Laboratorio 4: Pilas y Colas

Importar el proyecto Lab4.

Ejercicio 1.

En un centro de cálculo habitualmente se utilizan las impresoras 0, 1, 2 y 3 para imprimir documentos. Sin embargo, cuando hay algún problema en el suministro eléctrico, el sistema de seguridad enciende la impresora 4 y envía a dicha impresora todos los trabajos que están a la espera de ser impresos en las demás impresoras.

Más precisamente, durante un problema de suministro se funciona de la siguiente manera:

- Los trabajos que hubiera en las impresoras 0 a 3 se envían a la impresora 4, comenzando por la impresora 0 y respetando el orden en el que se encontraban en cada impresora.
- Mientras no se arregle el problema, los trabajos que llegan a las impresoras 0 a 3 se reenvían a la impresora 4.
- Una vez arreglado el problema, las impresoras 0 a 3 retoman su forma de trabajar habitual. Los trabajos que hubiera en la impresora 4 se quedan allí hasta ser impresos.

Debéis completar el código de la clase *CentroDeCalculo*:

(a) Crear la estructura necesaria para representar las impresoras

(b) Completar el método *simularEventos(String nomFich)*. Este método debe leer una serie de eventos del fichero que se le pasa como parámetro, simularlos y sacar por pantalla cuáles son los trabajos que quedan por imprimir en cada impresora una vez transcurridos todos los eventos.

Existen cuatro tipos de eventos:

- `S`: para solicitar una impresión. La línea tendrá la forma '`S numImpresora nomTrabajo`', donde `nomTrabajo` indica el nombre archivo a imprimir (supondremos que no tiene espacios) y `numImpresora` el número de la impresora a la que se envía.

- `I`: para imprimir el primer trabajo que esté pendiente en una impresora. La línea tendrá la forma '`I numImpresora`'.
- `P`: para indicar el inicio de un problema
- `F`: para indicar el fin de un problema

Ejemplo:

eventos1.txt

```
S   0   mate.pdf
S   1   eda.pdf
I   0
S   0   ia.txt
S   3   cierre.docx
I   3
S   3   pb.pdf
P
I   4
S   2   informe.odt
F
S   0   calculo.ods
```

Salida:

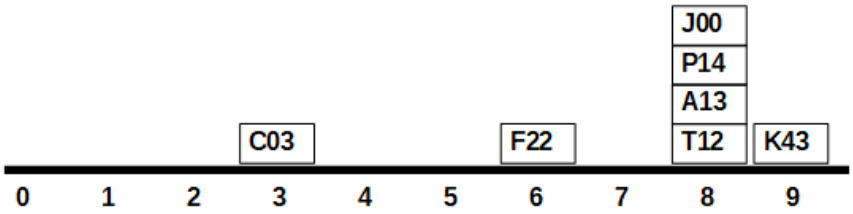
```
Impresora 0: <calculo.ods  >
Impresora 1: <>
Impresora 2: <>
Impresora 3: <>
Impresora 4: <eda.pdf  pb.pdf  informe.odt >
```

NOTA: para extraer los componentes de una línea del fichero ignorando los espacios, podéis utilizar split de la siguiente manera:

```
datos =  linea.split("\\s+");
```

Ejercicio 2.

El puerto de Pasaia dispone de una grúa para descargar contenedores que vienen en camiones, almacenarlos temporalmente en el muelle y cargarlos en barcos.



Concretamente, el muelle dispone de 10 parcelas para almacenar contenedores. La parcela 0 es especial, su uso se destina a dejar momentáneamente contenedores de otras parcelas cuando hay que desplazarlos para sacar otro contenedor que está por debajo. Una vez se ha conseguido sacar el contenedor, todos los que han sido situados en la parcela 0 se vuelven a colocar en su parcela original. Consideraremos además que cada parcela como máximo podrá tener 4 contenedores en todo momento.

Queremos simular el funcionamiento del puerto. Para ello tenemos una cola de Movimientos que indican las descargas y cargas de contenedores que se llevan a cabo. Un movimiento se representa así:

```
public class Movimiento {

	private char tipo; //D: descarga, C: carga

	private String contenedor;//Código del contenedor

	private int parcela; //Parcela donde se deja o de donde se coge el

                        // contenedor

	//Suponemos que existen los getters y setters

}
```

En los movimientos de tipo **descarga** se depositará el contenedor correspondiente en la parcela indicada por 'parcela'. Si dicha parcela ya tuviera 4 contenedores, el movimiento vuelve a la cola para ser procesado más adelante.

En los movimientos de tipo **carga** se retirará el contenedor correspondiente de la parcela indicada. Si el contenedor no está en la cima de dicha parcela, como se ha

dicho anteriormente, se utilizará la parcela 0 para dejar los contenedores que estén por encima, y tras extraer el contenedor buscado se volverán a dejar en la parcela original.

En la clase Puerto se pide:

(a) Crear la estructura necesaria para representar las parcelas

(b) Completar el método *simularMovimientos(Queue<Movimiento> movimientos)*. Este método simula los movimientos que hace la grúa para desplazar los contenedores, actualizando el contenido de las parcelas. Además deberá escribir por pantalla cada movimiento efectuado.

Ejemplo

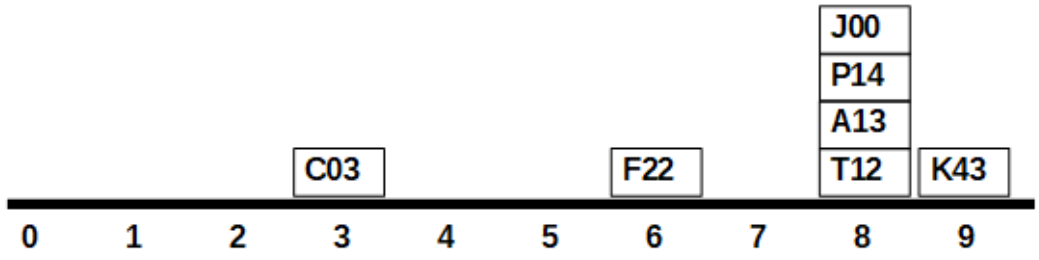
Movimientos:

```
D C03 3
D F22 6
D T12 8
D A13 8
D P14 8
D K43 9
D J00 8
D J11 8
C K43 9
C A13 8
```

Salida:

```
Descargar C03 en la parcela 3.
Descargar F22 en la parcela 6.
Descargar T12 en la parcela 8.
Descargar A13 en la parcela 8.
Descargar P14 en la parcela 8.
Descargar K43 en la parcela 9.
Descargar J00 en la parcela 8. (*)
La parcela 8 está llena. J11 vuelve a la cola. (**)
Cargar K43 desde la parcela 9.
Mover J00 de parcela 8 a parcela 0. (***)
Mover P14 de parcela 8 a parcela 0.
Cargar A13 desde la parcela 8.
Mover P14 de parcela 0 a parcela 8.
Mover J00 de parcela 0 a parcela 8.
Descargar J11 en la parcela 8. (****)
```

(*) Una vez llegado a este punto la situación de las parcelas es:



(**) Porque la parcela 8 ya tiene 4 contenedores.

(***) El movimiento a realizar era "C A13 8". Para poder sacar el contenedor A13 de la parcela 8, primero hay que depositar los que están por encima suyo (J00, P14) en la parcela 0, y después de sacar A13, volver a dejar esos dos contenedores donde estaban.

(****) Se descarga el contenedor que no se había podido descargar antes por estar llena la parcela.