Ejercicios Unidad 2

Programación multihilo

1 Creación de hilos

Escribir un programa concurrente que ejecute *N* hilos. Todos los hilos realizan el mismo trabajo:

- Imprimir una línea identificándose, anunciando el inicio de su ejecución y mostrando el tiempo que permanecerán dormidos.
- Dormir durante el tiempo especificado para cada uno de ellos.
- Imprimir una línea donde se identifiquen de nuevo para anunciar su finalización.

Cuando todos los hilos hayan finalizado su tarea, el hilo principal imprimirá un mensaje informando de ello.

2 TIC TAC

Crear un programa Java que ejecute dos hilos. Uno de ellos repetirá de forma indefinida la impresión en la consola de la palabra TIC a intervalos de un segundo. El otro hará lo mismo con la palabra TAC. Define una única clase que permita crear ambos hilos.

¿Se visualizarán los textos TIC y TAC perfectamente alternados: TIC TAC TIC TAC …?

3 Contador de palabras

Crear un programa que reciba a través de sus argumentos una lista de ficheros de texto y cuente cuántas líneas, palabras y caracteres hay en cada fichero, así como en el total entre todos los ficheros. El programa recibirá un argumento adicional que indique si los ficheros se procesarán de forma secuencial o en paralelo usando hilos.

Se considera como palabra cualquier cadena descrita por la expresión `\p{Alpha}+`.

El programa medirá el tiempo que emplea en procesar los ficheros. Usar esa información para comparar la velocidad de proceso de la versión secuencial con la velocidad de proceso de la versión concurrente.

4 Productores/consumidores

Dada la siguiente interfaz que define la funcionalidad de un almacén al que acceden concurrentemente productores y consumidores para almacenar y retirar productos respectivamente:

```
public interface Almacen<T> {

    public void almacenar(T producto);

    public T retirar();

}
```

Escribir un programa que gestione el acceso concurrente de múltiples productores y múltiples consumidores a un almacén con capacidad para *N* productos usando solamente semáforos.

5 Fumadores

Considerar un sistema formado por tres fumadores que se pasan el día liando cigarros para fumarlos inmediatamente.

Para liar un cigarro necesitan tres ingredientes: tabaco, papel y cerillas. Cada fumador dispone de un surtido ilimitado de sólo uno de los tres ingredientes, diferente del que tienen el resto de los fumadores.

Además de los fumadores, hay también un agente que dispone de unas reservas ilimitadas de cada uno de los tres ingredientes. El agente se encarga de poner encima de una mesa dos de los tres ingredientes escogidos de forma aleatoria y esperará a que un fumador, el que tenga el ingrediente que falta, los retire, lie un cigarro y lo fume (todo este proceso le llevará 1 segundo). Cuando el fumador termine de fumar, se repetirá el ciclo.

Simular todo el proceso mediante cuatros hilos, uno para cada fumador y uno para el agente.

6 Banco del parque

Se pretende simular el funcionamiento de un banco en un parque con N plazas disponibles para que se sienten las personas que pasean por el parque según las especificaciones siguientes:

- Todas las personas estarán en el parque desde el comienzo de la simulación e iniciarán su paseo inmediatamente.
- Cada una paseará durante un periodo de tiempo aleatorio entre 1000 y 3000 milisegundos. En este periodo se incluye el paseo y la llegada hasta el banco.
- Pasado el tiempo de paseo intentarán sentarse en el banco para descansar, momento en el que se dará una de las dos situaciones siguientes:
    - El banco tiene plazas libres, en cuyo caso se sentarán y descansarán durante un periodo de tiempo aleatorio entre 100 y 700 milisegundos. Transcurrido ese tiempo, se levantarán dejando libre su plaza.
    - El banco no tiene plazas libres, en cuyo caso esperarán hasta otra persona deje su plaza libre. Este proceso se puede repetir indefinidamente si hay varias personas compitiendo por una plaza.
- Se mostrarán en la consola mensajes que informen para cada persona del comienzo de cada una de sus acciones dentro de la simulación: pasea, espera una plaza en el banco, se sienta o se levanta.
- Pedir, antes de que comience la simulación, el número de plazas del banco y el número de personas en el parque.

7 La cena de los filósofos chinos

Cinco filósofos chinos, **孔夫子**, **楊朱**, **荀子**, **商鞅** y **莊子**, pasan su vida sentados alrededor de una mesa comiendo o pensando. Crear un programa Java que simule el proceso mediante hilos, uno por cada filósofo, para que ejecuten las acciones de pensar, esperar o comer satisfaciendo la exclusión mutua (dos filósofos no pueden emplear el mismo palillo a la vez), además de evitar el interbloqueo y la inanición. A continuación, se detallan las especificaciones básicas que habrá que tener en cuenta para resolver el problema:

- Cada filósofo tiene un plato de arroz, un palillo a la izquierda de su plato y otro a la derecha.
- Cada filósofo comparte los palillos a su izquierda y a su derecha con los filósofos que se sientan a su izquierda y a su derecha respectivamente.
- Cada filósofo se encontrará en una de las situaciones siguientes:
    - Pensando: durante un periodo de tiempo aleatorio en el que no estará usando ninguno de los palillos.
    - Esperando: desde el momento en que decide comer hasta que consiga coger los dos palillos. Sólo podrá coger cada palillo si no lo está usando otro filósofo.
    - Comiendo: durante un periodo de tiempo aleatorio que comenzará inmediatamente después de que haya cogido el segundo palillo. Cuando finalice este periodo de tiempo, dejará de usar los palillos.

Opcional:

Desarrollar la aplicación usando las clases de la librería Swing para mostrar una animación de la simulación con los elementos gráficos que se estimen oportunos, permitiendo la posibilidad de pausarla y reanudarla. A continuación, se propone una forma de representar la escena en cada instante:

- Representar la mesa con un círculo.
- Representar cada plato mediante un círculo relleno de un color distinto al resto, distribuyéndolos uniformemente alrededor de la mesa.
- Cuando un filósofo está usando un palillo, dibujar el palillo del mismo color que su plato.
- Mostrar dentro de cada plato un icono que represente la acción que esté realizando el filósofo en cada instante: pensar, esperar o comer.
- Cuando un palillo no está siendo utilizado por ningún filósofo, se mostrará de un color diferente a cualquiera de los asignados a los platos.

8 Cadena de Montaje

En una cadena de montaje existe un robot encargado de colocar productos de 3 tipos diferentes (1, 2 o 3) en la cadena de montaje. Otros robots, retiran los productos de la cadena de montaje para realizar su empaquetado, teniendo en cuenta que están especializados en un solo tipo de producto (1, 2 o 3), ignorando los que no son de su tipo. Finalmente, se quiere llevar un control del total de productos empaquetados (independientemente de su tipo).

Modelar el sistema descrito utilizando hilos con las siguientes indicaciones:

- Modelar cada robot como un hilo (1 colocador y 3 empaquetadores, uno para cada tipo de producto).
- Los productos son colocados de uno en uno en la cadena, y solamente en posiciones libres (se puede considerar que en la cadena de montaje caben un máximo N de elementos). Si no hay posiciones libres el robot colocador tendrá que esperar hasta que algún producto sea retirado de la cadena.
- Los robots empaquetadores se especializan en un tipo de producto (1, 2 o 3) en tiempo de inicialización.

- Los robots empaquetadores comprueban si hay algún elemento de su tipo en la cadena ignorando los productos que no sean de su tipo. Si hay algún producto de su tipo lo retiran de la cadena (sólo 1 producto cada vez) y la posición queda libre para colocar nuevos productos, en caso contrario se quedan a la espera de que haya nuevos productos.
- Los robots empaquetadores de distinto tipo pueden funcionar a la vez.
- Tanto el colocador como los empaquetadores nunca acaban.
- Cada vez que un robot empaquetador procesa un producto, la cuenta total de productos empaquetados debe aumentar y mostrarse un mensaje por pantalla.

9 El barbero dormilón

Crear una aplicación Java para consola que simule el funcionamiento de una barbería en la que hay un sillón de barbero y una sala de espera con *n* sillas.

En la barbería trabaja un barbero. La simulación de su trabajo, que va a consistir en atender a los clientes afeitándoles la barba (este proceso durará 1 segundo), se realizará creando un hilo para tal efecto. También se ha de simular que el barbero duerme cuando no está atendiendo a un cliente.

A la barbería acudirán varios clientes. La simulación de la llegada de cada cliente y su estancia en ella hasta que se vaya se simulará creando un hilo por cada uno de ellos. Se habrá de tener en cuenta que si un cliente llega a la barbería y la sala de espera está llena, se irá sin ser atendido y ya no volverá.

Si un cliente llega a la peluquería y el barbero está durmiendo, tendrá que despertarlo. Si se encuentra atendiendo a otro cliente, tendrá que esperar su turno. El orden en que serán atendidos los clientes no tendrá por qué ser el mismo que el de llegada.

Durante la ejecución de la aplicación será interesante saber en qué momento entra cada cliente en la peluquería, qué clientes acceden a la sala de espera, qué clientes se van sin ser atendidos y cuándo comienza el peluquero a cortar el pelo a cada cliente.

El número de clientes que acudirán a la peluquería y el número de sillas en la sala de espera serán datos que tendrá que introducir el usuario cuando ejecute el programa.

Cuando la aplicación esté terminada, se ha de hacer una prueba creando una barbería con 5 sillas en la sala de espera y 10 clientes.

10 Globos

Hacer un programa compuesto por las cuatro clases siguientes:

- `HinchaGlobos` que extiende a `Thread`, encargada de hinchar globos.
- `PinchaGlobos` que extiende a `Thread`, encargada de pinchar globos.
- `AmacenGlobos`.
- `Main` (contiene el método `main` desde que se instancia al resto de clases)

En relación a la clase `Globos` (se creará una 1 instancia de esta clase):

- En el almacén podrá contener un máximo de 10 globos.
- Se irán entregando de uno en uno numerándolos desde 1 a 10.

- Una vez entregados se hincharán hasta que estallen o los pinchen.
- Sólo podrá haber 3 globos hinchándose a la vez.
- El globo se entrega con un volumen inicial de 0 y se podrán hinchar hasta alcanzar un volumen máximo de 5. Una vez superado dicho volumen, estallarán
- Los globos pueden ser pinchados mientras se están hinchando.
- Se escribirá un mensaje cada vez que:
    - Se entregue un globo (ej.: GLOBO 5 ENTREGADO A HG3).
    - Se hinche un globo indicando el nuevo volumen (ej.: GLOBO 5 VOLUMEN 5)
    - Estalle un globo por superar el volumen máximo (ej.: GLOBO 5 ESTALLA)
    - Un PG pinche un globo (ej.: GLOBO 5 PINCHADO POR PG3)

En relación a la clase `HinchaGlobos` (se crearán 5 instancias de esta clase):

- Cada instancia se nombrará con HG seguido de un número que la identifique.
- Cada instancia obtendrá un globo de la clase globos y lo hinchará aumentando su volumen en una unidad cada segundo hasta que estalle o sea pinchando
- Si ya hubiera tres globos hinchándose, se esperará hasta que uno se pinche o estalle.
- Cuando un globo estalla o se pincha, volverá a por otro globo hasta que se agoten.

En relación a la clase `PinchaGlobos` (tendremos 5 instancias de esta clase)

- Cada instancia se nombrará con PG seguido de un número que la identifique.
- Intentará pinchar uno de los globos que se está hinchando, elegido de forma aleatoria, cada vez que transcurra un tiempo aleatorio entre 1 y 10 segundos.
- Si no hay globos que pinchar se quedará en espera.
- Dejará de pinchar cuando no queden globos que pinchar.

11 Generador de Secuencias

Crear un programa Java que utilice hilos para mostrar en la consola una secuencia de letras mayúsculas que se repite indefinidamente según las especificaciones siguientes:

- El usuario introducirá por teclado un patrón que represente la secuencia, usando para ello la sintaxis siguiente:

  `<LETRA><REPETICIONES><LETRA><REPETICIONES>...`

  Por ejemplo, el patrón `T3X2K2T5S4` representa la secuencia `TTTXXKKTTTTTSSSS` que el programa repetirá de forma indefinida en la consola:

  `TTTXXKKTTTTTSSSSTTTXXKKTTTTTSSSSTTTXXKKTTTTTSSSS...`

- Cada letra diferente la escribirá un hilo, es decir, si en el patrón hay n letras diferentes, se crearán n hilos. Por ejemplo, en el patrón `T3X2K2T5S4` hay 4 letras diferentes (la letra T que aparece dos veces cuenta sólo como una), por tanto, se crearán 4 hilos para generar la secuencia.

- Todos los hilos se crearán con una única clase de nombre `EscribeLetras` que extienda a la clase `Thread`.
- Se ha de implementar un sistema de control del turno de impresión basado en:
    - El uso de una variable u objeto compartidos que determinen a qué hilo le corresponde el turno.
    - Un protocolo de parada y avance usando sincronización y comunicación de hilos.
- Cada hilo imprimirá en su turno letras a razón de una cada medio segundo.
- Las secuencias se mostrarán siempre con letras mayúsculas, aunque se permitirá introducir el patrón con letras minúsculas.
- El programa finalizará inmediatamente mostrando un error si el usuario introduce un patrón incorrecto. Algunos ejemplos de patrones incorrectos son:

  `A3BC7X1`, `D3-H5`, `7A2B3C`

12 Piscina

Crear un programa Java para resolver el siguiente problema de concurrencia usando semáforos.

Se dispone de una piscina olímpica con 8 calles. En la piscina puede haber dos tipos de usuarios que se identificarán con un nombre:

- Nadadores que ocupan una única calle y que pueden ser hombre, mujer, niño o niña.
- Submarinistas que ocupan dos calles.

Ambos tipos de usuarios se lanzan a la piscina si hay calles libres suficientes y tardan entre 50 y 80 ms en realizar un largo, dejando la calle libre para otro usuario cuando finalizan.

Cuando entra o sale cualquier usuario, se debe de mostrar su nombre y el total de hombres, mujeres, niños, niñas y submarinistas que hay en la piscina a partir de este momento.

Realizar un programa que simule la entrada y salida de 20 usuarios entre nadadores y submarinistas en la piscina.

13 Métodos de ordenación

Crea una aplicación con interfaz gráfica de usuario basada en Swing que permita visualizar en la ventana principal una animación de los métodos de ordenación de inserción directa, selección directa e intercambio directo. El usuario podrá elegir que se ejecuten de forma simultánea los métodos que elija. En cualquier caso, cada método lo ejecutará un hilo creado a tal efecto.