30 DE ENERO DE 2021

PRÁCTICA 2: UKEA

PROGRAMACIÓN CONCURRENTE

ÁLVARO MARTÍNEZ QUIROGA
GRADO EN INGENIERÍA DE COMPUTADORES
URJC

ÍNDICE DEL DOCUMENTO

Contenido

Ilustraciones

Presentación

El objetivo principal de la práctica es el de conseguir una simulación en Java de una tienda donde se pueden realizar pedidos por los clientes y estos son tratados por los empleados de la tienda. El programa ha de ser completamente concurrente usando un hilo por persona simulada, es decir, una persona es representada por un hilo de ejecución, un empleado por otro hilo de ejecución…

En la práctica se incluyen diferentes clases para el correcto funcionamiento de la misma, estas son:

- Producto.java.
- ProductoEspecial.java.
- Pedido.java.
- Almacen.java.
- Ukea.java.
- Personal.java.

Estas clases son explicadas más adelante.

También se incluye en la práctica un fichero de texto dentro de la carpeta donde se encuentran estas clases anteriores, el cual contiene ideas del desarrollo del sistema de pedidos y las diferentes acciones que realizarán los hilos del programa, sin embargo, todo lo presentado en aquel fichero de texto será explicado a continuación.

Funcionalidad, atributos y métodos de las clases

Vamos a comenzar, clase por clase, explicando sus usos, de menor complejidad a mayor complejidad.

Clase Produco.java

Como el propio nombre indica esta clase se ocupará de crear nuevos tipos de productos para su venta.

Atributos:

- Id: identificador de tipo entero, para poder diferenciar los diferentes productos, no pueden existir en una misma ejecución dos productos con el mismo id.
- Precio: de tipo decimal, indica el coste del producto en concreto.

Métodos:

- Constructor: recibe el id y el precio.
- Getters y setters.
- ToString: devuelve en cadena de texto el id y el precio.

Clase ProductoEspecial.java

Esta clase será la que se ocupe de la creación de productos especiales en el sistema.

Atributos:

- Dos objetos de la clase producto anterior.

Métodos:

- Constructor: recibe dos objetos del tipo Producto.
- Getters y setters.
- ToString: devuelve en cadena de texto los métodos toString de ambos Productos.

Clase Pedido.java

Será la clase que se ocupe de almacenar una serie de productos para realizar una orden de compra en la tienda. Este puede ser un pedido especial o no especial, para que sea especial el pedido ha de tener objetos del tipo ProductoEspecial, para que no sea especial no tiene que tener ningún objeto de este tipo, solo objetos del tipo Producto.

Atributos:

- Id: identificador de tipo entero, para ser capaces de filtrar por id los pedidos más adelante y controlar que no haya repetidos.
- Número de productos en el pedido: de tipo entero, almacena el número de productos que hay en un pedido, no puede ser mayor que 5.
- Número de productos que ha pedido el cliente: de tipo entero, para poder iterar recogiendo los productos pedidos por los clientes.
- Dos listas: una de tipo Producto y la otra de tipo ProductoEspecial. Sirven para guardar las colecciones de productos seleccionados en el Pedido (los que pide el cliente).
- Precio: de tipo decimal, guarda el precio del Pedido.
- Lleno: de tipo booleano, indica si se pueden seguir metiendo o no productos en el pedido.
- A continuación, tenemos una serie de atributos para la gestión de la clase Ukea dentro del Pedido, estos son:
    - Id cliente: de tipo entero, representa el id (único) del cliente que ha realizado ese pedido.
    - Posición en la cola: de tipo entero, representa una posición en una cola de clientes, explicada a continuación.
    - Realizado: de tipo booleano, para conocer el estado del pedido, es decir, si se puede o no entregar al cliente.

Métodos:

- Constructor: recibe el número de productos y si tiene productos especiales o no el pedido.
- Añadir producto: recibe un producto simple y si el pedido no está lleno lo añade a la lista correspondiente.
- Añadir especial: recibe un producto especial y si el pedido no está lleno lo añade a la lista correspondiente.
- Calcula precio: calcula, guarda (en el pedido) y devuelve el precio del Pedido.
- Es especial: devuelve si verdadero si el pedido contiene productos especiales.
- Getters, Setters y toString.

Clase Almacen.java

Contiene el almacén de los productos de la tienda dentro de una lista de productos. Existen 10 productos diferentes y hay 15 unidades iniciales de cada uno al comienzo de la ejecución.

Atributos:

- Número de diferentes productos: de tipo entero y no modificable, indica los distintos tipos de productos.
- Número de existencias: de tipo entero y no modificable, indica cuánto stock hay en la tienda de cada producto.
- Dos enteros para marcar el máximo precio y el mínimo precio de los productos.
- Lista de productos.
- Existencias de cada producto: array de enteros que contiene el stock de cada producto. Tendrá 10 posiciones ya que hay 10 productos.

Métodos:

- Constructor: inicializa las variables.
- Dame producto simple: devuelve un objeto de la clase Producto con el id que se le pasa al método.
- Dame producto especial: devuelve un objeto de la clase ProductoEspecial con dos ids que se le pasa al método.
- Sacar producto simple: recibe un identificador de producto y saca un producto simple del stock siempre que haya existencias de este.
- Sacar producto especial: recibe dos identificadores de productos y saca un producto especial del stock siempre que haya existencias de este.
- Comprueba existencias: devuelve verdadero si hay existencias del identificador que se le pasa.
- Devolver pedido: actualiza las existencias que se encuentran en el pedido que se le pasa.
- Estado de almacén: imprime el número de existencias de cada producto.

Clase Ukea.java

Se ocupa de la sincronización para los diferentes procesos del sistema.

Atributos:

- Almacén: objeto de la clase almacén.
- Semáforo de acceso al almacén: solo 1 hilo puede acceder cada vez al almacén para sacar o depositar artículos.
- Lista de pedidos: de tipo mapa concurrente, por entrada tendrá un identificador y un objeto de la clase Pedido.
- Contador de pedidos: de tipo entero atómico para llevar una cuenta de los pedidos que se realizan.
- Contador de pedidos en preparación: de tipo entero atómico para saber si hay pedidos esperando a ser atendidos por los empleados y llevarlos a línea de producción.
- Dispensadores: de tipo cola bloqueante, uno donde se colocarán aquellos pedidos de tipo especial y más tarde serán enviados a la cola de entrega de cliente la cual es el otro dispensador.
- Colas de preparación: de tipo cola bloqueante, donde se almacenan los pedidos que requieren de preparación y empaquetado a realizar por los empleados.

- Se ha decidido añadir otro módulo al enunciado inicial de la práctica, este está relacionado con los siguientes atributos:
    - Número máximo de clientes en la tienda: de tipo entero y no modificable. Se ha decidido que tenga el valor de 5, es decir que si hay 5 clientes esperando a que sus pedidos sean empaquetados los demás clientes que quieran realizar pedidos tendrán que esperar a que una posición en la cola este libre.
    - Línea de cajas: simulado por un array de semáforos. Cuando un cliente hace un pedido se le asigna una posición en línea de cajas y hasta que el pedido no ha sido procesado, espera en esa posición.
    - Línea de cajas: de tipo booleano, indica las posiciones ocupadas y las libres como verdadero (ocupado) y falso (libre).

Métodos:

- Constructor: inicia las variables y el tamaño de los dispensadores con un número que recibe.
- Posición libre la cola: devuelve la posición libre en la cola de clientes (si hay), si no, devuelve -1.
- Realiza pedido: método usado por los hilos cliente. Recibe un pedido, el id del cliente que lo realiza y su dinero. Inicializa el pedido con los productos que haya solicitado el cliente. Más tarde, busca una posición libre en la cola de pedidos, si no hay espera un tiempo y lo volverá a intentar más adelante. Cambia las variables respectivas a ese pedido y se queda esperando a su preparación en línea de cajas. Cuando el pedido es entregado de vuelta, se comprueba si se ha realizado (para saber si pagar) y si es así y el cliente tiene dinero lo paga.
- Devolver pedido: método usado por los hilos cliente. Recibe un pedido y actualiza (devuelve) sus respectivos productos en el almacén. Este método se usa si el cliente no puede pagar su pedido.
- Entrega pedido: recibe el id de un empleado para la información que se imprimirá por pantalla. El método comprueba si hay pedidos que entregar, si hay alguno este es entregado en la posición de la cola al cliente en concreto que lo realizó previamente.
- Pedido a producción: recibe el id de un empleado para la información que se imprimirá por pantalla. Comprueba si el número de pedidos que se encuentran en preparación es menor que el número de pedidos general. Si es así, obtiene aquel pedido que espere a pasar a producción y lo pone en la línea de preparación correspondiente (línea de preparar normales o línea de preparar especiales).
- Añadido de pedidos en colas correspondientes, son dos métodos diferentes:
    - Añade pedido normal: añade un pedido sencillo a la cola de producción de pedidos normales.
    - Añade pedido especial: añade un pedido especial a la cola de producción de pedidos especiales.
- Prepara pedido especial: recibe el id de un empleado para la información que se imprimirá por pantalla. Comprueba si hay pedidos especiales que preparar. Si es así coge el primero de estos y lo elimina de la lista de pedidos especiales en espera de preparación. Accede al almacén para obtener los artículos seleccionados y cuando tiene el pedido listo lo deposita en el dispensador de pedidos especiales a la espera de ser recogidos.

- Cambiar pedido especial a línea de entrega principal: recibe el id de un empleado para la información que se imprimirá por pantalla. Comprueba si existe algún pedido especial que requiera de ser transportado a la línea principal de entrega, si es así, lo transporta de línea.
- Prepara pedido normal: recibe el id de un empleado para la información que se imprimirá por pantalla. Comprueba si existe algún pedido normal que preparar, si es así comienza con el proceso de preparación. Cuando lo tiene listo lo deposita en el dispensador de pedidos a clientes.
- Información tienda: no recibe parámetros, es un método desarrollado para uso exclusivo del encargado (explicado a continuación) de la tienda.

Clase Personal.java

Contiene el ciclo de vida de los procesos.

Atributos:

- Constantes finales para el número de hilos de ejecución y el número de dispensadores:
  - NUM_CLIENTES.
  - NUM_EMP_COGE_PEDIDOS.
  - NUM_EMP_ELABORA_PRODUCTOS.
  - NUM_EMP_PREPARA_PEDIDO.
  - NUM_DISPENSADORES.
- Número de hilos a ejecutar: N_THREADS, contiene la suma de las constantes anteriores.
- Ejecutor: objeto al que se le enviarán las tareas (hilos) para que los ejecute.
- Ukea: objeto de la clase Ukea para el acceso a sus métodos y gestión de la tienda.
- Constantes para el número mínimo y máximo de productos.
- Velocidad de llegada: hace referencia a la velocidad de llegada de pedidos a la tienda. Cada hilo cliente se duerme un tiempo al finalizar y pagar (o no) un pedido. Como indica el enunciado el tiempo de llegada ha de ser fácilmente modificable por eso se ha decidido incluir esta constante fácil de cambiar para que los pedidos puedan tardar más o menos en llegar.

Gestión de apertura de la tienda y turnos de trabajo

Para llevar a cabo esta simulación de gestión de la tienda se ha desarrollado la siguiente idea:

- Se simulará 1 día natural en segundos, cada 2 segundos pasará 1 hora. Estos tiempos son marcados por el encargado de la tienda.
- A las 6:00 se abrirá la tienda.
- A las 13:00 se cambia de turno de empleados.
- A las 20:00 se cierra la tienda hasta el próximo día.

Atributos para dicha gestión:

- Tienda abierta: de tipo booleano atómico. Indica si la tienda está abierta (verdadero) o cerrada (falso).
- Hora: de tipo entero atómico. Marca la hora actual simulada del programa.
- Turno: de tipo booleano atómico. Indica que empleados tienen que trabajar.

La gestión de apertura, cierre, entrada a la tienda (turno de trabajo) o salida de la tienda (turno de trabajo) es controlada por las siguientes variables:

- Hilos esperando apertura de la tienda: de tipo entero atómico. Cuenta el número de hilos que han intentado acceder a la tienda, pero estaba cerrada en el momento de la consulta.
- Espera apertura: de tipo semáforo. Contiene 0 permisos para bloquear a los procesos (hilos).
- Hilos esperando trabajar: de tipo entero atómico. Similar al contador de hilos esperando acceder a la tienda, pero en este caso es exclusivo para los empleados.
- Espera trabajar: de tipo semáforo. Contiene también 0 permisos para bloquear a los hilos (empleados).

La gestión de la tienda es sencilla y la lleva a cabo el hilo encargado, esta es:

- Si la tienda está cerrada y cualquier hilo intenta acceder a la misma, el contador de hilos intentando entrar a la tienda es incrementado en 1 y el hilo que ha intentado acceder a la tienda se bloquea en el semáforo de espera apertura.
- En el momento de la apertura, el encargado comprueba si hay hilos esperando a entrar, si es así desbloquea el semáforo de espera apertura tantas veces como hayan intentado entrar, esta variable se guarda en hilos esperando apertura tienda (entero).
    - Ejemplo:
        - 3 hilos intentan entrar a la tienda, esta está cerrada.
        - Los hilos incrementan el contador y este obtiene el valor de 3.
        - Cuando el encargado abre la tienda comprueba si esta variable es distinta de 0.
        - Si es así libera el semáforo con la constante, es decir, semáforo.release(numeroIntentosEntrarATienda).
- Para la gestión de hilos empleados a la hora de trabajar se realizan exactamente los mismos pasos, pero con las variables respectivas, es decir, con hilos esperando trabajar (contador) y espera trabajar (semáforo).

Se ha decidido como implementación de simulación que el primer turno de 6:00 a 13:00 trabajarán los hilos con id par (incluye el 0) y el segundo turno de 13:00 a 20:00 trabajarán los hilos con id impar.

Se ha seguido esta implementación ya que si no se dispusiera de estos semáforos y variables para la espera los hilos estarían constantemente intentando entrar a la tienda y sin dejar que, por ejemplo, el encargado hiciese una progresión de las horas correcta.

Métodos:

- Hilos clientes: intentan acceder a la tienda, si pueden hacen un pedido. Cuando se lo entregan comprueban si lo pueden pagar, si es así lo pagan y esperan un tiempo para pedir de nuevo. Si no lo pueden pagar lo devuelven.
- Hilos empleados coge pedidos: si es su turno de trabajar primero comprueban si hay pedidos que entregar a clientes y después atienden si hay clientes que hayan realizado pedidos. Se ha decidido de esta manera ya que si la cola de estrega estuviese llena esta tendría prioridad de vaciado para agilizar la gestión de la tienda.
- Hilos empleados elabora productos: si es su turno de trabajo comprueba si hay pedidos con productos especiales a preparar los prepara y cuando están listos los deposita en el dispensador de pedidos especiales.

- Hilos empleados prepara pedidos: si es su turno de trabajo comprueba si hay pedidos especiales listos que mover del dispensador especiales al dispensador de entrega a clientes para agilizar el flujo de la tienda. Después de esto, comprueba si hay pedidos normales que preparar, si es así lo prepara y lo inserta en la línea de entrega a clientes.
- Hilo empleado: gestiona el paso del tiempo, cierre, apertura y cambios de turno de la tienda. También desbloquea a los hilos bloqueados.
- Exec: crea las tareas y las envía al ejecutor. Le da un tiempo de ejecución máximo de 30 minutos.
- Main: método ejecutor del programa principal. Hace uso del método exec anterior.

Ejecución

Vamos a llevar a cabo un ejemplo de ejecución con imágenes de la salida por pantalla del programa.

Inicio del sistema

```
***    UKEA    ***


Encargado activo.
empCogePedidos 0 activo.
empCogePedidos 2 activo.
empCogePedidos 1 activo.
empCogePedidos 3 activo.
empElaboraProductos 0 activo.
empElaboraProductos 1 activo.
empElaboraProductos 2 activo.
empPreparaPedido 0 activo.
empPreparaPedido 1 activo.
empPreparaPedido 2 activo.
Cliente 0 activo.
Cliente 1 activo.
Cliente 2 activo.
Cliente 3 activo.
Cliente 4 activo.
Cliente 5 activo.
Cliente 6 activo.

  Tiempo: 1:00


  Tiempo: 2:00
```

Ilustración 1: inicio de ejecución

Podemos observar que en el comienzo de la ejecución se informa de que hilos son los que han comenzado a ejecutar, estos hilos se encuentran dentro del ejecutor en este momento. Más tarde comienza la simulación de horas del sistema. Cada uno de ellos entra en ejecución en orden de 0 a n ya que son creados con bucles de tipo for y nada más están disponibles entran al ejecutor, en orden de creación.

Primeros pedidos

```
  Tiempo: 5:00


  Tiempo: 6:00


Encargado: tienda abierta.

Cliente 5, quiere realizar un pedido.
Cliente 5, buscando una posicion libre en la cola.
Cliente 2, quiere realizar un pedido.

  Tiempo: 7:00

Cliente: 5, ha hecho el pedido: [id=1, numeroProductos=3, precio=264.84679188928015]. Esperando...
Cliente 2, buscando una posicion libre en la cola.
EmpCogePedidos: 0, pedido: [id=1, numeroProductos=3, precio=264.84679188928015] a produccion.
EmpPreparaPedidos 2, pedido: [id=1, numeroProductos=3, precio=264.84679188928015] en preparacion...
EmpPreparaPedidos: 2, pedido [id=1, numeroProductos=3, precio=264.84679188928015] incluido en linea de entrega a cliente
Cliente: 2, ha hecho el pedido: [id=2, numeroProductos=2, precio=128.50951369985108]. Esperando...
EmpCogePedidos: 2, entregando pedido: [id=1, numeroProductos=3, precio=264.84679188928015]
EmpCogePedidos: 0, pedido: [id=2, numeroProductos=2, precio=128.50951369985108] a produccion.
Cliente 0, quiere realizar un pedido.
Cliente 0, buscando una posicion libre en la cola.
```

Ilustración 2: primera apertura

Cuando la tienda se encuentra abierta el encargado informa y observamos que los empleados trabajando son aquellos con id par. Los clientes pueden acceder todos a la tienda.

Ciclo de pedidos en ejecución

```
  Tiempo: 8:00

El cliente 5 no dispone de fondos para pagar el pedido [id=1, numeroProductos=3, precio=390.531268508936]. ABORTANDO OPERACION...
Devuelto el pedido: [id=1, numeroProductos=3, precio=390.531268508936]
EmpPreparaPedidos 0, pedido: [id=2, numeroProductos=2, precio=128.50951369985108] en preparacion...
Cliente 6, quiere realizar un pedido.
Cliente 6, buscando una posicion libre en la cola.
Cliente: 0, ha hecho el pedido: [id=3, numeroProductos=3, precio=225.0103426052715]. Esperando...
Cliente 1, quiere realizar un pedido.
EmpPreparaPedidos: 0, pedido [id=2, numeroProductos=2, precio=128.50951369985108] incluido en linea de entrega a cliente
Cliente 1, buscando una posicion libre en la cola.
EmpCogePedidos: 2, pedido: [id=3, numeroProductos=3, precio=225.0103426052715] a produccion.
EmpCogePedidos: 0, entregando pedido: [id=2, numeroProductos=2, precio=128.50951369985108]
Cliente: 6, ha hecho el pedido: [id=4, numeroProductos=3, precio=253.95068048940882]. Esperando...
Cliente 3, quiere realizar un pedido.
Cliente 3, buscando una posicion libre en la cola.
EmpPreparaPedidos 0, pedido: [id=3, numeroProductos=3, precio=225.0103426052715] en preparacion...

  Tiempo: 9:00

EmpCogePedidos: 2, pedido: [id=4, numeroProductos=3, precio=253.95068048940882] a produccion.
El cliente 2 dispone de fondos para pagar el pedido [id=2, numeroProductos=2, precio=213.2067102635835]. REALIZANDO OPERACION...
Cliente: 1, ha hecho el pedido: [id=5, numeroProductos=2, precio=109.36272036346445]. Esperando...
EmpCogePedidos: 0, pedido: [id=5, numeroProductos=2, precio=109.36272036346445] a produccion.
EmpPreparaPedidos 2, pedido: [id=4, numeroProductos=3, precio=253.95068048940882] en preparacion...
Cliente: 3, ha hecho el pedido: [id=6, numeroProductos=3, precio=227.1588668996482]. Esperando...
EmpCogePedidos: 2, pedido: [id=6, numeroProductos=3, precio=227.1588668996482] a produccion.
EmpPreparaPedidos: 0, pedido [id=3, numeroProductos=3, precio=225.0103426052715] incluido en linea de entrega a cliente
EmpCogePedidos: 0, entregando pedido: [id=3, numeroProductos=3, precio=225.0103426052715]

  Tiempo: 10:00

El cliente 0 no dispone de fondos para pagar el pedido [id=3, numeroProductos=3, precio=320.48595847635846]. ABORTANDO OPERACION...
Cliente 4, quiere realizar un pedido.
EmpPreparaPedidos 0, pedido: [id=5, numeroProductos=2, precio=109.36272036346445] en preparacion...
Cliente 4, buscando una posicion libre en la cola.
Cliente 5, quiere realizar un pedido.
Cliente 5, buscando una posicion libre en la cola.
```

Ilustración 3: continuación de ejecución

Se observa en esta imagen que el tiempo va progresando en el programa. También, si un cliente no dispone de fondos devuelve el pedido.

Cambio de turno

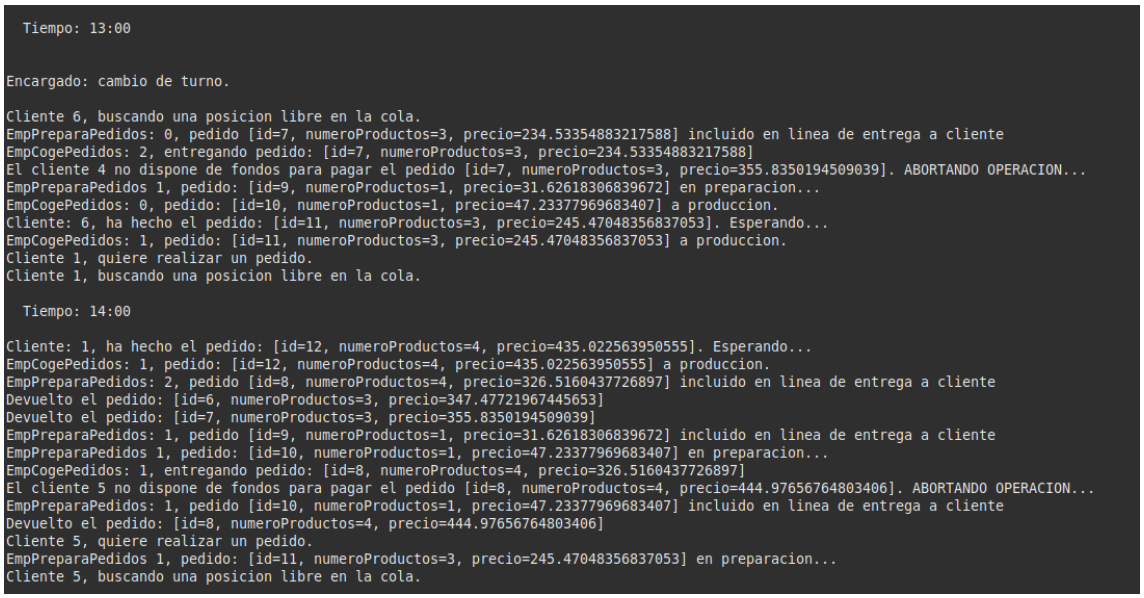

Ilustración 4: cambio de turno

En este momento los empleados con id par han de salir de la tienda y simular que están haciendo otras cosas. Podemos observar que han de terminar su ejecución en la que se encuentran y después de esto salir de la tienda. También vemos que pasan a tienda aquellos empleados con id impar.

Fin de jornada y cierre de tienda

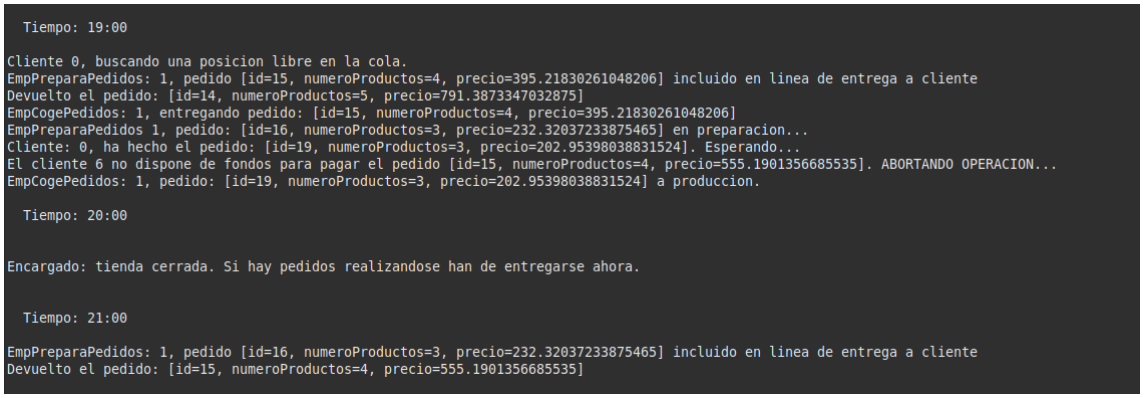

Ilustración 5: fin de jornada y cierre

Cuando dan las 20:00 la tienda se cierra y el encargado informa de que todos los pedidos que se están preparando han de entregarse en ese momento. No se podrán hacer más pedidos hasta el próximo día.

Información de un día

```
  Tiempo: 24:00


ESTADO TIENDA :
Pedidos: 19
FIN ESTADO TIENDA.


ESTADO ALMACEN :
-> Producto: Producto [id=0, precio=31.62618306839672], existencias: 13
-> Producto: Producto [id=1, precio=48.954271832893916], existencias: 15
-> Producto: Producto [id=2, precio=23.148771714433686], existencias: 15
-> Producto: Producto [id=3, precio=43.5881479880464], existencias: 15
-> Producto: Producto [id=4, precio=31.46231935822041], existencias: 15
-> Producto: Producto [id=5, precio=14.89516096979631], existencias: 14
-> Producto: Producto [id=6, precio=40.88487942761371], existencias: 14
-> Producto: Producto [id=7, precio=45.10402171836521], existencias: 14
-> Producto: Producto [id=8, precio=47.23377969683407], existencias: 12
-> Producto: Producto [id=9, precio=43.81231713611868], existencias: 14
FIN ESTADO ALMACEN.


  Tiempo: 1:00
```

Ilustración 6: información tienda

Se ha decidido que a las 24:00 el encargado informe de algunos parámetros para conocer el estado de la tienda.

Dificultades de la práctica

He encontrado dificultades a la hora de simular el paso de las horas en el programa y conseguir que los hilos se sincronizasen bien con los métodos de la clase Ukea. Por lo demás, la concurrencia del sistema es fácil usando las diferentes estructuras de datos y métodos concurrentes que nos proporciona el API de Java.

Opinión personal

Considero esta práctica fundamental para el entendimiento de cómo funciona la concurrencia y las diferentes maneras de sincronizar hilos o procesos. También para entender el uso de las diferentes estructuras de datos proporcionadas por el API de Java, que como se ha dicho en el apartado anterior, permiten que el desarrollo de programas o aplicaciones de este tipo sea más sencillo. Tuve que desarrollar otra clase Ukea para el sistema ya que la primera de ellas era demasiado complicada de entender y enrevesada. Sin embargo, en la siguiente implementación decidí usar un mapa concurrente y listas de colas bloqueantes, lo que ha permitido que el desarrollo del sistema sea muchísimo más sencillo.