COLECCIONES CONCURRENTES

CONCURRENCIA EN COLECCIONES

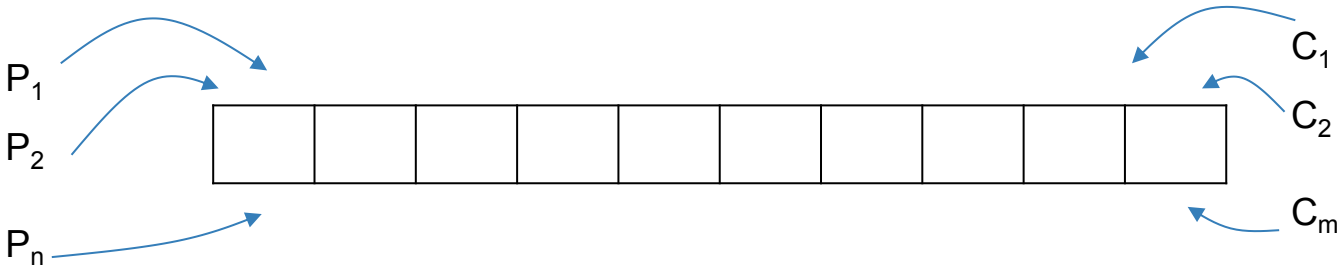
Las colecciones no están exentas de problemas de concurrencia.

- Acceso en modo lectura/escritura de dos o más hilos de ejecución diferente (productor/consumidor)
- Operaciones de ordenación
- …

PRODUCTOR/CONSUMIDOR

Problema clásico en concurrencia

- Colección compartida de tamaño finito.
- Varios hilos que producen datos
- Varios hilos que consumen datos
- Problemas de condición de carrera (colección vacía, colección llena…)
- **Requiere de espera y notificación.**



WAIT Y NOTIFY

- Podemos poner en espera a un hilo, hasta que se cumpla una determinada condición.
- Posteriormente, podemos notificarle que ya puede continuar con su ejecución, ya que la condición **bloqueante** ha dejado de cumplirse.
- *Ejemplo: si la colección se llena, los productores deben esperar a que los consumidores tomen elementos, para poder seguir produciendo.*

COLECCIONES CONCURRENTES

Nos evitan *reinventar* la rueda, programando soluciones como la del productor – consumidor.

- ***BlockingQueue*: estructura FIFO que bloquea si la cola se queda llena o vacía.**
- *ConcurrentMap*: Map con operaciones atómicas.
- *ConcurrentNavigableMap*: NaviagleMap con búsquedas aproximadas.

BLOCKINGQUEUE

Estructura FIFO: *first-in-first-out*. Conocida como cola.
Diferentes implementaciones.

- *ArrayBlockingQueue*: capacidad fija
- *LinkedBlockingQueue*: capacidad *Integer.MAX_VALUE*.
- *PriorityBlockingQueue*: ordena según prioridad.
- …