Programación multihilo

Logística de almacén

Índice:

Enunciado

En un centro logístico, dos vehículos operan simultáneamente en una zona de transferencia:

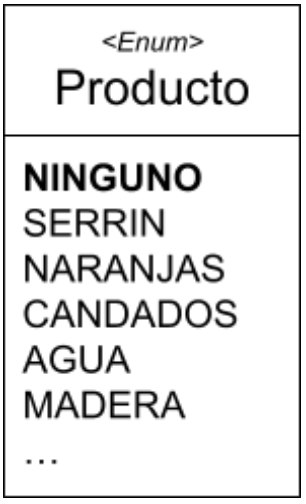
- **`VehiculoDescarga`**: un camión que lleva la mercancía del almacén y la descarga en el punto de intercambio, cambiando su contenedor de mercancía por otro contenedor vacío para seguir cargando.
- **`VehiculoCarga`**: una furgoneta que lleva contenedores vacíos. Cuando entrega un contenedor vacío, debe cargar un contenedor lleno para llevarlo al punto de distribución.

Ambos vehículos se encuentran en un **punto de intercambio**.

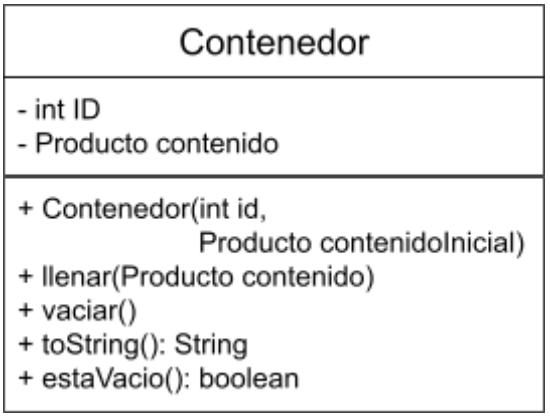
Usa la clase **`Exchanger`** para la resolución de este ejercicio.

Implementación general

Implementa un enumerado que indique las direcciones de los coches:



También la interfaz **`Contenedor`**:



Donde el **ID** es el atributo que identifica al contenedor y el **contenido** inicialmente será **NINGUNO**.

El constructor establecerá el estado del contenedor.

`llenar(Producto contenido)` llenará el contenido del contenedor con el producto introducido por parámetro.

`vaciar()` vaciará el contenedor, dejando el contenido a NINGUNO.

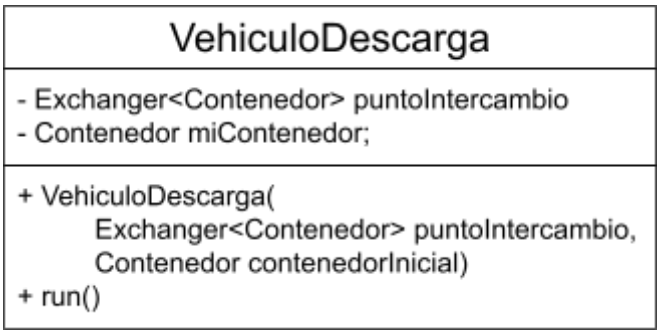
`toString()` devolverá un mensaje como el siguiente:

```
Contenedor <id> en estos momentos contiene <contenido>
```

`estaVacio()` indicará si el contenedor tiene algún producto en su interior o NINGUNO.

VehiculoDescarga

Crear la siguiente clase:



Este vehículo, que será un hilo, realizará las siguientes acciones:

1. Llenará su contenedor con un producto y mostrará un mensaje por consola que indique:

   ```
   <Nombre del vehículo> . Ciclo <i> Contenedor preparado para
   INTERCAMBIAR: <contenedor>.
   ```

   Donde contenedor será el del vehículo.

2. Mostrará el mensaje:

   ```
   <Nombre del vehículo>: esperando punto de encuentro para intercambiar
   por un contenedor vacío..."
   ```

3. Esperará hasta que el otro vehículo deje un contenedor vacío.

4. Cuando obtenga el nuevo contenedor vacío, mostrará:

   ```
   <Nombre del vehículo>: se ha hecho el intercambio. Contenedor
   recibido: <contenedor vacío>
   ```

5. El contenedor de este vehículo será ahora el que ha recibido en el intercambio.

Si el contenedor recibido está vacío, mostrará:

```
<Nombre del vehículo>: contenedor vacío. Listo para el próximo ciclo.
```

En caso contrario mostrará:

```
<Nombre del vehículo>: ERROR! El contenedor recibido no estaba vacío.
```

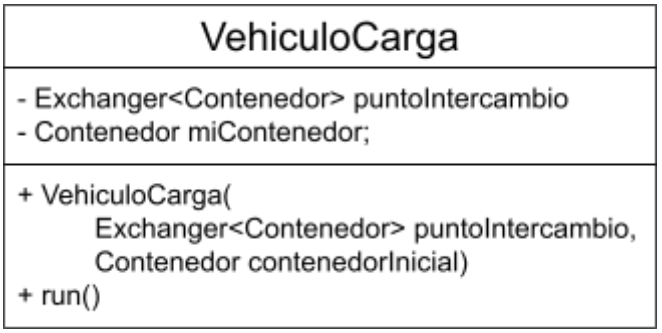
Todos estos pasos se realizarán en un bucle **entre 3 y 5 veces**, donde **cada iteración representa un ciclo**. Además, entre una iteración y otra **hay que esperar 3 segundos**.

Cuando el hilo acabe su ejecución, debe mostrarse el mensaje:

```
<Nombre del vehículo>: trabajo finalizado!
```

VehiculoCarga

Crear la siguiente clase:



Este hilo ejecutará el mismo número de iteraciones que el vehículo anterior.

En este caso:

1. Preparará el contenedor vacío, mostrando el mensaje:

   ```
   <Nombre del vehículo> . Ciclo <i> Contenedor para INTERCAMBIAR:
   <contenedor>.
   ```

   Donde contenedor será el del vehículo.

2. Esperará para realizar el intercambio, mostrando el mensaje:

   ```
   <Nombre del vehículo> . Esperando punto de encuentro para intercambiar
   por un contenedor lleno...
   ```

3. Esperará hasta que el otro vehículo también llame al punto de intercambio y deje su contenedor lleno de mercancía.

4. Cuando obtenga el nuevo contenedor con mercancía, mostrará:

   ```
   <Nombre del vehículo>: se ha hecho el intercambio.

   Contenedor con mercancía: <contenedor lleno>
   ```

5. El contenedor de este vehículo será ahora el que ha recibido en el intercambio.

   Si el contenedor recibido está vacío, mostrará:

   ```
   <Nombre del vehículo>: ERROR! El contenedor recibido está vacío.
   ```

   En caso contrario mostrará:

   ```
   <Nombre del vehículo>: procesando contenedor lleno... Vaciando.
   ```

   Después **esperará un segundo** y lo vaciará. Al hacerlo mostrará lo siguiente:

   ```
   <Nombre del vehículo>: contenedor vaciado. Listo para el próximo
   intercambio.
   ```

Main

Crea un `Main.java` donde:

- Se debe crear el punto de intercambio.
- Se debe crear el contenedor para VehiculoDescarga y otro para VehiculoCarga, ambos inicialmente vacíos.
- Mostrar el siguiente mensaje por pantalla antes de empezar con el procesamiento:

  ```
  --- INICIO SIMULACIÓN LOGÍSTICA DE CONTENEDORES ---
  Vehículo de descarga comienza con: <su contenedor inicial>
  Vehículo de carga comienza con: <su contenedor inicial>
  ------------------------------------------------
  ```

- Se deben crear e iniciar los hilos de ambos vehículos y lanzarlos.

  El hilo principal esperará a que ambos hilos acaben.

- Para mostrar la finalización del programa se mostrará lo siguiente:

  ```
  ------------------------------------------------------------
  Todos los vehículos han completado sus ciclos. Fin del programa.
  ```

Lanzar la ejecución y ver el resultado por consola.

Entrega

Adjunta en la entrega tanto el **código** como un **documento** (en formato Google Documentos) que explique la solución.