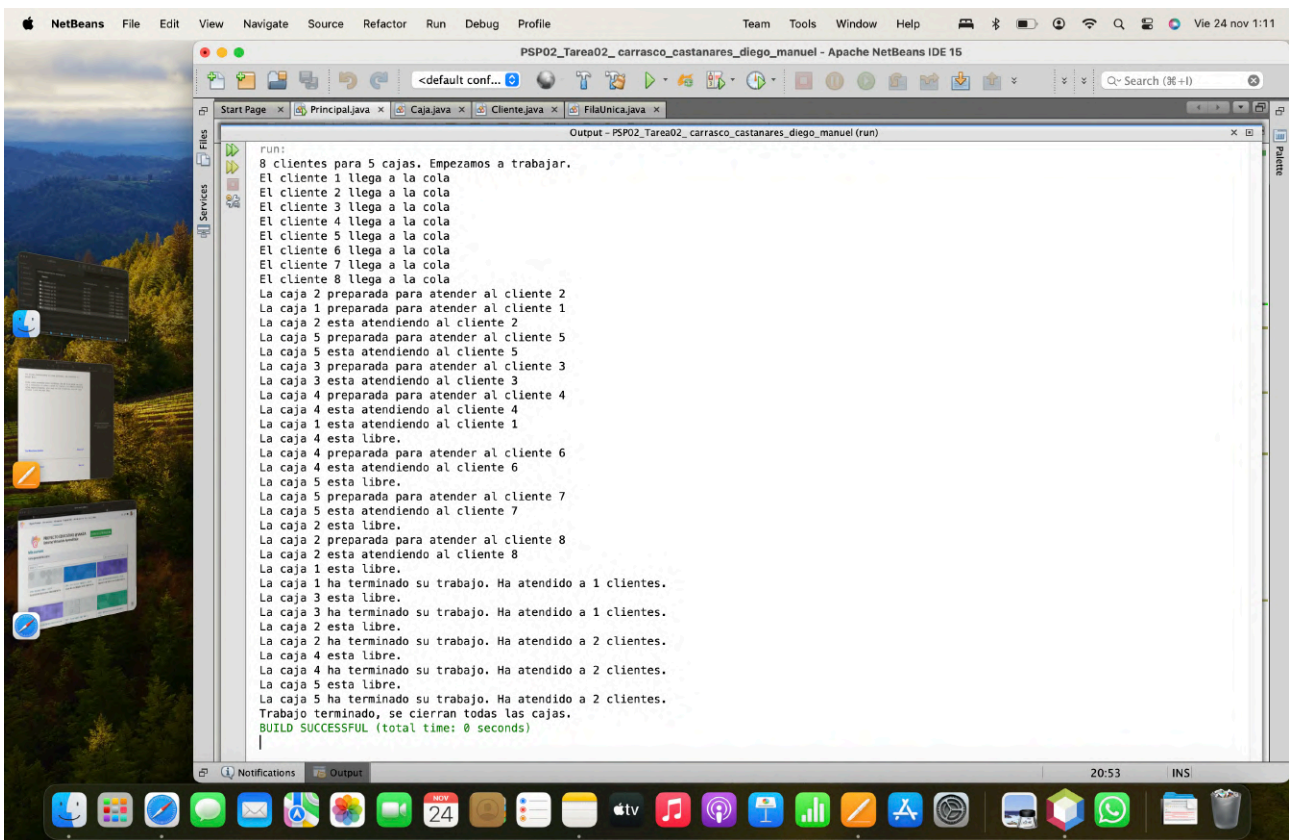


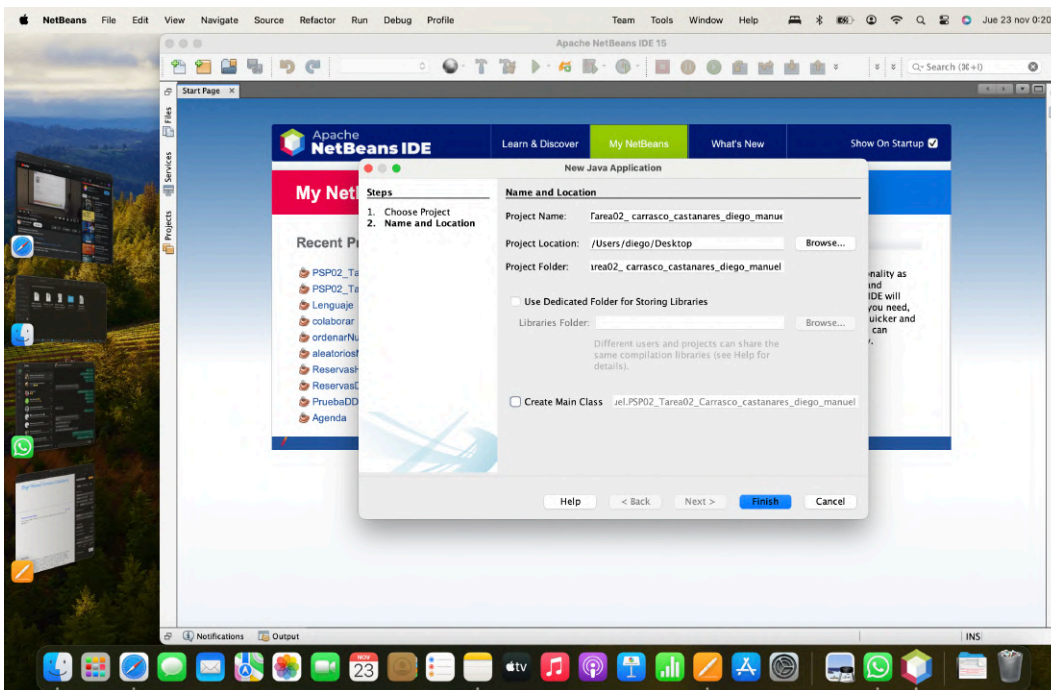
Tarea 2.2 para Programación de Servicios y Procesos



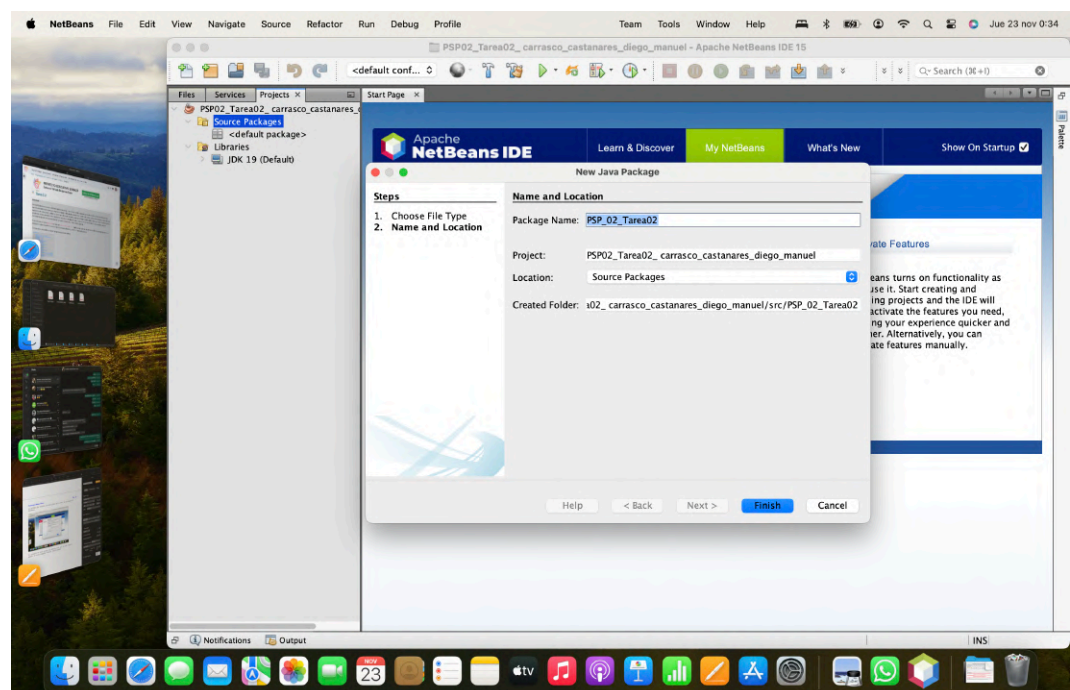
```
run:
8 clientes para 5 cajas. Empezamos a trabajar.
El cliente 1 llega a la cola
El cliente 2 llega a la cola
El cliente 3 llega a la cola
El cliente 4 llega a la cola
El cliente 5 llega a la cola
El cliente 6 llega a la cola
El cliente 7 llega a la cola
El cliente 8 llega a la cola
La caja 2 preparada para atender al cliente 2
La caja 1 preparada para atender al cliente 1
La caja 2 esta atendiendo al cliente 2
La caja 5 preparada para atender al cliente 5
La caja 5 esta atendiendo al cliente 5
La caja 3 preparada para atender al cliente 3
La caja 3 esta atendiendo al cliente 3
La caja 4 preparada para atender al cliente 4
La caja 4 esta atendiendo al cliente 4
La caja 1 esta atendiendo al cliente 1
La caja 4 esta libre.
La caja 4 preparada para atender al cliente 6
La caja 4 esta atendiendo al cliente 6
La caja 5 esta libre.
La caja 5 preparada para atender al cliente 7
La caja 5 esta atendiendo al cliente 7
La caja 2 esta libre.
La caja 2 preparada para atender al cliente 8
La caja 2 esta atendiendo al cliente 8
La caja 1 esta libre.
La caja 1 ha terminado su trabajo. Ha atendido a 1 clientes.
La caja 3 esta libre.
La caja 3 ha terminado su trabajo. Ha atendido a 1 clientes.
La caja 2 esta libre.
La caja 2 ha terminado su trabajo. Ha atendido a 2 clientes.
La caja 4 esta libre.
La caja 4 ha terminado su trabajo. Ha atendido a 2 clientes.
La caja 5 esta libre.
La caja 5 ha terminado su trabajo. Ha atendido a 2 clientes.
Trabajo terminado, se cierran todas las cajas.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Diego Manuel Carrasco Castañares

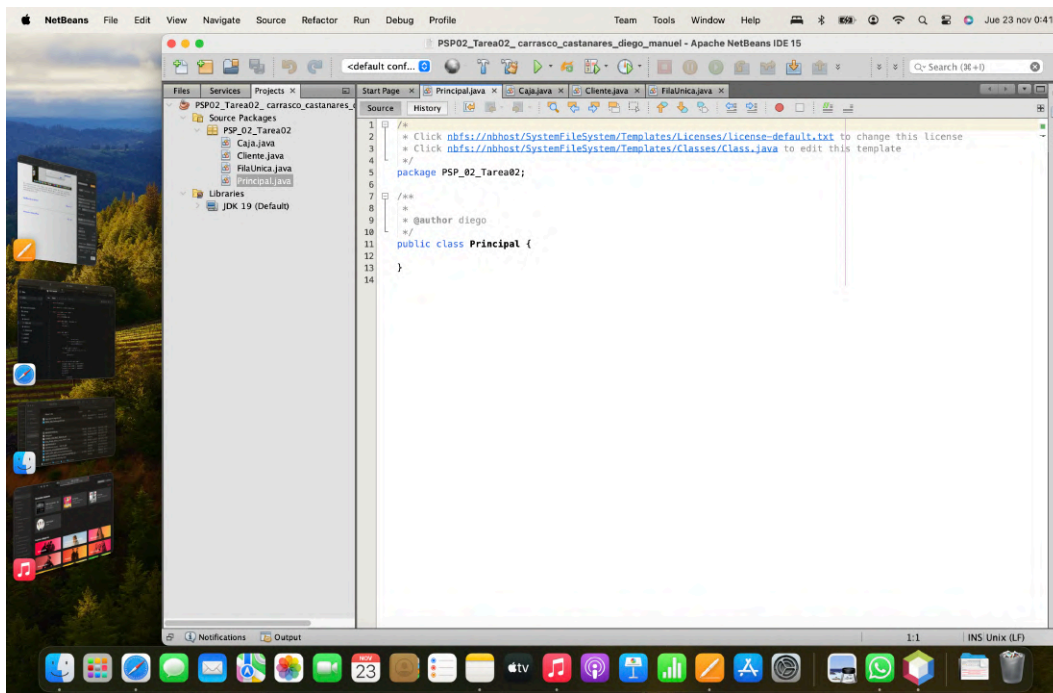
Lo primero que haremos en esta tarea será crear el proyecto en NetBeans.



El proyecto ha sido creado sin método main por lo que vamos a añadir un nuevo paquete llamado PSP02_Tarea2



El siguiente paso será crear las cuatro clases de java, la primera que será la principal y contendrá el método Main, la segunda que será Caja, la tercera que será Cliente y la cuarta que será FilaUnica y será la que tenga la región crítica.



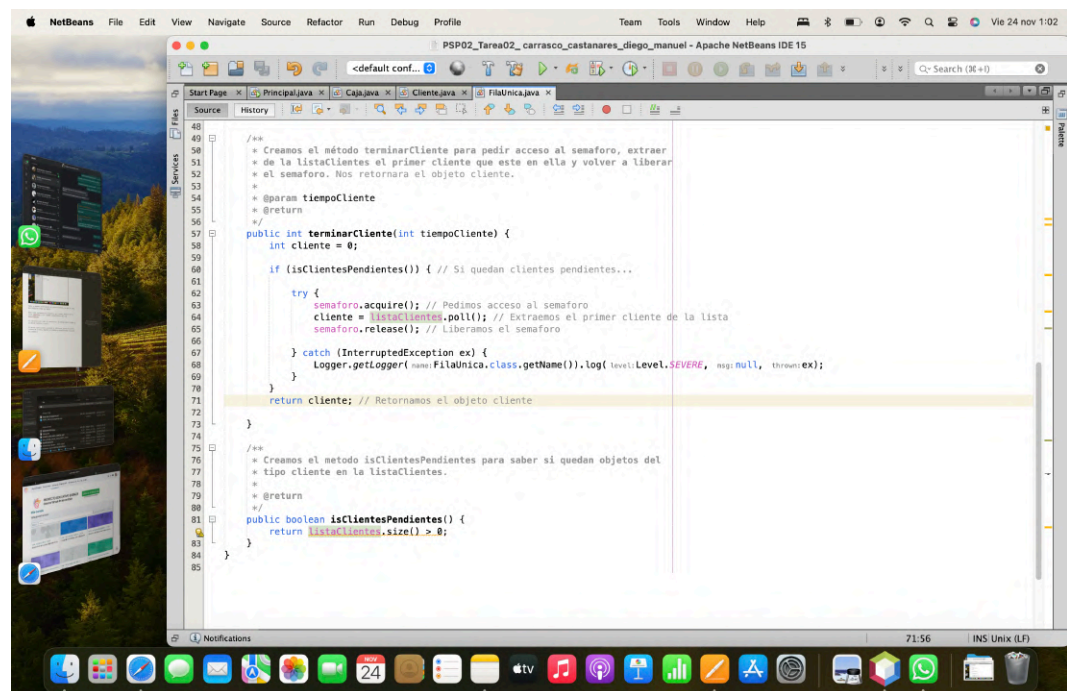
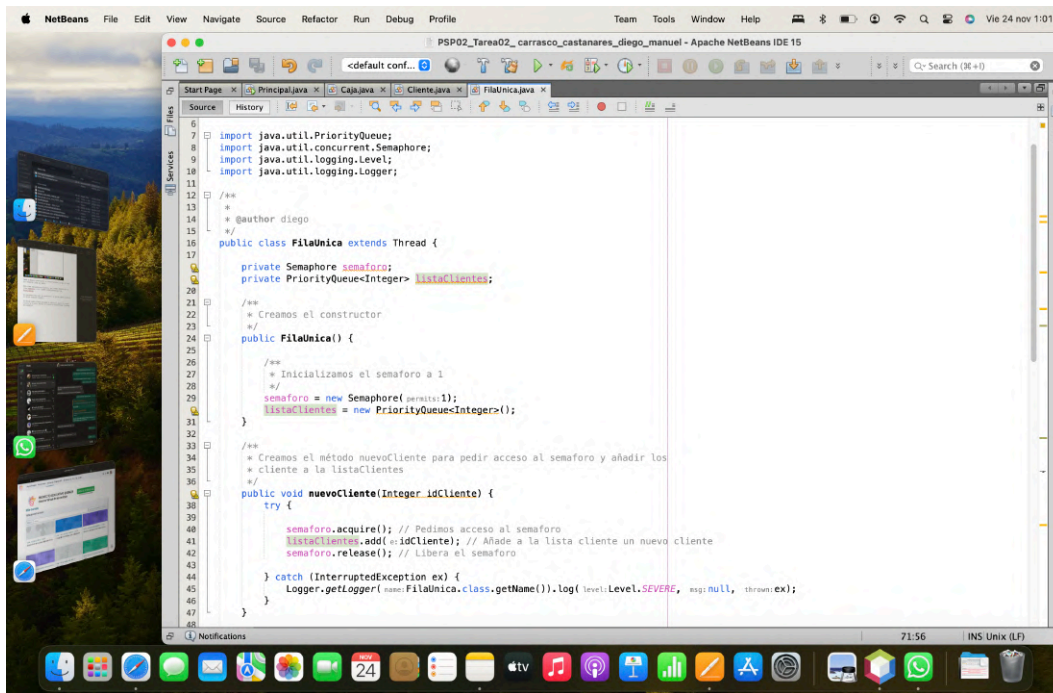
Vamos a empezar construyendo la clase `FilaUnica`, ya que es la clase que comparten todas las demás.

Esta clase contendrá dos atributos, uno llamado `semáforo` de la clase `semaphore` y otro llamado `listaClientes` del tipo `PriorityQueue`.

Lo siguiente será crear el constructor con sus dos atributos, el método `nuevo cliente`, al cual le pasaremos el `idCliente` de la clase cliente y el método `terminar cliente`, al cual le pasaremos el `tiempoCliente`. El método `nuevoCliente` recibirá el `idCliente` y empezará pidiendo acceso al semáforo, añadirá al cliente recibido a la `listaClientes` y saldrá del semáforo.

El método `terminarCliente` comprobará si quedan clientes en la cola. Si es así pedirá acceso al semáforo, extraerá el cliente que se encuentre en la primera posición de la lista `listaClientes` y liberará el semáforo.

Por último crearemos el método `isClientesPendientes` que será el encargado de controlar si quedan clientes en la lista `listaClientes`.

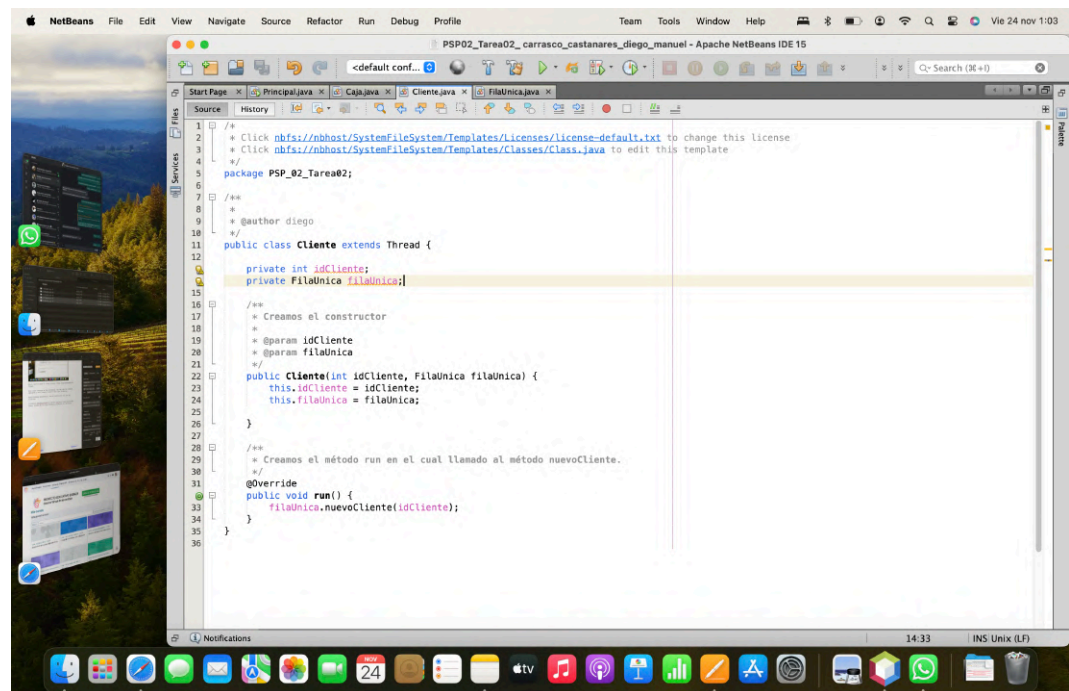


Ahora construiremos la clase cliente. Dicha clase extenderá de Thread.

Esta clase constara de dos atributos, uno de tipo int llamado idCliente y otro llamado filaUnica del tipo FilaUnica.

Posteriormente crearemos el método constructor con sus dos atributos.

Finalmente sobrescribimos el método run para llamar al método número cliente de la clase FilaUnica pasándole el idCliente.



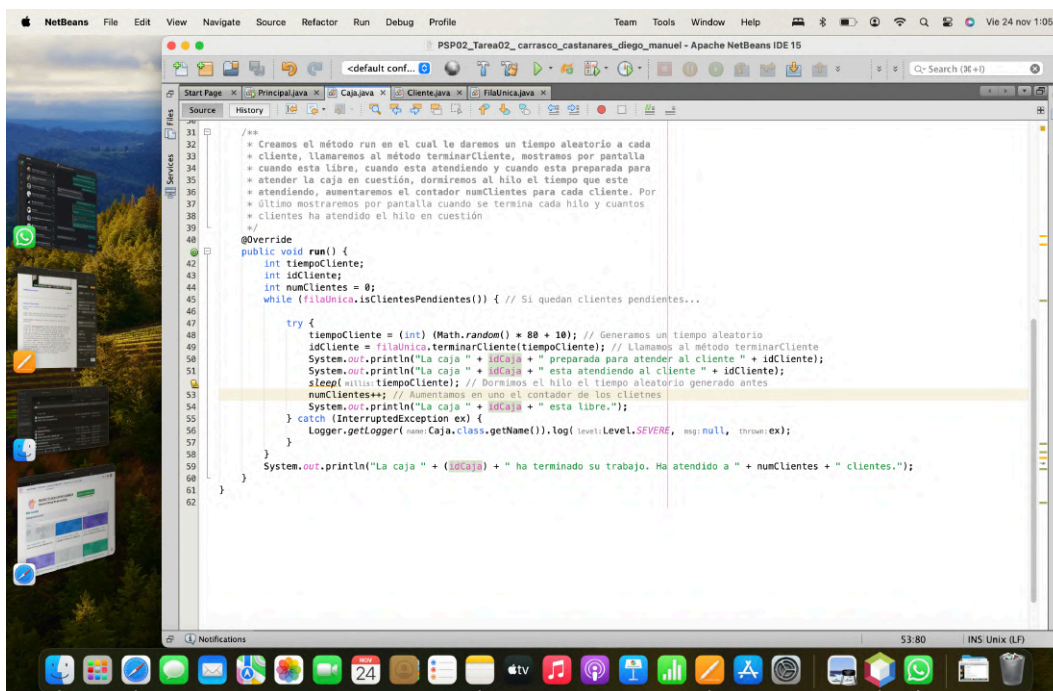
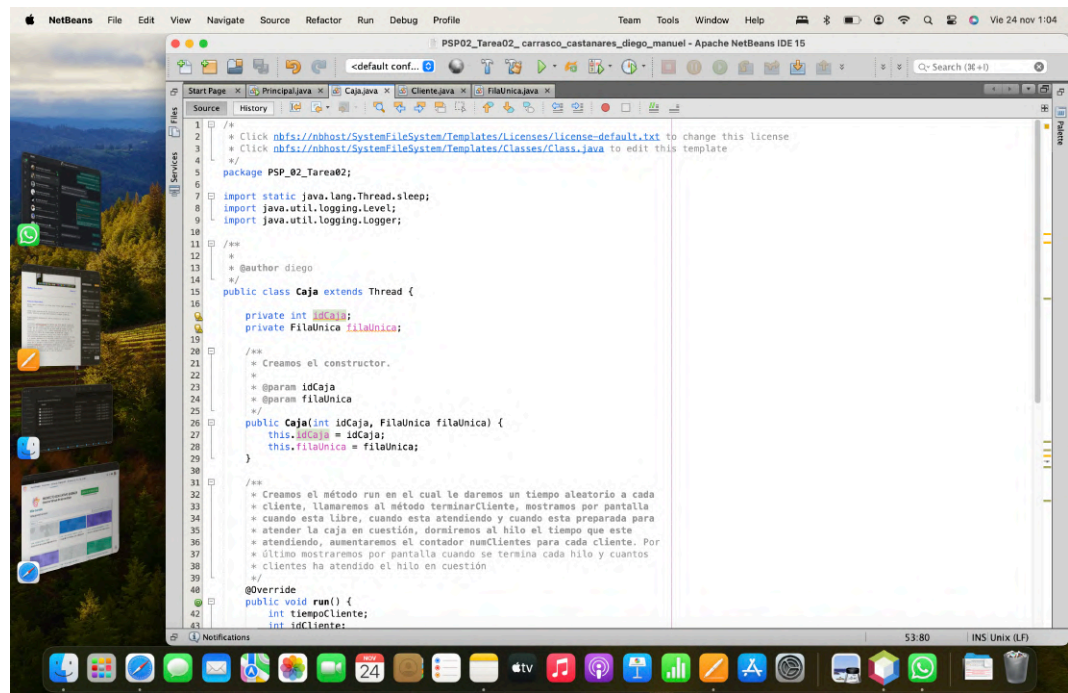
Ahora vamos a construir la clase caja. Dicha clase extenderá de Thread.

Dicha clase constara de dos atributos, uno de tipo int con el nombre idCaja y otro del tipo FilaUnica llamado filaUnica.

Posteriormente crearemos el método constructor con sus dos atributos.

Finalmente sobrescribimos el método run. Este método tendrá tres atributos, uno llamado tiempoCliente de tipo int que contendrá el tiempo que el cliente pasa en la caja (será un tiempo al azar creado por un round), otro llamado numCliente de tipo int que contendrá el número de cliente según su orden de llegada y otro llamado numClientes de tipo int y que inicializaremos a 0 para usar de contador y así saber cuantos clientes atiende cada caja.

Posteriormente crearemos un while, que llamará al método isClientesPendientes para comprobar si quedan clientes en la cola y si quedan guardara en la variable tiempoCliente un tiempo aleatorio, luego llamaremos al método terminarCliente pasándole el valor de la variable tiempoCliente y guardaremos su retorno en la variable idCliente. Mostraremos por pantalla si el hilo esta preparado para tender clientes y si esta atendiendo a un cliente. Seguidamente dormiremos el hilo el tiempo guardado en la variable tiempoCliente y mostraremos por pantalla que la caja en cuestión esta libre. Cuando el hilo finalice mostraremos por pantalla que la caja ha cerrado.



Por último construiremos la clase principal, que contendrá el método Main. Dicha clase contendrá cinco atributos, dos de ellos serán de tipo int y contendrán un número random de clientes y un número random de cajas respectivamente, otro será de tipo FilaUnica, otro de tipo Cliente y otro de tipo Caja.

Empezaremos mostrando por pantalla el número de clientes y el número de cajas generados.

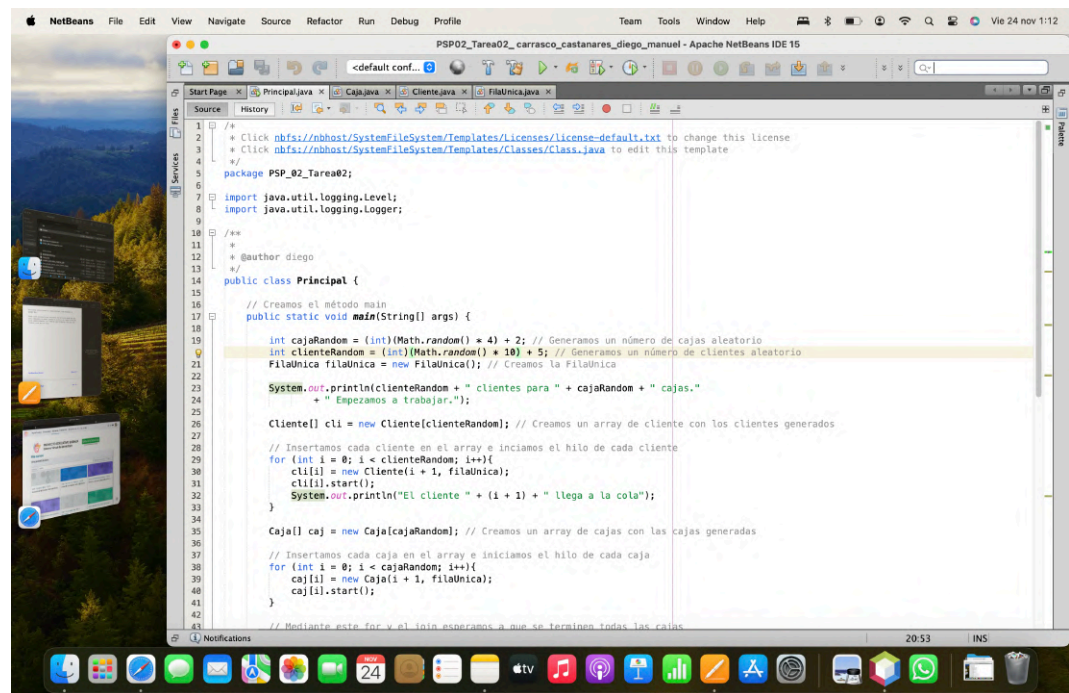
Mediante un bucle for crearemos y añadiremos cada cliente creado en el random al array cli que hará de cola, se lo pasaremos al

semáforo, iniciaremos el hilo y mostraremos por pantalla cada cliente que ha sido añadido a la cola.

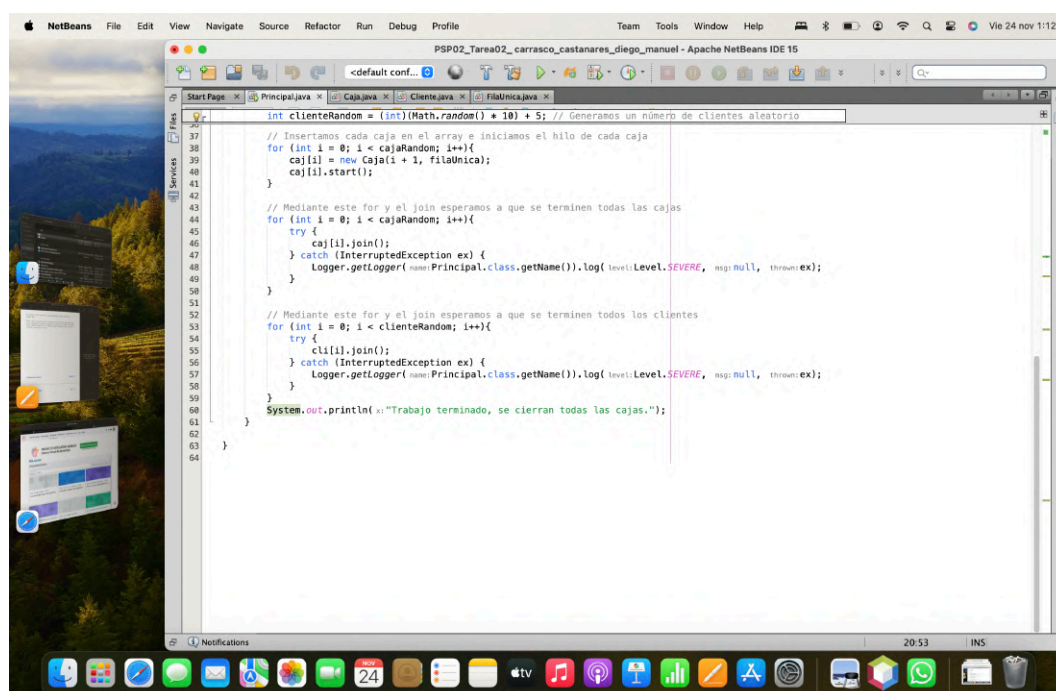
Mediante otro bucle for crearemos y añadiremos al array caj cada una de las cajas creadas en el random, se lo pasaremos a filaUnica e iniciaremos el hilo.

Posteriormente crearemos un bucle for para contener un join por cada caja generada y otro bucle for con la misma función pero para cada cliente generado.

Al final mostraremos cuando todos los hilos hayan finalizado



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package PSP_02_Tarea02;
6
7  import java.util.logging.Level;
8  import java.util.logging.Logger;
9
10 /*
11 *
12 * @author diego
13 */
14 public class Principal {
15
16     // Creamos el método main
17     public static void main(String[] args) {
18
19         int cajaRandom = (int)(Math.random() * 4) + 2; // Generamos un número de cajas aleatorio
20         int clienteRandom = (int)(Math.random() * 10) + 5; // Generamos un número de clientes aleatorio
21         FilaUnica filaUnica = new FilaUnica(); // Creamos la FilaUnica
22
23         System.out.println(clienteRandom + " clientes para " + cajaRandom + " cajas."
24             + " Empezamos a trabajar.");
25
26         Cliente[] cli = new Cliente[clienteRandom]; // Creamos un array de cliente con los clientes generados
27
28         // Insertamos cada cliente en el array e iniciamos el hilo de cada cliente
29         for (int i = 0; i < clienteRandom; i++){
30             cli[i] = new Cliente(i + 1, filaUnica);
31             cli[i].start();
32             System.out.println("El cliente " + (i + 1) + " llega a la cola");
33         }
34
35         Caja[] caj = new Caja[cajaRandom]; // Creamos un array de cajas con las cajas generadas
36
37         // Insertamos cada caja en el array e iniciamos el hilo de cada caja
38         for (int i = 0; i < cajaRandom; i++){
39             caj[i] = new Caja(i + 1, filaUnica);
40             caj[i].start();
41         }
42
43         // Mediante este for y el join esperamos a que se terminen todas las cajas
44     }
45 }
```



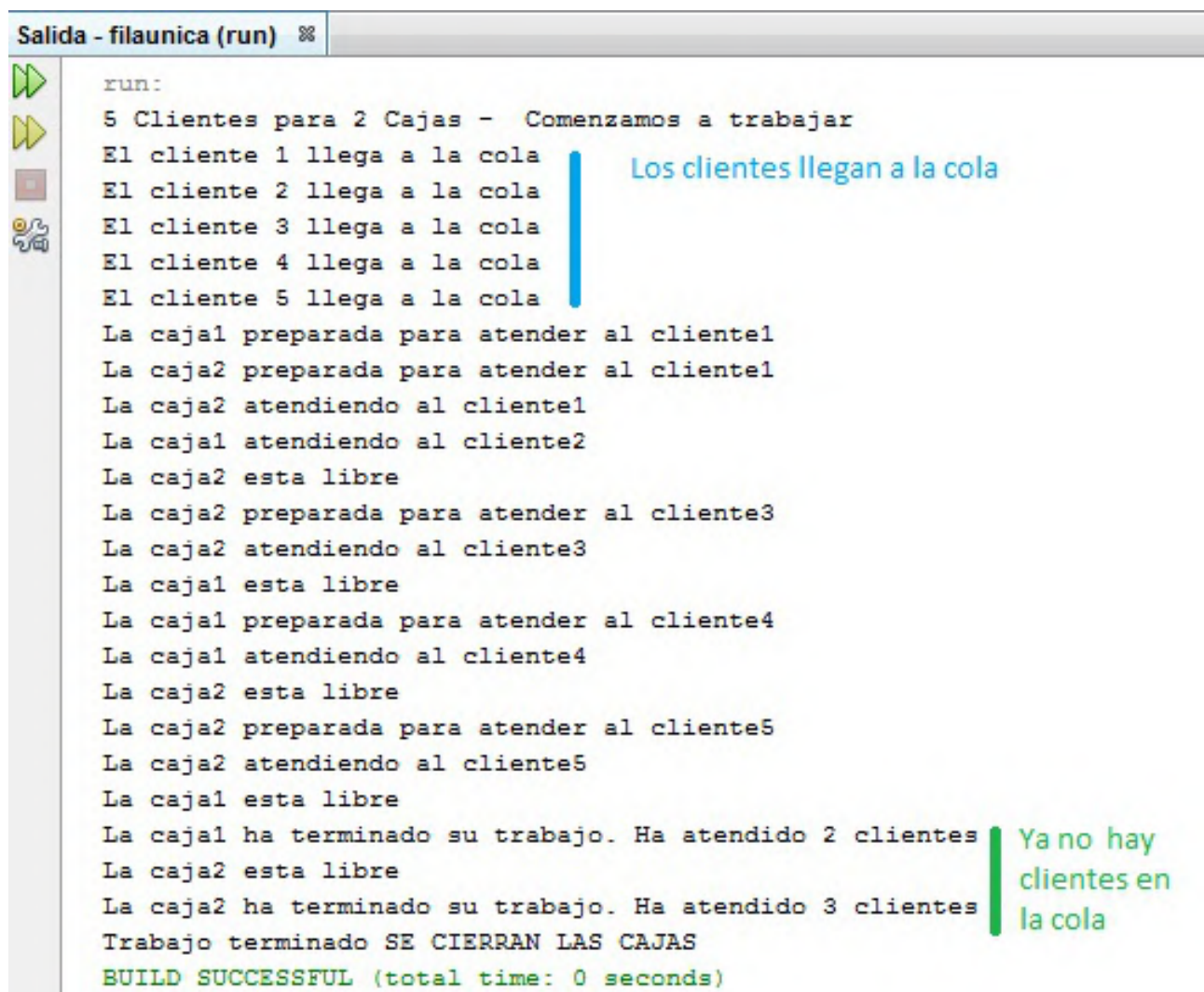
```
43         // Mediante este for y el join esperamos a que se terminen todas las cajas
44         for (int i = 0; i < cajaRandom; i++){
45             try {
46                 caj[i].join();
47             } catch (InterruptedException ex) {
48                 Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
49             }
50         }
51
52         // Mediante este for y el join esperamos a que se terminen todos los clientes
53         for (int i = 0; i < clienteRandom; i++){
54             try {
55                 cli[i].join();
56             } catch (InterruptedException ex) {
57                 Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
58             }
59         }
60         System.out.println("Trabajo terminado, se cierran todas las cajas.");
61     }
62 }
63
64 }
```

Enunciado

Un estudio determinó que un ciudadano pasa una media de cinco años de su vida esperando en distintas colas. Uno de los métodos más sencillos para disminuir el tiempo de espera se basa en la **fila única**. Se trata de un sistema que evita el estrés a los clientes, ya que mejora la experiencia de cada usuario y minimiza los tiempos de espera. Además aumenta la eficiencia del establecimiento en cuanto a rendimiento y a número de clientes atendidos, de un modo más relajado y controlado. Puede incluso ser un proceso dinámico, ya que existen algunas herramientas que de forma automática van avisando al siguiente cliente en la fila general.

Pues utilizando la clase `Semaphore` del paquete `java.util.concurrent`, vamos a implementar un sistema de fila única donde tendremos n clientes y m cajas disponibles, nuestra región crítica será `filaUnica` y el `main` simulara el proceso.

Te mostramos una posible salida del programa que debes realizar



```
run:
5 Clientes para 2 Cajas - Comenzamos a trabajar
El cliente 1 llega a la cola
El cliente 2 llega a la cola
El cliente 3 llega a la cola
El cliente 4 llega a la cola
El cliente 5 llega a la cola
La caja1 preparada para atender al cliente1
La caja2 preparada para atender al cliente1
La caja2 atendiendo al cliente1
La caja1 atendiendo al cliente2
La caja2 esta libre
La caja2 preparada para atender al cliente3
La caja2 atendiendo al cliente3
La caja1 esta libre
La caja1 preparada para atender al cliente4
La caja1 atendiendo al cliente4
La caja2 esta libre
La caja2 preparada para atender al cliente5
La caja2 atendiendo al cliente5
La caja1 esta libre
La caja1 ha terminado su trabajo. Ha atendido 2 clientes
La caja2 esta libre
La caja2 ha terminado su trabajo. Ha atendido 3 clientes
Trabajo terminado SE CIERRAN LAS CAJAS
BUILD SUCCESSFUL (total time: 0 seconds)
```

Los clientes llegan a la cola

Ya no hay clientes en la cola

Criterios de puntuación.

- Cliente: 1 punto
- Caja: 2 puntos
- FilaUnica: 5 puntos
- Main o programa principal: 2 puntos

Total: 10 puntos

Recursos mínimos necesarios para realizar la Tarea.

- Ordenador personal
- Sistema operativo Windows o Linux
- Conexión a Internet a través de ADSL
- JDK y JRE de Java
- Entorno NetBeans IDE 8.2.

Consejos y recomendaciones.

Tomar como referencia los ejercicios resueltos de los contenidos de la unidad de trabajo.

Indicaciones de entrega.

Debes crear un único fichero comprimido en el que debes incluir el o los proyectos Netbeans y documentos que correspondan a la realización de los ejercicios de la tarea.

Además adjuntarás un documento realizado con un procesador de texto con unas breves explicaciones de cómo has realizado los ejercicios, con las capturas de pantallas que estimes oportunas.

El fichero comprimido debe seguir la nomenclatura **PSPXX_TareaZZ_apellido1_apellido2_nombre**, donde XX se corresponde con la unidad y ZZ con el numero de la tarea.

Realiza el envío de dicho archivo comprimido a través de la plataforma. Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna **Begoña Sánchez Mañas para la primera unidad del MP de PSP**, debería nombrar esta tarea como...

PSP02_TAREA022_sanchez_manas_begona