CICLE FORMATIU: CFGS Desenvolupament Aplicacions Multiplataforma DAM

MÒDL PROFESSIONAL: MP09 Programació de serveis i processos

UNITAT FORMATIVA: UF2. Processos i fils

==========

Pràctica 6. Sincronització multifil.

Programarem amb threads i accessos concurrents un sistema de vendes de entrades.

Per a cada espectacle es treuen a la venda 100 entrades. Utilitzarem un vector de 100 posicions per controlar-les. Valors del vector: 0 entrada no venuda, 1 entrada venuda.

El sistema comença posant a zero tot el vector de entrades.

Després crearà 10 threads que simulen 10 compradors que accedeixen desde diferents llocs. Cada thread farà el següent:

Genera un numero de 1 al 5 que son les entrades que volen.
Per cada entrada genera un numero del 1 al 100 que es la entrada que es vol. Si la entrada està venuda generem altre cop el numero de entrada. El sistema espera 1s per simular que l'usuari s'ho està rumiant.
L'usuari compra la entrada, aquesta compra cal que sigui amb accés exclusiu. La entrada es posa com venuda (+1 al valor del vector) si encara es lliure (cal tornà a comprovar, si s`ha venut es perd).

Es recullen els threads i llistem quantes entrades estan a 0, 1 y 2 o mes (si es 2 vol dir que la entrada s'ha venut dos vegades).

Lliura el codi font ben comentat , i un parell d'execucions.

// MAIN //

```
package threads;

public class Main {

	public static void main(String[] args) throws InterruptedException {
		// TODO Auto-generated method stub
		/*
		 * Inicializamos la array con 100 posiciones y sus valores a 0
		 */
		initializeArray();

		/*
		 * 10 hilos distintos de 10 compradores diferentes.
		 */
		Comprador c = new Comprador();
		c.start();
		Comprador c1 = new Comprador();
		c1.start();
		Comprador c2 = new Comprador();
		c2.start();
		Comprador c3 = new Comprador();
		c3.start();
		Comprador c4 = new Comprador();
		c4.start();
		Comprador c5 = new Comprador();
		c5.start();
		Comprador c6 = new Comprador();
		c6.start();
		Comprador c7 = new Comprador();
		c7.start();
		Comprador c8 = new Comprador();
		c8.start();
		Comprador c9 = new Comprador();
		c9.start();

		/*
		 * Estos join son para esperar que todos los hilos acaben y a si poder
imprimir
		 * un resultado final
		 */
		c.join();
		c1.join();
		c2.join();
		c3.join();
		c4.join();
		c5.join();
		c6.join();
		c7.join();
		c8.join();
		c9.join();

		System.out.println(Entrada.entradas);
	}

	/*
	 * Metodo para inicializar la array
	 */
	public static void initializeArray() {
		for (int i = 0; i < 100; i++) {
```

```
                Entrada.entradas.add(0);
            }
            System.out.println(Entrada.entradas);
        }

}
```

// ENTRADA //

```
package threads;

import java.util.ArrayList;
import java.util.List;

public class Entrada extends Thread {

        static List<Integer> entradas = new ArrayList<>(); // Array de 100 posiciones de
las entradas

        /*
         * Metodo que se encarga de comprar las entradas mirando si la posicion de la
         * entrada no esta comprada, es decir, que su valor es = 0 si es igual = 1 no
         * compra la entrada y vuelve a llamar al metodo. con el synchronized podemos
         * sincronizar los hilos que estan llamando a este metodo para que se queden
         * esperando y a si que no nos salgan que hay una misma entrada vendida 2 veces.
         * Es decir que la posicion de una entrada sea = 2
         */
        public synchronized static void buyTickets() {
            int entrada = (int) (Math.random() * 100);
            if (entradas.get(entrada) == 1) {
                buyTickets();

            } else {
                entradas.set(entrada, entradas.get(entrada) + 1);
            }

        }
}
```

// COMPRADOR //

```
package threads;

public class Comprador extends Thread {

        /*
         * Metodo para mirar cuantas entradas va a comprar cada comprador.
         */
        public int inputQuantity() {
            int result = (int) (Math.random() * 5 + 1);
            return result;
        }

        /*
         * El run de los hilos.
         */
        public void run() {

            int quantityTickets = inputQuantity();
            /*
             * Recorremos tantas veces como entradas vaya a comprar.
```

```
		 */
		for (int i = 0; i < quantityTickets; i++) {

			Entrada.buyTickets();
		}
	}
}
```