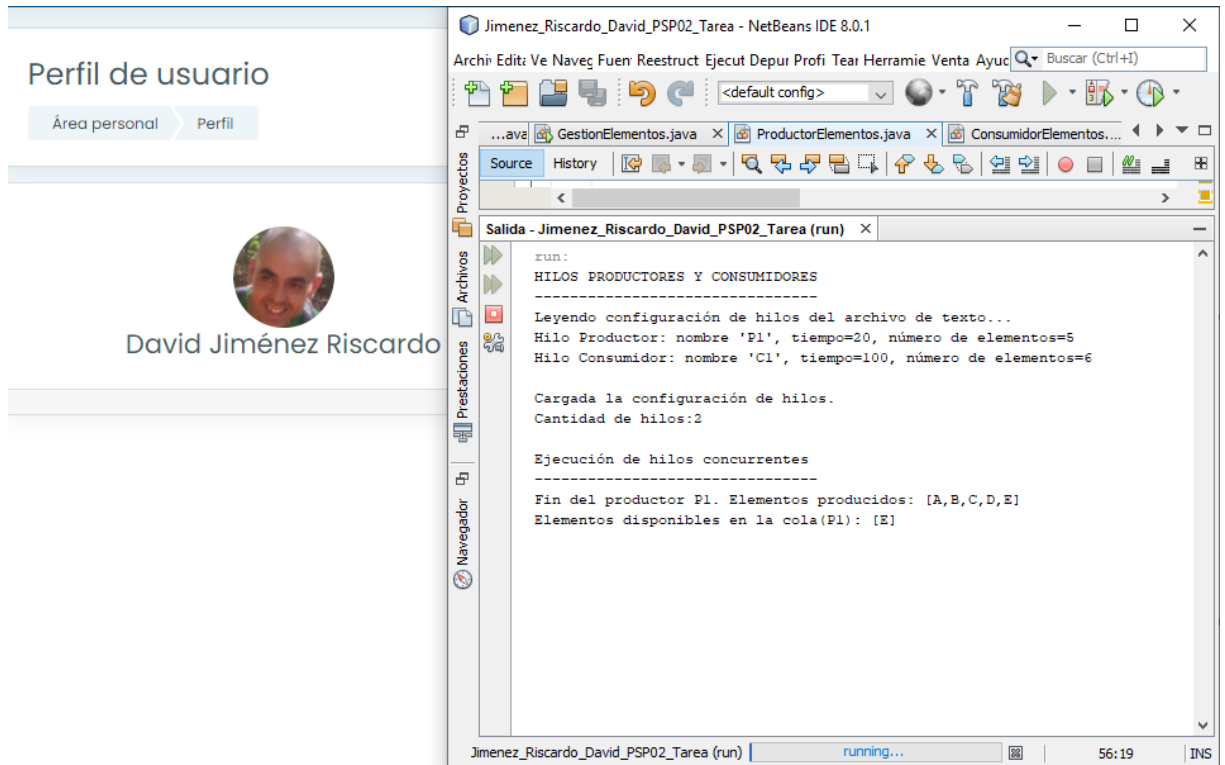
5.4 Caso de prueba 4: dos productores



Productor::P1::20::8
Productor::P2::20::6

Esta prueba no presenta problemas, finaliza correctamente. No existen elementos consumidores por lo que los productores agregan los elementos generados a la cola. Podría mejorarse en el caso que quisieramos que los elementos agregados a la cola sean diferentes haciendo una comprobación previa.

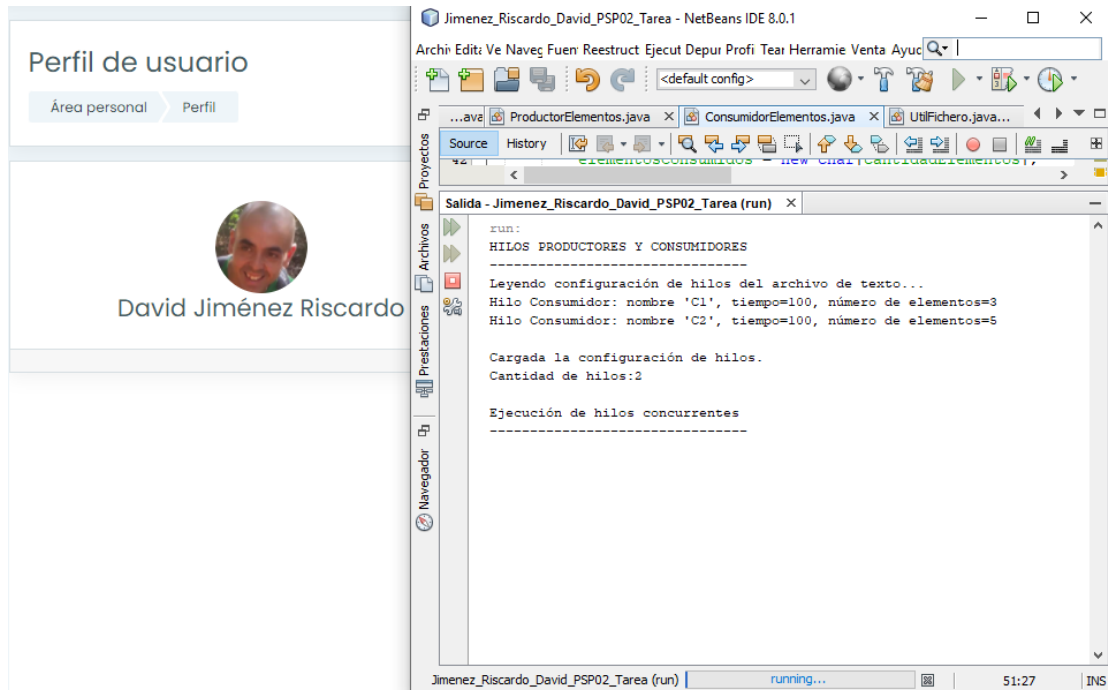
5.5 Caso de prueba 5: un productor y un consumidor



Productor::P1::20::5
Consumidor::C1::100::6

Este caso de prueba si presenta un problema y es que se consumen más elementos de los que se producen, por lo cual el consumidor queda esperando a que se produzca el elemento que le falta por consumir dando lugar a que el programa no finalice. Para solventar este problema haremos como en el caso 5.2, llevaremos un **conteo del número de elementos que se van consumiendo en la cola** de forma que si llegamos a consumir el número total de elementos producidos no dejaremos a ningún hilo en espera.

5.6 Caso de prueba 6: dos consumidores



Consumidor::C1::100::3
Consumidor::C2::100::5

El programa no finaliza porque ambos consumidores quedan en espera pues la cola está vacía al no existir ningún productor que la llena, dando lugar a un problema. Para solventarlo deberíamos de **controlar la existencia de al menos un productor**. En el caso de no existir al menos un productor no lanzaríamos los hilos consumidores.

6 Modo depuración y modo silencioso

Una funcionalidad extra que nos piden añadir a la nuestra gestión de productores y consumidores es disponer una función de "depuración" para poder observar por pantalla el funcionamiento paso a paso de nuestro programa. Para activar esta función de depuración, los constructores de las clases `ProductorElementos` y `ConsumidorElementos` disponen de un parámetro específico de tipo entero. Si ese parámetro es 0, la función de depuración estará activa. Si tiene cualquier otro valor estará desactivada.

Para arrancar el programa principal con la función de depuración habrá que pasarle el valor 0 como parámetro desde la línea de órdenes. Para ello puedes ejecutar tu programa desde la consola añadiéndole un 0 como argumento, o bien desde Netbeans indicándolo en las propiedades del proyecto (sección "Run", parámetro "Arguments" a 0).

Si la función de depuración está activada (valor 0), el hilo productor debe mostrar los siguientes mensajes de depuración en pantalla:

1. al iniciarse el hilo se mostrará un mensaje inicial del tipo "Inicio del productor XXX", donde XXX será el nombre del productor ("P1", "P2", "P3", etc.);
2. cada vez que se produzca un elemento y se añada a la cola de elementos disponibles se mostrarán los siguientes mensajes:
 - "Productor XXX generando y añadiendo elemento YYY", donde XXX será el nombre del productor e YYY será el nombre del elemento que se acaba de producir ("A", "B", "C", etc.);
 - "Elementos disponibles en la cola: ZZZZ", donde ZZZZ será una representación textual de los elementos disponibles en la cola justo en ese momento;
3. cuando el hilo vaya a finalizar su ejecución se mostrarán los siguientes mensajes:
 - "Fin del productor XXX. Elementos producidos: ZZZZ", donde XXX será el nombre del productor y ZZZZ será una representación textual de todos los elementos producidos por este productor;
 - "Elementos disponibles en la cola: ZZZZ", donde ZZZZ será una representación textual de los elementos disponibles en la cola justo en ese momento.

En el caso del hilo consumidor, deben mostrarse la siguiente información de depuración:

1. al iniciarse el hilo se mostrará un mensaje inicial del tipo "Inicio del consumidor XXX", donde XXX será el nombre del consumidor ("C1", "C2", "C3", etc.);
2. cada vez que se intente consumir un elemento:
 - si había elemento en la cola disponible para ser consumido se mostrarán los mensajes:
 1. "Consumidor XXX. Consumiendo elemento YYY. Consumidos hasta ahora: ZZZZ", donde XXX será el nombre del consumidor, YYY será el nombre del elemento que se acaba de consumir ("A", "B", "C", etc.) y ZZZZ será una representación textual de la lista de elementos consumidos hasta ese momento por este consumidor;
 2. "Elementos disponibles en la cola: ZZZZ", donde ZZZZ será una representación textual de los elementos disponibles en la cola justo en ese momento;
 - si por el contrario no había elementos disponibles en la cola, el mensaje será "Consumidor XXX. No hay elementos disponibles" y el consumidor quedará bloqueado en espera de que algún productor genere nuevos elementos y los coloque en la cola.

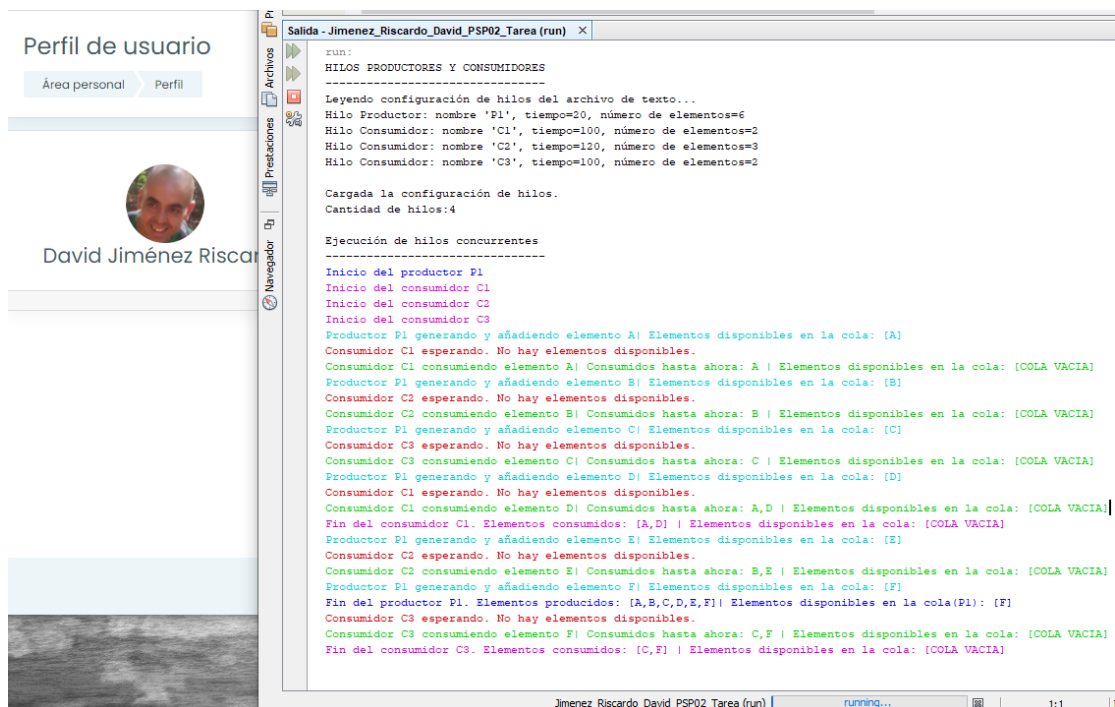
3. cuando el hilo vaya a finalizar su ejecución se mostrarán los siguientes mensajes:
 - "Fin del consumidor XXX. Elementos consumidos: ZZZZ"), donde XXX será el nombre del consumidor y ZZZZ será una representación textual de la lista de todos los elementos consumidos por este productor hasta el momento;
 - "Elementos disponibles en la cola: ZZZZ", donde ZZZZ será una representación textual de los elementos disponibles en la cola justo en ese momento.

Se debe incluir:

1. una captura de una ejecución de la prueba del escenario planteado en el enunciado de la tarea (es OBLIGATORIO que sea ese escenario de configuración de hilos y no otro). OBLIGATORIO tener como fondo de la captura la plataforma con el perfil del alumno/a;

Productor::P1::20::6
 Consumidor::C1::100::2
 Consumidor::C2::120::3
 Consumidor::C3::100::2

2. una breve descripción y análisis de lo que sucede en esa ejecución (inicio y finalización de hilos; bloqueos temporal de hilos cuando se intente consumir y no haya elementos disponibles; reinicio de hilos; etc.).



The screenshot shows a web application interface on the left with a user profile for "David Jiménez Riscador" and a terminal window on the right titled "Salida - Jimenez_Riscador_David_PSP02_Tarea (run)". The terminal displays the following output:

```

run:
HILOS PRODUCTORES Y CONSUMIDORES
-----
Leyendo configuración de hilos del archivo de texto...
Hilo Productor: nombre 'P1', tiempo=20, número de elementos=6
Hilo Consumidor: nombre 'C1', tiempo=100, número de elementos=2
Hilo Consumidor: nombre 'C2', tiempo=120, número de elementos=3
Hilo Consumidor: nombre 'C3', tiempo=100, número de elementos=2

Cargada la configuración de hilos.
Cantidad de hilos:4

Ejecución de hilos concurrentes
-----
Inicio del productor P1
Inicio del consumidor C1
Inicio del consumidor C2
Inicio del consumidor C3
Productor P1 generando y añadiendo elemento A| Elementos disponibles en la cola: [A]
Consumidor C1 esperando. No hay elementos disponibles.
Consumidor C1 consumiendo elemento A| Consumidos hasta ahora: A | Elementos disponibles en la cola: [COLA VACIA]
Productor P1 generando y añadiendo elemento B| Elementos disponibles en la cola: [B]
Consumidor C2 esperando. No hay elementos disponibles.
Consumidor C2 consumiendo elemento B| Consumidos hasta ahora: B | Elementos disponibles en la cola: [COLA VACIA]
Productor P1 generando y añadiendo elemento C| Elementos disponibles en la cola: [C]
Consumidor C3 esperando. No hay elementos disponibles.
Consumidor C3 consumiendo elemento C| Consumidos hasta ahora: C | Elementos disponibles en la cola: [COLA VACIA]
Productor P1 generando y añadiendo elemento D| Elementos disponibles en la cola: [D]
Consumidor C1 esperando. No hay elementos disponibles.
Consumidor C1 consumiendo elemento D| Consumidos hasta ahora: A,D | Elementos disponibles en la cola: [COLA VACIA]
Fin del consumidor C1. Elementos consumidos: [A,D] | Elementos disponibles en la cola: [COLA VACIA]
Productor P1 generando y añadiendo elemento E| Elementos disponibles en la cola: [E]
Consumidor C2 esperando. No hay elementos disponibles.
Consumidor C2 consumiendo elemento E| Consumidos hasta ahora: B,E | Elementos disponibles en la cola: [COLA VACIA]
Productor P1 generando y añadiendo elemento F| Elementos disponibles en la cola: [F]
Fin del productor P1. Elementos producidos: [A,B,C,D,E,F]| Elementos disponibles en la cola(P1): [F]
Consumidor C3 esperando. No hay elementos disponibles.
Consumidor C3 consumiendo elemento F| Consumidos hasta ahora: C,F | Elementos disponibles en la cola: [COLA VACIA]
Fin del consumidor C3. Elementos consumidos: [C,F] | Elementos disponibles en la cola: [COLA VACIA]
  
```

En este caso de ejemplo el productor produce 6 elementos y tenemos 3 consumidores cuyo total de elementos a consumir es de 7 elementos, una unidad más de las producidas. Como puede verse en el ejemplo los consumidores C1 y C3 terminan su

trabajo consumiendo 2 elementos cada uno. El consumidor C2 no llega a finalizar pues le falta un elemento por consumir produciendo esto que el programa no finalice.

He intentado poner toda la información con diversos colores para que pueda seguirse de forma más sencilla la traza del programa.

- Color azul: inicio y fin del productor
- Color cyan: productor produciendo elementos
- Color púrpura: inicio y fin de los consumidores
- Color rojo: hilos consumidores esperando a poder hacer uso de la cola
- Color verde: consumidores consumiendo elementos de la cola

Podemos apreciar también como el productor “tiene mucho trabajo” pues tiene muchos hilos demandando elementos para consumir lo que provoca que continuamente pasen los consumidores a espera. Esto quizás podría mejorarse estableciendo prioridades en la ejecución de los hilos o estableciendo tiempos de espera más largos para que al productor le de tiempo de producir todos los elementos.

7 Evaluación de la tarea

Criterios de evaluación implicados

- a) Se han identificado situaciones en las que resulte útil la utilización de varios hilos en un programa.
- b) Se han reconocido los mecanismos para crear, iniciar y finalizar hilos.
- c) Se han programado aplicaciones que implementen varios hilos.
- d) Se han identificado los posibles estados de ejecución de un hilo y programado aplicaciones que los gestionen.
- e) Se han utilizado mecanismos para compartir información entre varios hilos de un mismo proceso.
- f) Se han desarrollado programas formados por varios hilos sincronizados mediante técnicas específicas.
- g) Se ha establecido y controlado la prioridad de cada uno de los hilos de ejecución.
- h) Se han depurado y documentado los programas desarrollados.

¿Cómo valoramos y puntuamos tu tarea?

Como criterios de corrección:

- Cualquier funcionalidad pedida en el enunciado debe poderse probar y funcionar correctamente, de acuerdo a las especificaciones del enunciado.

- La introducción de datos erróneos o del tipo equivocado no provocará nunca que la aplicación aborte, ya que todas las excepciones serán capturadas y tratadas convenientemente. Se penalizará con hasta 2 puntos menos la no observación de este criterio.
- Se deben incluir comentarios en el código para que éste quede autodocumentado. Especialmente, se valorará que la documentación automática generada con **Javadoc** sea completa y correcta.
- Los mensajes al usuario deben ser siempre claros, correctos ortográfica y gramaticalmente, y ofrecer al mismo toda la información que le resulte relevante de forma completa. Se tendrá especialmente en cuenta que los comentarios tengan presente en todo momento que el usuario no está en el contexto del programador respecto al problema y a la solución, así que debemos preocuparnos de hacer explícito cuál es ese contexto. Por eso, debemos trabajar en la medida de lo posible asumiendo que el usuario no tiene por qué poseer conocimiento alguno sobre el problema, ni sobre la solución que aporta nuestra aplicación, salvo la información que le demos por medio de los mensajes. Si el programa no es fácilmente distribuible se restará hasta 1 punto.
- La corrección ortográfica y gramatical, así como la coherencia en las expresiones, tanto en los comentarios como en los mensajes al usuario. La presencia de este tipo de errores se podrá penalizar con hasta un máximo de 1 punto.
- Es fundamental que el código esté correctamente indentado, se penalizará con hasta 1 punto la no contemplación de este criterio..

<i>CE</i>	<i>Elementos de la rúbrica</i>		<i>Puntuación</i>
RA2.a RA2.h RA1.h	General	RA2.a RA2.h RA1.h. Se sigue la estructura propuesta de declaración de variables, entrada de datos, procesamiento, comunicación de elementos, tal y como se indica en las plantillas y en el enunciado, comentando de forma adecuada los programas desarrollados.	4 puntos
RA1.d	General	RA1.d. Los distintos elementos del proyecto compilan.	2 puntos
RA1.h RA2.h	General	RA1.h RA2.h. El código está correctamente indentado.	1 punto
RA1.a RA1.d	Cola de elementos	RA1.a RA1.d. Se construye la cola de elementos de forma adecuada teniendo en cuenta que varios hilos podrán acceder a ella.	2 puntos
RA1.a RA1.d	Cola de elementos	RA1.a RA1.d. Se construyen los cuatro métodos sobre la cola de elementos: Añadir, Tomar, Obtener tamaño y Obtener representación.	2 puntos
RA2.b RA2.c	Productor de elementos	RA2.b RA2.c. Se construye el productor de elementos.	4 puntos
RA2.b RA2.c	Consumidor de elementos	RA2.b RA2.c. Se construye el consumidor de elementos.	4 puntos
RA2.c RA2.e RA2.f	Programa principal	RA2.c RA2.e RA2.f. El programa principal lee correctamente el fichero de configuración y lanza los hilos consumidor y productor de la manera que se haya indicado en el fichero hilos.cfg	2 puntos
RA2.c RA2.d RA2.e RA2.f RA2.g	Programa principal	RA2.c RA2.d RA2.e RA2.f RA2.g. Se construye el programa principal	4 puntos

RA2.e RA2.f	Casos de prueba	RA2.e RA2.f. Se ejecuta el programa principal recogiendo evidencias de las pruebas tal como se indica en el enunciado, siendo las mismas detalladas y siguiendo las indicaciones dadas en la tarea.	4 puntos
RA2.d RA2.e	Modo depuración y modo silencioso	RA2.d RA2.e. Se implementan los modos opcionales de depuración y el modo silencioso.	1 punto
RA2.a RA2.h	Pruebas modo depuración/silencioso	RA2.a RA2.h. Se prueban los modos de depuración/silencioso recogiendo evidencias de ambos y documentando la nueva funcionalidad.	1 punto