Práctica 6

Gestión de establecimiento de pizzas

Grado Ingeniería Telemática
GSyC, Universidad Rey Juan Carlos

1. Introducción

El objetivo de esta práctica es familiarizarse con el uso de *Threads* de Java. Un problema fundamental a la hora de usar hebras es que no sabemos cuándo van a ejecutarse. Así que, para que la concurrencia funcione, necesitamos alguna forma de impedir que dos tareas accedan al mismo recurso simultáneamente, al menos en ciertos periodos *críticos*. Como hemos visto, para resolver este problema de colisión de hebras, debemos serializar el acceso a los recursos compartidos mediante métodos `synchronized`.

2. Descripción

Para aprender a usar *Threads* y controlar convenientemente las zonas de exclusión mutua, o zonas *mutex*, vamos a hacer un programa que simule la gestión de un establecimiento de pizzas. Este programa estará compuesto por las siguientes cuatro clases:

1. `HornoPizzas` (HP): encargada de calentar las pizzas.
2. `RepartidorPizzas` (RP): encargada de extraer las pizzas del horno y repartirlas en cualquier momento, cuando considere oportuno.
3. `Pizzas`: stock de pizzas.
4. `Main`: clase de inicio, o código principal (main), desde la que se instancia al resto.

2.1. Clase Pizzas

Tendremos 1 instancia de esta clase. En el establecimiento habrá un máximo de 100 pizzas que se irán introduciendo de una en una enumerándolas desde 1 a 100.

Una vez dentro del horno se podrán calentar, habiendo un máximo de 3 pizzas calentándose a la vez. El tiempo de cocción está estimado de 1 a 5 minutos. Si están más de ese tiempo se queman y, por tanto, no serán comestibles. Obviamente, el tiempo inicial con el que se introduce una pizza es de 0 minutos. También puede ocurrir que las pizzas sean extraídas del horno.

En definitiva, se escribirá un mensaje con el número de pizza cada vez que:

1. Se introduzca una pizza en el horno. Indicaremos el nombre del hilo; por ejemplo: “PIZZA 5 INTRODUCIDA EN HP4”.
2. Transcurra un minuto de cocción, indicando en este caso el nuevo tiempo transcurrido. Por ejemplo: “PIZZA 5 TIEMPO 5”.
3. Se queme una pizza por calentar de más. Ejemplo: “PIZZA 5 QUEMADA”.
4. Un RP extraiga una pizza. Por ejemplo: “PIZZA 5 EXTRAIDA POR RP3”.

Hay que tener en cuenta que, aquellos métodos que puedan afectar al rendimiento oportuno de la aplicación, deberán tratarse convenientemente; esto es, posibles zonas de exclusión mutua, comprobaciones de error, etc.

2.2. Clase HornoPizzas

Tendremos 5 instancias de esta clase. Será la encargada de recibir y calentar las pizzas en orden, a menos que ya no queden. Si ya hubiera tres calentándose, se esperará hasta que se extraiga una.

Estos hornos irán numerados, por ejemplo: "HP3". Un horno calienta una pizza hasta que ésta se extraiga o se queme (en cuyo caso se extrae inmediatamente). Cada segundo equivaldrá a un minuto de cocción de la pizza; de forma que si llega a más de 5, ésta se habrá quemado.

2.3. Clase RepartidorPizzas

Tendremos 5 instancias de esta clase. Esta extraerá, cada `tiempoAleatorio` —de 1 a 10 segundos reales (equivalente a 1-10 minutos ficticios de cocción—, una de las pizzas que se están calentando para repartirlas. Si no hubiera pizzas para sacar, se quedará en espera. Evidentemente, cuando ya no queden pizzas, dejará de repartir.

Igual que en los anteriores casos, hemos de enumerar sus instancias, por ejemplo con "RP2". Los distintos apartados serán comprobados por la impresión de los mensajes detallados en la clase `Pizzas`.

Hay que tener en cuenta que aquellos hilos dormidos (HP dormidos y RP dormidos) deberán ser despertados convenientemente.

3. Implementación

A continuación se muestra el código íntegro de la clase `Main` y los esqueletos de las otras tres clases para su implementación.

3.1. Clase Main

```
public class Main {

  public static void main(String[] args) {
    Pizzas p = new Pizzas(); // instanciamos las pizzas

    for (int i=1;i<=5;i++)
      new HornoPizzas (p,i); // bucle que crea 5 hilos de HornoPizzas, le pasamos
                             // las pizzas e i para saber el número del hilo
    for (int i=1;i<=5;i++)
      new RepartidorPizzas (p,i); // bucle que crea 5 hilos de RepartidorPizzas, le pasamos
                                  // las pizzas e i para saber el número del hilo
  }
}
```

3.2. Clase Pizzas

```
public class Pizzas {
  private int maxPizzHorno=3; // número máximo de pizzas cociñendose a la vez
  private int maxPizzas = 100; // número máximo de pizzas en el establecimiento
  private int maxCoccion = 5; // tiempo máximo de cocción de pizzas
  private int nPizza = 1; // número a dar a la pizza, inicialmente 1
```

```
  private int cociendoAhora = 0; // cuántas pizzas se están cociendo ahora mismo
  private int pizzas[];
  /* El contenido de cada posición de este array será:
  - 0: no se ha metido pizza en horno
  - 1..maxCoccion: tiempo que lleva la pizza cociéndose
  - maxCoccion+1: pizza quemada
  - maxCoccion+2: pizza extraida para ser repartida */

  public Pizzas () { // Constructor de Pizzas
    pizzas = new int[maxPizzas];
    for (int i=0;i<maxPizzas;i++)
      pizzas[i]=0; // ponemos el estado de las pizzas a "sin hornear"
  }

  /* Métodos a implementar:
  cogerPizza(); // método que devuelve la siguiente pizza del stock, si no quedan devuelve -1
  repartirPizza(); // extrae pizza del horno para repartir, si no quedan devuelve true
  cocerPizza (int num); // método para cocer las pizzas del horno
  */
}
```

3.3. Clase HornoPizzas

```
public class HornoPizzas extends Thread {
  // Recordar las pizzas se queman cuando pasan de 5 minutos
  private Pizzas p;
  private int numero;

  public HornoPizzas(Pizzas pi,int pnumero) {
    p=pi;
    numero=pnumero;
    setName("HP"+numero);
    start();
  }

  /* Métodos a implementar:
  run(); // irá llamando a "cogerPizza", hasta que no haya; comprueba si se queman...
  */
}
```

3.4. Clase RepartidorPizzas

```
public class RepartidorPizzas extends Thread {
  private Pizzas p;
  private int numero;

  public RepartidorPizzas(Pizzas pi,int pnumero){
    p=pi;
    numero=pnumero;
    setName("RP"+numero);
    start();
  }

  /* Métodos a implementar:
  run(); // irá llamando a "repartirPizza" en intervalos aleatorios hasta que no haya
  */
```

```
}
```

4. Entrega

Deberás implementar convenientemente las clases cuyos esqueletos y directrices se facilitan —`Pizzas`, `HornoPizzas` y `RepartidorPizzas`— para conseguir la funcionalidad descrita.

Al finalizar, deberás comprimir el proyecto completo bajo el nombre `tulogin-p6.zip` y, como siempre, entregarlo como adjunto a la tarea (Moodle).