Universidad de Valladolid

Departamento de Informática

Programación Orientada a Objetos

Primera entrega de Prácticas

Curso 2020/21

1. Supuesto práctico

El comercio de mercancías siempre ha sido uno de los motores de la economía de los países. Los sistemas de transporte de mercancías son necesarios y vitales para este comercio. Las tareas que conllevan estos sistemas son principalmente: el transporte, almacenamiento y distribución de estas mercancías de forma segura. Una de las formas más populares de transporte es utilizar contenedores. ***MArkse-Co*** es una empresa especializada en este nicho de mercado. Opera principalmente en puertos y desea ofrecer una gestión eficiente y planificada de esta distribución y almacenamiento de contenedores de mercancías para sus clientes. Los clientes compran uno o varios de servicios de transporte para cada contenedor. Estos servicios de transporte son de puerto a puerto, (es decir en barco de un puerto a otro).

Figura 1 Descarga de contenedores

Suponemos que los contenedores son traídos por los clientes a un muelle de uno de los puertos que gestiona ***MArkse-Co*** para comenzar el trayecto en barco. En cada muelle se pueden almacenar un número limitado de contenedores. En los muelles, los contenedores se van almacenando en una serie de plazas. En cada plaza se pueden apilar hasta 4 contenedores si todos tienen las mismas dimensiones. Hay contenedores que no se pueden disponer encima de otros porque no tienen techo.

Los clientes de ***MArkse-Co*** contratan el servicio de transporte de un contenedor desde un muelle de un puerto a otro puerto; puede ocurrir que ese transporte se realice de forma directa o que, se divida en varios trayectos.

1.1. Funcionalidad a requerida

Esta práctica no consiste en la implementación de la interfaz de usuario de este sistema sino del conjunto de clases que controlan el sistema utilizado para esta gestión. En ningún caso se trata de clases que interaccionen con los usuarios.

La clase `Puerto` debe proporcionar la gestión de un puerto de una localidad. Cada puerto se identifica con dos letras mayúsculas con el nombre del país, un guión y finalmente tres letras mayúsculas con el código de la localidad. En cada puerto hay una serie de muelles a los que llegan barcos llenos de contenedores. La funcionalidad mínima de cada puerto es:

- añadir un nuevo `Muelle` a un puerto
- eliminar un `Muelle` de un puerto a partir de su identificador
- obtener si el puerto está completo o no; entendiendo por completo que no hay posibilidad de almacenar ningún contenedor más
- obtener una lista de los `Muelles` que estén operativos
- obtener una lista de los `Muelles` que tengan espacio
- obtener una lista de los `Muelles` que se encuentran a una distancia inferior dada de cierto punto GPS.

La clase `Muelle` está caracterizada por un identificador un número de dos dígitos, un punto GPS, un estado (operativo o fuera de servicio). Cada muelle tiene un número variable de plazas. En cada plaza se pueden apilar como máximo cuatro contenedores. Será necesario proporcionar al menos la funcionalidad que permita:

- conocer el número de plazas que tiene el muelle

- conocer el número de plazas vacías, semi-llenas y completas
- dado un código de contenedor, indicar la plaza en la que está
- dado un código de contenedor, indicar en qué nivel de una plaza está apilado
- asignar un contenedor a una plaza y apilarlo encima de otro si es posible
- sacar un contenedor de una plaza y desapilarlo

La clase `Contenedor` está caracterizada [1] por el código del dueño (3 letras mayúsculas), una letra (U, J o Z) que indica el equipamiento, un número de serie de 6 dígitos, un dígito de control obtenido de un algoritmo (ver [2]), el peso de la tara (es decir el peso del contenedor), la máxima carga útil permitida y el volumen. Las unidades de medidas para el peso estarán dadas en Kilogramos y/o Libras y las de volumen en metros cúbicos y/o pies cúbicos.

Cada `Contenedor` necesita un estado que indique si se encuentra en tránsito o en recogida, asimismo hay que indicar si el contenedor tiene techo o no (si no tiene techo, entonces no se puede disponer otro contenedor encima de él).

La funcionalidad mínima requerida será:

- cambiar el estado de un contenedor para reflejar que está en recogida
- cambiar el estado de un contenedor para reflejar que está en tránsito
- cambiar a contenedor tiene techo o no
- obtener el volumen del contenedor en metros cúbicos
- obtener el volumen del contenedor en pies cúbicos
- obtener el peso del contenedor en Kilogramos
- obtener el peso del contenedor en Libras
- obtener el precio del transporte total de un contenedor a partir de sus trayectos.

Cada `Contenedor` puede realizar uno o varios viajes (de puerto en puerto) hasta llegar a destino. Esta información se va a almacenar gracias a la clase `Trayecto`. Las instancias de la clase `Trayecto` deben almacenar un muelle de origen, puerto de origen, una fecha de inicio del trayecto, muelle de destino, puerto de destino y la fecha del fin de trayecto.

La funcionalidad mínima requerida será:

- conocer si la fecha de fin de trayecto es superior a una dada
- a partir del coste por día de trayecto dado y el coste por milla marina, obtener el precio de un trayecto en euros
- obtener la distancia en millas marinas de un trayecto
- obtener información completa del trayecto: indicando localidad del puerto de origen, país y fecha de inicio del trayecto y localidad del puerto de destino, país y fecha de fin de trayecto

1.2. La clase `GPSCoordinate`

Se aporta el bytecode de una clase externa `GPSCoordinate`.

Se dispone de la documentación de dicha clase[3] y el bytecode (.class) para incorporar al proyecto[4].

Como puede verse, la clase `GPSCoordinate` tiene métodos para consultar las coordenadas y obtener la distancia de una coordenada a otra.

[1] https://www.bic-code.org/bic-codes
[2] https://en.wikipedia.org/wiki/ISO_6346
[3] http://www.infor.uva.es/~yania/pub/poo/GPSCoordinate/doc/
[4] http://www.infor.uva.es/~yania/pub/poo/GPSCoordinate/bin/GPSCoordinate.class

1.3. Clases

Se espera que las clases que forman el proyecto Eclipse de la entrega sean las clases `Puerto`, `Muelle`, `Contenedor`, `Trayecto`, y las correspondientes clases de prueba `PuertoTest`, `MuelleTest`, `ContenedorTest` y `TrayectoTest`.

Las clases de prueba deben ser clases **JUnit 4** A modo de ejemplo, se ofrecen las pruebas a las que se ha sometido a la clase `GPSCoordinate` en el aula virtual de la asignatura.

Si la solución presentada está basada en alguna otra clase adicional a las mencionadas, cada clase del proyecto debe venir acompañada de su correspondiente clase de prueba implementada mediante **JUnit 4** y nombrada con el mismo esquema.

2. Condiciones de entrega

- La entrega consistirá en un único archivo `.zip`.
- El archivo contendrá el proyecto Eclipse compatible con la versión Eclipse 2020-06.
- El proyecto debe llamarse `entrega1-idAlumno1-idAlumno2`, donde `idAlumno` se refiere al identificador de la cuenta de laboratorio de cada alumno y residir en un directorio del mismo nombre.

  Ejemplo: `entrega1-javper-margar`
- El proyecto debe compilar.
- La entrega se realizará mediante la subida del archivo zip a una tarea habilitada al respecto en el aula virtual. La entrega debe hacerse una sola vez por equipo. El equipo es responsable de decidir cuál de sus integrantes es el encargado de subir la práctica.
- El límite para la entrega de la práctica se establece en las **23:55 del 20 de noviembre de 2020**
- **No se admitirán entregas que incumplan estas condiciones**.
- En caso de incumplimiento de las condiciones anteriores, se considerará la práctica como **no presentada**.
- Es necesario que el código cumpla las convenciones de código **Java**[5].
- Es necesario documentar las clases mediante comentarios **JavaDoc**[6].
- Es deseable que cada archivo `.java` contenido en la entrega tenga en la cabecera (comentarios **JavaDoc**) el nombre de los autores (mediante la etiqueta `@author`). Para indicar el nombre, se preferirá el identificador de la cuenta de laboratorio de cada alumno en lugar de su nombre completo.

2.1. Aclaración relativa a las defensas

Las defensas deberán realizarse a lo largo de la **semana del 23-27 de Noviembre** [7]. Debido a las actuales circunstancias, las defensas se podrán hacer telemáticas. Cada equipo que desee concertar con su profesor de prácticas la fecha y hora de la defensa, enviará un correo con su propuesta a su profesor de prácticas. No es necesario esperar a la fecha de la entrega para ir acordando con el profesor la fecha y hora de la defensa. Incluso sería deseable tenerlo acordado con antelación suficiente para poder aprovechar mejor la semana posterior a la entrega.

Es necesario que todos los miembros del equipo estén presentes en la defensa.

En caso de no realizar la defensa, la práctica tendrá la consideración de **no presentada**.

[5] http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

[6] http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

[7] Si algún equipo considera haber terminado antes del 20 de noviembre y desea realizar la defensa en la semana del 16 al 20 deberá hablarlo con su profesor de prácticas antes de la entrega y acordar con él la fecha y hora de defensa.

3. Referencias

- https://www.bic-code.org/bic-codes/
- https://www.stocklogistic.com/codigos-contenedor-maritimo/
- https://en.wikipedia.org/wiki/ISO_6346#Check_Digit