UNIVERSIDAD DE MÁLAGA | uma.es

Dpto. de Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

Examen parcial Java 2023

Descripción del Sistema

La empresa Paket S.L. tiene un servicio de entrega en ciudad rápido y con garantías, siempre y cuando te acepten la entrega. Tienen una flota de ***NC*** conductores. Cada conductor tiene una **capacidad máxima** de reparto en kilos, asumimos que la dimensión del paquete no es relevante en esta solución. Esa **capacidad máxima** limita el número de paquetes que pueden llevar. Como la velocidad de entrega es importante, Paket S.L. establece que un conductor sale a reparto cuando tiene el 70% o más de su **capacidad máxima** cargada.

Esta foto de Autor desconocido está bajo licencia CC BY

Un cliente llega con su paquete a la empresa, **si hay algún conductor que esté cargando** y el peso de su paquete **no supera la capacidad máxima** del conductor, el paquete se le entrega al conductor y **queda a la espera**, hasta que el conductor le notifica que se ha entregado. Si un paquete completa la carga de un conductor por encima del 70%, el conductor está listo para salir a reparto. Si no hay conductores disponibles, o el paquete que entrega el cliente es muy pesado en ese momento para el transporte, **el cliente se da un paseo y vuelve a intentarlo más tarde**.

Cuando el conductor comienza el reparto, **se van repartiendo los paquetes en el mismo orden el que se cargaron**, siendo el primero en ser cargado el primero en ser repartido.

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura

Paket S.L. tiene una política estricta de horarios**, cuando se cierra el negocio, no se admiten más paquetes**, y los conductores, si tienen algún paquete cargado**, salen en un último reparto**.

Para modelar este comportamiento, se proporciona un esqueleto con las siguientes clases:

- **EstadoPaquete(ya implementado):** Enumerado que lista los estados en los que se puede encontrar un paquete tras intentar entregarlo:
  - `EstadoPaquete.ENTREGADO:` Paquete que ha sido entregado.
  - `EstadoPaquete.REINTENTAR:` Paquete que no ha podido ser cargado en ningún conductor, se debe volver a intentar su deposito en la empresa.
  - `EstadoPaquete.SINSERVICIO:` Paquete que se ha intentado depositar cuando la empresa ya ha cerrado.
- **Conductor (ya implementado)**. Es una hebra que modela el comportamiento de un **Conductor**. El método *run()* de esta clase llama al método *puedoTransportar* de la clase **Gestor** cuando se inicia, e indica su id y su capacidad máxima. Tras esto, de forma iterativa, mientras no esté cerrado el gestor, primero llama a *cargarPaquetes*, y luego a *repartir*.
- **Cliente (ya implementado):** Es una hebra que simula la entrega, y espera, en el reparto de un paquete. Primero simula un peso de paquete entre 10 y 15 kilos y luego itera llamando a *entregaPaquete* de la clase **Gestor**. Si no se puede entregar a un conductor, esperará un tiempo y luego volverá a entregarlo.
- **Gestor**. Esta clase modela el recurso compartido con los siguiente métodos que deben ser implementados:
  - `public Gestor(int numConductores, float capacidad):` Constructor que inicializa las variables que se necesiten haciendo uso del número de conductores que tiene ese día la empresa y la capacidad a partir de la cual se considera que el conductor puede salir a reparto. La capacidad por el peso máximo que puede transportar un conductor, indica el umbral a partir del cual un conductor sale a reparto. Por ejemplo, si el conductor puede transportar 100kg y la capacidad mínima que indica la empresa es 0.7 (70%), si el peso de los paquetes que ha cargado superan o son iguales 70kg (100*0.7), puede salir a reparto.
  - `public boolean entregaPaquete(int id, int peso):` Método que implementa la entrega del paquete con identificación *id*, y con peso *peso*. Se debe asignar el paquete a uno de los conductores que pueda llevarlo, y esperar a que ese conductor notifique que ha entregado el paquete, tras esto devuelve `estadoPaquete.ENTREGADO`. Si no hay conductores disponibles o no pueden transportar un paquete tan pesado, el método devuelve `EstadoPaquete.REINTENTAR.`
    Si el negocio está cerrado, devuelve `EstadoPaquete.SINSERVICIO`.
  - `public void puedoTransportar(int id, int peso):` Inicializa el conductor con identificación *id*, y cuya capacidad máxima es *peso*.
  - `public void cargarPaquetes(int id):` Método en el que espera el conductor mientras se cargan sus paquetes. Cuando se termina de cargar, "sale de las instalaciones".
  - `public void repartir(int id):` En este método el conductor reparte y notifica cuando terminan las entregas.
  - `public void fin():` Este método es el encargado de cerrar las instalaciones, no permitiendo que se acepten más paquetes para

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura

enviar. Además, se encarga de que los posibles paquetes que ya estén cargados lleguen a reparto.

- `public boolean estaCerrado()`: Método que devuelve `true` si las instalaciones de reparto están cerradas y `false` si no.

Nota que el sistema que se plantea tiene varias condiciones de sincronización:

1. **CS-Conductor**: El **Conductor** debe esperar a repartir hasta que han cargado paquetes suficientes (peso cargado superior o igual a la capacidad por el peso máximo que puede transportar). Excepto si las instalaciones han cerrado y le quedan paquetes cargados, en ese caso debe salir a repartir **independientemente** de si llega a la capacidad mínima.
2. **CS-Cliente**: Un cliente, que ha depositado un paquete para su entrega, debe esperar a que esté sea repartido.

Desarrolla la implementación haciendo uso de semáforos binarios, puedes usar arrays de semáforos y la opción de que sean justos si lo necesitas. Ojo, que el número de conductores lo conoce el gestor sólo en tiempo de ejecución, es decir, cuando llaman al constructor de la clase.

A continuación se muestra una posible ejecución con 3 Conductores y 40 paquetes.

```
Nuevo Conductor 0, puede transportar 88 kilos.
Nuevo Conductor 1, puede transportar 91 kilos.
Nuevo Conductor 2, puede transportar 93 kilos.
Paquete 35 con peso 11 reparte conductor 0, tiene carga disp.77
Paquete 34 con peso 10 reparte conductor 0, tiene carga disp.67
Paquete 33 con peso 6 reparte conductor 0, tiene carga disp.61
Paquete 37 con peso 5 reparte conductor 0, tiene carga disp.56
Paquete 38 con peso 7 reparte conductor 0, tiene carga disp.49
Paquete 39 con peso 7 reparte conductor 0, tiene carga disp.42
Paquete 36 con peso 10 reparte conductor 0, tiene carga disp.32
Paquete 24 con peso 12 reparte conductor 0, tiene carga disp.20
Conductor 0 está cargado
Conductor 0 sale de las instalaciones
Paquete 30 con peso 13 reparte conductor 1, tiene carga disp.78
Conductor 0 reparte
Paquete 32 con peso 10 reparte conductor 1, tiene carga disp.68
        Paquete 35 ha sido entregado
Paquete 26 con peso 11 reparte conductor 1, tiene carga disp.57
Paquete 21 con peso 8 reparte conductor 1, tiene carga disp.49
Paquete 27 con peso 9 reparte conductor 1, tiene carga disp.40
Paquete 15 con peso 14 reparte conductor 1, tiene carga disp.26
Conductor 1 está cargado
Conductor 1 sale de las instalaciones
Paquete 20 con peso 9 reparte conductor 2, tiene carga disp.84
```

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura

Conductor 1 reparte
Paquete 30 ha sido entregado
Paquete 2 con peso 5 reparte conductor 2, tiene carga disp.79
Paquete 14 con peso 9 reparte conductor 2, tiene carga disp.70
Paquete 31 con peso 13 reparte conductor 2, tiene carga disp.57
Paquete 8 con peso 9 reparte conductor 2, tiene carga disp.48
Paquete 25 con peso 10 reparte conductor 2, tiene carga disp.38
Paquete 3 con peso 14 reparte conductor 2, tiene carga disp.24
Conductor 2 está cargado
Conductor 2 sale de las instalaciones
No hay hueco 7 con peso 6
Conductor 2 reparte
No hay hueco 12 con peso 8
No hay hueco 9 con peso 7
Paquete 20 ha sido entregado
No hay hueco 29 con peso 8
No hay hueco 13 con peso 6
No hay hueco 18 con peso 11
No hay hueco 19 con peso 6
No hay hueco 0 con peso 10
No hay hueco 6 con peso 7
No hay hueco 28 con peso 10
No hay hueco 1 con peso 6
No hay hueco 22 con peso 9
No hay hueco 23 con peso 12
No hay hueco 11 con peso 13
No hay hueco 10 con peso 11
No hay hueco 5 con peso 6
No hay hueco 16 con peso 11
No hay hueco 17 con peso 13
No hay hueco 4 con peso 13
Paquete 34 ha sido entregado
Paquete 2 ha sido entregado
Paquete 32 ha sido entregado
Paquete 14 ha sido entregado
Paquete 33 ha sido entregado
Paquete 31 ha sido entregado
Paquete 26 ha sido entregado
Paquete 8 ha sido entregado
Paquete 37 ha sido entregado
Paquete 25 ha sido entregado
Paquete 21 ha sido entregado

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura

Paquete 3 ha sido entregado
Paquete 38 ha sido entregado
Paquete 27 ha sido entregado
Paquete 39 ha sido entregado
Paquete 15 ha sido entregado
Paquete 36 ha sido entregado
Paquete 24 ha sido entregado
Paquete 7 con peso 6 reparte conductor 0, tiene carga disp.82
Paquete 12 con peso 8 reparte conductor 0, tiene carga disp.74
Paquete 29 con peso 8 reparte conductor 0, tiene carga disp.66
Paquete 0 con peso 10 reparte conductor 0, tiene carga disp.56
Paquete 28 con peso 10 reparte conductor 0, tiene carga disp.46
Paquete 19 con peso 6 reparte conductor 0, tiene carga disp.40
Paquete 13 con peso 6 reparte conductor 0, tiene carga disp.34
Paquete 4 con peso 13 reparte conductor 0, tiene carga disp.21
Conductor 0 está cargado
Paquete 6 con peso 7 reparte conductor 1, tiene carga disp.84
Conductor 0 sale de las instalaciones
Paquete 18 con peso 11 reparte conductor 1, tiene carga disp.73
Conductor 0 reparte
Paquete 9 con peso 7 reparte conductor 1, tiene carga disp.66
Paquete 7 ha sido entregado
Paquete 11 con peso 13 reparte conductor 1, tiene carga disp.53
Paquete 5 con peso 6 reparte conductor 1, tiene carga disp.47
Paquete 10 con peso 11 reparte conductor 1, tiene carga disp.36
Paquete 23 con peso 12 reparte conductor 1, tiene carga disp.24
Conductor 1 está cargado
Conductor 1 sale de las instalaciones
Conductor 1 reparte
Paquete 22 con peso 9 reparte conductor 2, tiene carga disp.84
Paquete 6 ha sido entregado
Paquete 17 con peso 13 reparte conductor 2, tiene carga disp.71
Paquete 1 con peso 6 reparte conductor 2, tiene carga disp.65
Paquete 16 con peso 11 reparte conductor 2, tiene carga disp.54
Paquete 12 ha sido entregado
Paquete 18 ha sido entregado
Paquete 29 ha sido entregado
Paquete 9 ha sido entregado
Paquete 0 ha sido entregado
Paquete 11 ha sido entregado
Paquete 28 ha sido entregado
Paquete 5 ha sido entregado

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura

Paquete 19 ha sido entregado
Paquete 10 ha sido entregado
Paquete 13 ha sido entregado
Paquete 23 ha sido entregado
Paquete 4 ha sido entregado
Las instalaciones se cierran
Conductor 2 sale en ultimo reparto
Conductor 2 sale de las instalaciones
Conductor 2 reparte
Paquete 22 ha sido entregado
Paquete 17 ha sido entregado
Paquete 1 ha sido entregado
Paquete 16 ha sido entregado

Casos que se recomienda probar:

//Un solo conductor, y muchos paquetes.

```
NC = 1
NP = 40
```

//Un solo conductor y pocos paquetes, interesa que no se llene y comprobar el comportamiento cuando se cierra en negocio.

```
NC = 1
NP = 2
```

//Muchos conductores, muchos paquetes.

```
NC = 3
NP = 40
```

This work is only accessible in the Virtual Campus of UMA.
Autor: Profesores de la asignatura