



Ejercicios 5. MONITORES (III)

A partir del proyecto “PC Ejercicios Enunciado 5. Monitores (III)” de la plataforma, añade todo el código necesario en la clase *AccesoAlmacen* usando **cerrojos y condiciones** para satisfacer los requisitos de la solución del siguiente problema:

En una fábrica hay *NumMaquinas* máquinas que fabrican un cierto tipo de piezas y las depositan en un almacén que tiene capacidad para albergar un máximo de *MaxPiezas* piezas en cada instante. Además, en la fábrica hay *NumRobots* robots que recogen las piezas del almacén y las transportan a un determinado destino.

Se desea controlar el funcionamiento de la fábrica mediante un conjunto de procesos cuyo ciclo de vida es el siguiente:

```
PROCESS TYPE TMaquina;  
BEGIN  
  REPEAT  
    Fabricar;  
    Depositar;  
  FOREVER  
END;
```

```
PROCESS TYPE TRobot;  
BEGIN  
  REPEAT  
    Recoger;  
    Transportar  
  FOREVER  
END;
```



SE PIDE:

Codificar lo necesario en la clase *AccesoAlmacen* para sincronizar el acceso de las máquinas y los robots al almacén de piezas, de forma que se cumplan los siguientes requisitos:

- Las **máquinas** deben poder fabricar piezas concurrentemente con el resto de actividades de los demás procesos.
- Las **máquinas** deben poder depositar piezas en el almacén concurrentemente.
- Una **máquina** no puede depositar una pieza en el almacén si el almacén está lleno.
- Una **máquina** no puede depositar una pieza en el almacén si hay un robot recogiendo una pieza.
- Los **robots** deben poder transportar piezas concurrentemente con el resto de actividades de los demás procesos.
- Los **robots** deben recoger piezas del almacén bajo exclusión mutua.
- Un **robot** no puede recoger una pieza del almacén si el almacén está vacío.
- Un **robot** no puede recoger una pieza del almacén si hay máquinas depositando piezas.
- Cuando un **robot** termina de recoger una pieza del almacén tienen prioridad para acceder al almacén los robots que esperan.



ALGORITMO:

```
class AccesoAlmacen {

    private final int MAX_PIEZAS;
    private int numPiezas;
    private int maquinasEsperando;
    private int maquinasDepositando;
    private int robotsEsperando;
    private boolean robotRecogiendo;

    public AccesoAlmacen(int maxPiezas) {
        MAX_PIEZAS = maxPiezas;
        numPiezas = 0;
        maquinasEsperando = 0;
        maquinasDepositando = 0;
        robotsEsperando = 0;
        robotRecogiendo = false;
    }

    public void inicioDepositar() throws InterruptedException {
        <Si hay un robot recogiendo o el almacen esta lleno, entonces esperar>
        <Hay una maquina mas depositando>
        <Hay una pieza mas en el almacen>
    }

    public void finDepositar() throws InterruptedException {
        <Hay una maquina menos depositando>
        <Si no quedan maquinas depositando, entonces liberar a un robot>
    }

    public void inicioRecoger() throws InterruptedException {
        <Si hay un robot recogiendo o hay maquinas depositando o el almacen
        esta vacio, entonces esperar>
        <Hay un robot recogiendo>
        <Hay una pieza menos en el almacen>
    }

    public void finRecoger() throws InterruptedException {
        <No hay un robot recogiendo>
        <Si hay robots esperando y quedan piezas, liberar a un robot>
        <En otro caso, mientras haya maquinas esperando y queden huecos,
        liberar a una maquina>
    }
}
```