**Ejercicio Java de Almundo**

Requerimientos:

Java 1.7

Maven 3.2

Eclipse Mars

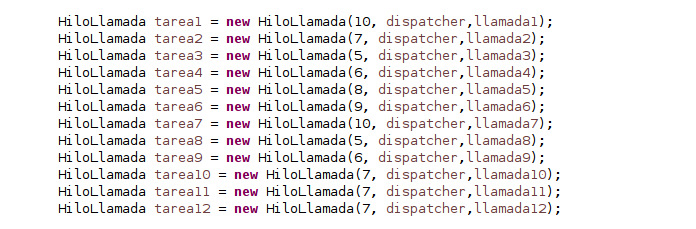
Explicación:

Para la realización del ejercicio se crearon las siguientes clases:

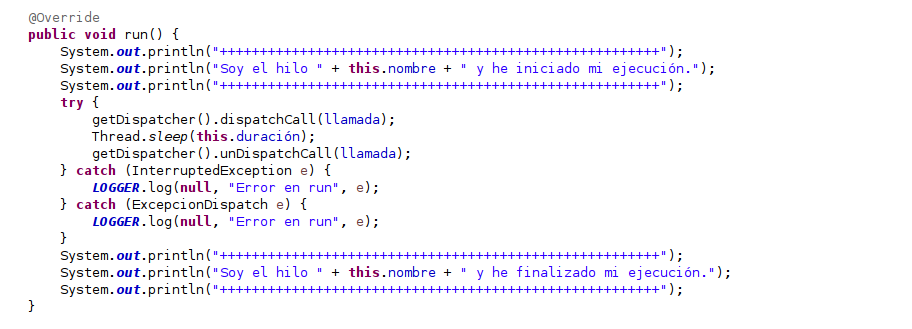
HiloLlamada encargada de llamar a dispatchCall para que le asigne un empleado a esa llamada.

Por cada HiloLlamada debe existir una instancia de Llamada para que le pase por parámetro a dispatchCall.

Cuando se instancia HiloLlamada se le pasa por parámetro la duración, el dispatcher y la llamada de ese hilo.

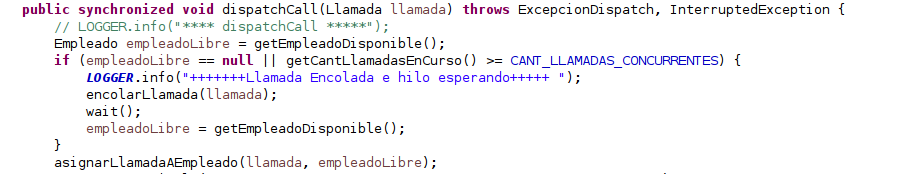


El método run() lo único que hace es llamar al dispatchCall, esperar lo que dura la llamada y llamar al unDispatchCall que desasigna al empleado de la llamada y lo vuelve a poner como disponible.



Clase Dispatcher:

El método dispatchCall se hizo synchronized ya que pueden llegar llamadas de otros hilos y sino puede ocurrir que le asigne el mismo empleado a llamadas diferentes (hilos)

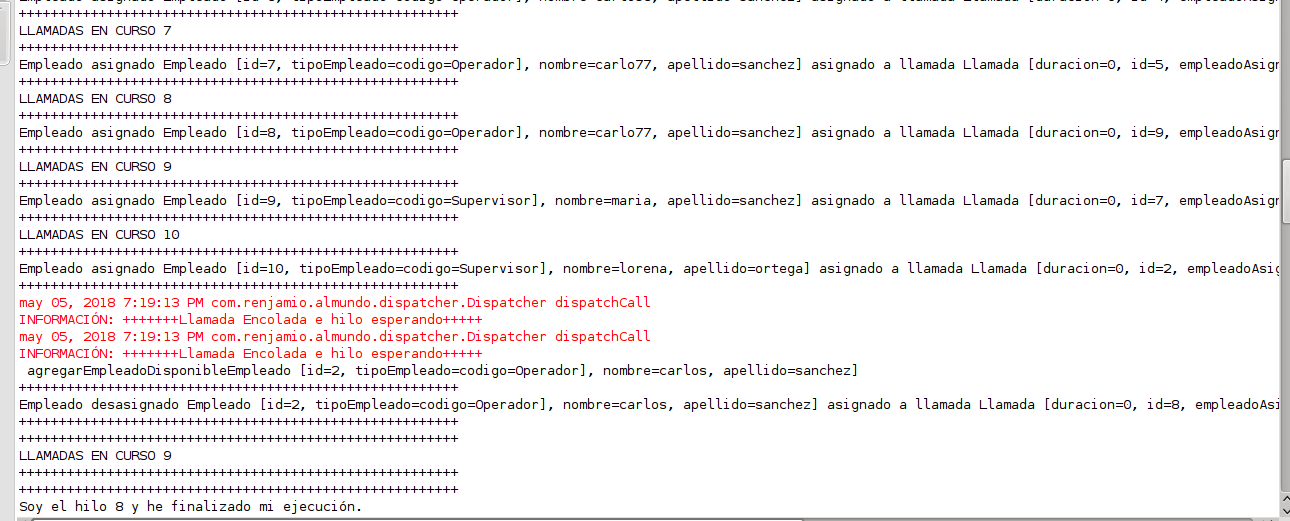


Lo mismo ocurre con la clase unDispatchCall, ya que varios hilos comparten la misma lista de empleados.

Extras:

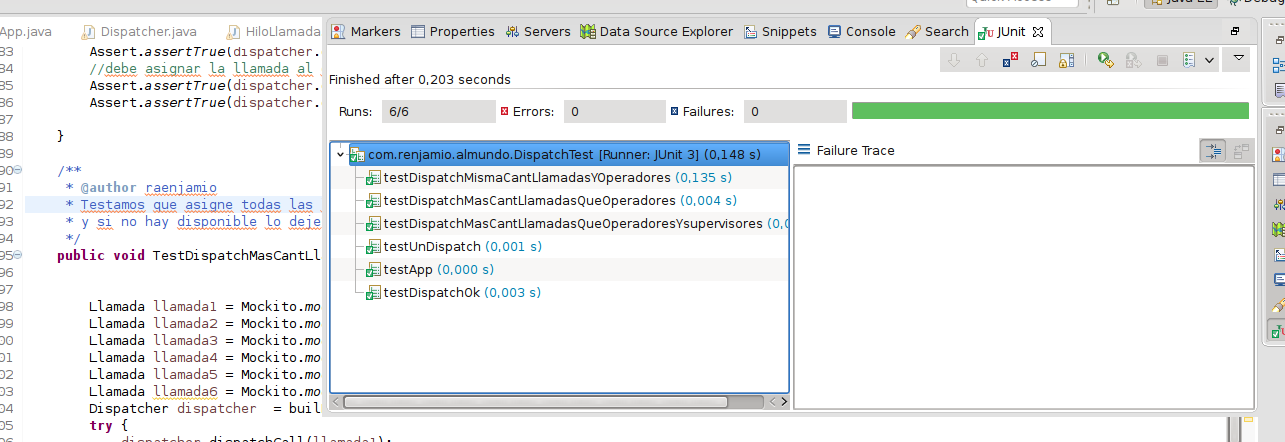
Para los casos en los cuales hay mas de 10 llamadas concurrentes o no hay empleados libres se bloquea al hilo con un wait() hasta que la algún empleado se libere o algún hilo finalice su ejecución, lo cual liberara al hilo con un notify().

En este ejemplo se puede ver como hay 10 llamadas en curso, y hay 2 hilos esperando (llamadas encoladas).



Evidencias Pruebas:

Dispatch Test



Tests de concurrencias:

