



Ejercicio Productor-Consumidor: Las Abejas y el Oso Goloso

Descripción del problema

Simularemos un sistema de **productor-consumidor** con  *abejas recolectoras de néctar* y un  *oso goloso* que llega para para comer miel.

Las **abejas** actúan como **productores** que depositan néctar en una **colmena como recurso compartido**, mientras que el **oso** es el **consumidor** que se despierta cuando hay suficiente miel para comer.

Objetivo

Implementar una simulación concurrente en Java utilizando hilos, sincronización y control de acceso a un recurso compartido (la colmena) por hilos productores y consumidores.

Reglas del sistema

- La **colmena** (buffer compartido) tiene una **capacidad máxima de 30 unidades de miel**.
 - Hay dos **abejas** (hilos productores) que recolectan néctar y depositan 1 unidad de miel en la colmena. Tras depositar, se marchan en busca de más néctar, por lo que tardarán un tiempo aleatorio entre 2 y 5 segundos en recolectar y depositar.
 - Las abejas **no pueden depositar miel** si la colmena está llena.
 - El **oso** (hilo consumidor):
 - Se despierta y solo puede consumir de la colmena **solo cuando hay al menos 3 unidades de miel**.
 - Se **come toda la miel** acumulada y luego se marcha, volviendo pasados unos segundos (entre 2 y 15 segundos) para volver a comer.
 - Repite esto hasta que haya comido **5 veces**.
 - Tras la **quinta comilona**, el oso queda satisfecho y se va a **hibernar**.
 - Cuando el oso hiberna, las abejas **se retiran tras rellenar la colmena** y el programa **termina ordenadamente**.
 - El acceso a la colmena debe estar correctamente **sincronizado** usando `synchronized`, `wait()` y `notifyAll()`.
-

Requisitos técnicos

- Utilizar clases que implementen `Thread` o `Runnable` para modelar a las abejas y al oso.

- Usar métodos sincronizados para controlar el acceso al recurso compartido (Colmena).
- Implementar correctamente las señales de espera y notificación entre hilos (wait(), notifyAll()).
- Mostrar mensajes en consola que describan la actividad, por ejemplo:
 - 🐝 Abeja-1 trajo néctar. Total miel: 7
 - 🐻 Comilona #3. Miel antes: 8
 - 🐻 El oso está lleno. ¡Hora de hibernar!

💡 Posible salida por consola

```

--- Abeja-3 trajo néctar. Miel en el panal: 10
--- Abeja-3 trajo néctar. Miel en el panal: 11
>>> Comilona del oso #3. Miel en la panza: 40
--- Abeja-2 trajo néctar. Miel en el panal: 1
--- Abeja-3 trajo néctar. Miel en el panal: 2
--- Abeja-1 trajo néctar. Miel en el panal: 3
--- Abeja-2 trajo néctar. Miel en el panal: 4
--- Abeja-1 trajo néctar. Miel en el panal: 5
--- Abeja-3 trajo néctar. Miel en el panal: 6
--- Abeja-2 trajo néctar. Miel en el panal: 7
--- Abeja-1 trajo néctar. Miel en el panal: 8
--- Abeja-3 trajo néctar. Miel en el panal: 9
>>> Comilona del oso #4. Miel en la panza: 49
--- Abeja-2 trajo néctar. Miel en el panal: 1
--- Abeja-1 trajo néctar. Miel en el panal: 2
--- Abeja-3 trajo néctar. Miel en el panal: 3
--- Abeja-1 trajo néctar. Miel en el panal: 4
>>> Comilona del oso #5. Miel en la panza: 53
>>> Oso lleno hasta arriba se marcha a hibernar una larga temporada.
Abeja-3 ve que el panal esta cerrado y se marcha.
Abeja-2 ve que el panal esta cerrado y se marcha.
```

Abeja-1 ve que el panal esta cerrado y se marcha.

Panal de abejas cerrado. Fin del dia.

BUILD SUCCESSFUL (total time: 42 seconds)



Objetivos de aprendizaje

- Comprender y aplicar el patrón **productor-consumidor**.
- Usar correctamente la **sincronización de hilos** en Java (wait() / notifyAll()).
- Diseñar una solución concurrente que **finalice de forma controlada**.
- Manejar múltiples hilos con condiciones de parada bien definidas.