Productor-Consumidor

1. Describe los cambios que has realizado sobre la clase Buffer vista anteriormente, indicando qué objetos condición has usado y el propósito de cada uno.

En la clase Buffer tenemos:

- Una variable numSlots que indica el tamaño del buffer.
- Una variable cont que cuenta el número de elementos que tiene el buffer
- Un vector buffer que almacena los elementos que van llegando del productor
- Dos condiciones:
  - puede_leer:
    - Hacemos un await() cuando la variable cont es 0, es decir, cuando el buffer no tiene ningún elemento y por lo tanto no podemos leer ningún valor.
    - Hacemos un signal() cuando almacenamos un valor en el buffer, con lo cual ya podemos leer un valor.
  - puede_escribir:
    - Hacemos un await() cuando la variable cont es igual a numSlots, es decir, cuando el buffer está lleno y por lo tanto no podemos escribir más valores ahí.
    - Hacemos un signal() cuando se consume un valor del buffer, ya que se ha leído y podemos almacenar otro elemento en el vector.

En los métodos públicos *depositar* y *extraer* hemos quitado *synchronized* de la cabecera. Dentro de los métodos hemos agregado los métodos del monitor enter() y leave() al principio y al final de ellos, respectivamente. Hemos cambiado los notifyAll() por signal() y wait() por await(). Por último, los bucles *while* se cambian por un *if* ya que los monitores estilo Hoare implementan una semántica SU (señalar y espera urgente).

2. Incluye un listado de la salida del programa (para 2 productores, 2 consumidores, un buffer con tamaño 3 y 5 iteraciones por hebra).

Ejecutamos: java ProductorConsumidor 2 5 2 5 3

```
productor 0, produciendo 0.0
consumidor 0, consumiendo 0.0
productor 0, produciendo 1.0
productor 1, produciendo 100.0
consumidor 1, consumiendo 1.0
productor 0, produciendo 2.0
consumidor 0, consumiendo 100.0
productor 1, produciendo 101.0
productor 0, produciendo 3.0
consumidor 0, consumiendo 101.0
consumidor 1, consumiendo 2.0
consumidor 0, consumiendo 3.0
productor 0, produciendo 4.0
productor 1, produciendo 102.0
consumidor 1, consumiendo 4.0
consumidor 0, consumiendo 102.0
productor 1, produciendo 103.0
consumidor 1, consumiendo 103.0
productor 1, produciendo 104.0
consumidor 1, consumiendo 104.0
```

Fumadores

1. Describe qué objetos condición has usado y el propósito de cada uno.

Cuatro condiciones:

- estanquero:
  - Hacemos un await() cuando el estanquero ha puesto un ingrediente sobre el mostrador pero el fumador correspondiente aún no lo ha recogido.
  - Hacemos un signal() cuando el fumador correspondiente recoge su ingrediente y comienza a fumar.
- fumador: (es un array de 3 condiciones porque hay 3 fumadores y tienen la misma función)
  - Hacemos un await() cuando el ingrediente que hay sobre la mesa no es el que le corresponde al fumador.
  - Hacemos un signal() cuando el estanquero pone sobre el mostrador el ingrediente que le falta al fumador correspondiente.

2. Incluye un listado de la salida del programa

```
El estanquero pone tabaco
El fumador 1 comienza a fumar
El estanquero pone tabaco
El fumador 1 termina de fumar
El fumador 1 comienza a fumar
El estanquero pone papel
El fumador 0 comienza a fumar
El estanquero pone tabaco
El fumador 0 termina de fumar
El fumador 1 termina de fumar
El fumador 1 comienza a fumar
El estanquero pone papel
El fumador 0 comienza a fumar
El estanquero pone papel
El fumador 0 termina de fumar
El fumador 0 comienza a fumar
El estanquero pone cerillas
El fumador 2 comienza a fumar
El estanquero pone papel
El fumador 1 termina de fumar
El fumador 2 termina de fumar
El fumador 0 termina de fumar
El fumador 0 comienza a fumar
El estanquero pone cerillas
El fumador 2 comienza a fumar
El estanquero pone cerillas
El fumador 2 termina de fumar
El fumador 2 comienza a fumar
El estanquero pone cerillas
El fumador 0 termina de fumar
```

Barbero Durmiente

1. Descripción de los objetos condición usados y su propósito.

Tres condiciones:

- cliente_sala_espera:
  - Hacemos un await() cuando el cliente entra a la barbería pero el barbero está ocupado y no puede atenderle.
  - Hacemos un signal() cuando el barbero termina de pelar a un cliente y tiene que llamar al siguiente.
- barbero:
  - Hacemos un await() cuando no hay ningún cliente en la barbería, con lo cual el barbero se va a dormir.
  - Hacemos un signal() cuando la barbería no tiene ningún cliente y entra el primero. Como el barbero estaba dormido, el cliente lo despierta.
- cliente_ocupado:
  - Hacemos un await() cuando el barbero está pelando al cliente.
  - Hacemos un signal() cuando el barbero avisa al cliente de que ya ha terminado su trabajo, con lo cual ya puede salir de la barbería.

2. Listado de la salida del programa

```
No hay ningún cliente esperando y el barbero se va a dormir
Un cliente despierta al barbero
El barbero comienza a pelar al cliente
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 1 clientes esperando.
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 2 clientes esperando.
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 3 clientes esperando.
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 4 clientes esperando.
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 3 clientes esperando.
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 2 clientes esperando.
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 2 clientes esperando.
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 3 clientes esperando.
Un cliente pasa a la sala de espera porque el barbero está ocupado. Hay 4 clientes esperando.
El barbero termina de pelar al cliente y el cliente sale por la puerta
El barbero llama a un cliente
El barbero comienza a pelar al cliente
El barbero termina de pelar al cliente y el cliente sale por la puerta
```