Programación Orientada a Objetos

Práctica 1. Curso 2017-18

1. Supuesto práctico

Los sistemas de préstamos de bicicletas se encuentran en la mayor parte de las grandes ciudades. Los usuarios pueden coger las bicicletas aparcadas por la ciudad en puntos provistos para ese fin.

En la ciudad hay una serie de puntos de aparcamiento de bicis en diferentes localizaciones. Cada punto de aparcamiento tiene una cantidad de módulos de anclaje que pueden estar vacíos u ocupados. Los puntos de aparcamiento tienen un identificador único en el sistema y una posición GPS para indicar su localización en la ciudad. La posición se indica con coordenadas GPS; estas coordenadas se pueden expresar en grados decimales(GD) o grados, minutos y segundos sexagesimales (GMS). En la sección 1.2 se explican cuestiones relativas a GD y GMS para coordenadas GPS.

Para coger una bicicleta, el usuario debe tener una tarjeta que da acceso al servicio. La tarjeta es una tarjeta monedero en la que el usuario debe garantizar que haya saldo suficiente. Este saldo será utilizado por el sistema en concepto de fianza cuando se le presta una bici.

El sistema cobra temporalmente la fianza por prestar la bici. Es decir, comprueba que en la tarjeta haya saldo suficiente para cubrir la fianza, lo cobra al coger la bici y lo reembolsa al devolverla.

La cantidad de fianza se establece al crear el sistema de préstamos en la ciudad, pero es revisable y cada cierto tiempo el ayuntamiento podrá modificar el valor de esta fianza.

En un punto de aparcamiento de bicis, el usuario se acerca a un módulo de anclaje y pasa la tarjeta. Se consulta el saldo de la tarjeta, y si éste es suficiente para cubrir la fianza, se cobra temporalmente. Entonces el módulo permite coger una bicicleta abriendo el anclaje y éste pasa a estar `vacío' (`no ocupado'). Si el saldo no es suficiente, el módulo no abrirá el anclaje y se considerará una situación no válida. Si en el módulo por el que se ha pasado la tarjeta, no hay una bicicleta (está `vacío'), se considerará una situación no válida.

Cuando se devuelve una bicicleta, el usuario pasará su tarjeta monedero por el lector de un módulo de anclaje `vacío' para que se reembolse la fianza y anclará la bicicleta a dicho módulo. Se reembolsa la fianza al saldo de la tarjeta y el anclaje en el módulo pasa a estar `ocupado'. Al devolver la bici, si el módulo de anclaje que se pretende usar está `ocupado', se considerará una situación no válida.

UVa **Universidad** de **Valladolid.** Dpto. de Informática. Edificio de Tecnologías de la Información y las Telecomunicaciones, Campus Miguel Delibes. 47011 Valladolid. Tfno. 983.423000/Fax 983.423671.
e-mail:yania,marga,felix@infor.uva.es

1.1. Funcionalidad a proveer

Se recuerda que no se está implementando la interfaz de uso del módulo de anclaje sino las clases del dominio que controlan el sistema. No son clases frontera para la interacción con el usuario.

Se nos solicita que programemos dos clases que llamaremos `CityBikeSystem` y `CityBikeParkingPoint`.

La clase `CityBikeSystem` debe brindar funcionalidad que permita como mínimo: añadir un punto de aparcamiento de bicis, eliminarlo, modificar la cantidad establecida para la fianza, obtener todos los puntos de aparcamientos de bicis que hay en la ciudad, obtener todos los puntos de aparcamiento con bicis disponibles (que tengan anclajes ocupados), obtener todos los puntos de aparcamiento en los que haya sitio para devolver (anclar) una bici (es decir que tengan anclajes vacíos) y obtener puntos de aparcamiento de bicis cercanos a una coordenada GPS dada dentro de un radio en metros dado.

La clase `CityBikeParkingPoint` debe brindar funcionalidad que permita como mínimo: conocer la cantidad de anclajes que tiene el punto de aparcamiento, conocer la cantidad de anclajes ocupados (en los que hay una bici aparcada), conocer la posición GPS en la que se encuentra el punto de aparcamiento, conocer el identificador del punto de aparcamiento, dado un módulo de anclaje en ese aparcamiento[1], saber si está vacío u ocupado, prestar una bici de un módulo de anclaje, devolver una bici a un módulo de anclaje y conocer la distancia del punto de aparcamiento a una coordenada GPS dada o a otro punto de aparcamiento de bicis dado.

Cuando se crea un punto de aparcamiento, se crea con una cantidad de módulos de anclaje inicial. En esta versión no se va a permitir añadir módulos de anclaje una vez creado el punto de aparcamiento.

1.2. GD y GMS y coordenadas GPS

Las coordenadas GPS localizan una posición de un objeto en toda la Tierra. La localización se realiza caracterizando la latitud y la longitud de la posición.

- La latitud es la distancia angular que hay desde un punto de la superficie de la Tierra hasta el paralelo del Ecuador (paralelo $0°$); se mide en grados, minutos y segundos sexagesimales (GMS) sobre los meridianos. Hay que especificar si se encuentra en latitud Norte o en latitud Sur.

- La longitud es la distancia angular entre un punto dado de la superficie terrestre y el meridiano que se toma como $0°$; se mide en grados, minutos y segundos sexagesimales (GMS). Hay que especificar si se encuentra en longitud Este o longitud Oeste.

Una forma más sencilla de indicar la latitud y la longitud es expresar cada una de ellas en grados decimales. Los grados decimales tienen signo, puede ser una cantidad positiva o una cantidad negativa. Ambas medidas: en grados decimales GD o GMS se utilizan de forma indistinta.

1.2.1. Transformar de GMS a GD, y viceversa

En primer lugar hay que tener en cuenta el signo que va a tener en GD. El signo del valor del GD será positivo si se refiere a una latitud Norte (N) o a una longitud Este (E). Será negativo en caso contrario, es decir, en los casos Sur (S) y Oeste (O).

El método para transformarlo es muy simple. Si se tiene que la posición de latitud o longitud está dada en GMS por $A°\ B'\ C''$, la transformación resultante sería: $GD = (signo)(A + B/60 + C/3600)$, donde el signo tendría en cuenta Latitud N/S o Longitud E/O.

Ejemplo: Si tenemos que la longitud en GMS es $47°15'10''E$, entonces Longitud en GD = $+(47 + 15/60 + 10/3600)$, es decir $+47{,}25278$. Si la latitud en GMS es $21°0'54''S$, entonces Latitud en GD = $-(21+0/60+54/3600)$,

[1] Esta indicación de qué módulo de anclaje en ese punto de aparcamiento puede ser sencillamente un número de orden.

es decir $-21{,}015$. En la red se encuentran calculadoras que pueden ser utilizadas para comprobar estos, y otros, ejemplos[2].

La transformación inversa de GD a GMS es fácil de deducir a partir de lo anterior.

1.2.2. Distancia entre dos puntos dados por coordenadas GPS

El cálculo de la distancia entre dos puntos por su latitud y longitud está relacionado con la suposición de que la Tierra es una esfera. Con esta suposición se consiguen unos resultados más que fiables. Por lo que no se utiliza la conocida distancia euclídea sino la fórmula del semiverseno o haversine.

En algunas páginas en la red pueden encontrarse explicaciones sencillas sobre este tema[3],[4].

Sea $dlon = lon2 - lon1$ la diferencia entre las longitudes de ambos puntos en radianes y $dlat = lat2 - lat1$ la diferencia entre las latitudes de ambos puntos en radianes. Entonces, siendo R el radio de la tierra 6371 kilómetros, para hallar la distancia d entre dos puntos habría que calcular:

- $a = (\sin(dlat/2))^2 + \cos(lat1) * \cos(lat2) * (\sin(dlon/2))^2$
- opción a) $c = 2 * \arcsin(min(1, \sqrt{(a)}))$, opción b) $c = 2 * atan2(\sqrt{(a)}, \sqrt{(1-a)})$
- $d = R * c$

1.3. La clase `TarjetaMonedero`

Se aporta el bytecode de una clase `TarjetaMonedero` distribuida por el proveedor del sistema de tarjetas. Se dispone de la documentación de dicha clase[5] y el bytecote (.class) para incorporar al proyecto[6].

Como puede verse, la clase `TarjetaMonedero` tiene métodos para consultar el saldo actual de la tarjeta y descontar del saldo actual. La clase `TarjetaMonedero` también tiene un constructor para crear una tarjeta con su saldo inicial y un método para cargar saldo en la tarjeta. Todos los métodos que modifican el estado de la tarjeta necesitan una credencial para ser utilizados. La credencial de descontar del saldo es diferente de la de crear la tarjeta con un saldo inicial y de cargar el saldo.
Las credenciales actuales para ambas situaciones son:

saldo inicial, cargar saldo: A156Bv09_1zXo894

descontar del saldo: 6Z1y00Nm31aA-571

1.4. Clases

Se espera que las clases que forman el proyecto Eclipse de la entrega sean: una clase `CityBikeSystem`, una clase `CityBikeParkingPoint` y las clases de test `CityBikeSystemTest` y `CityBikeParkingPointTest`.

Las clases `CityBikeSystemTest` y `CityBikeParkingPointTest` deben ser clases de prueba JUnit 4. A modo de ejemplo, se ofrecen las pruebas a las que se ha sometido a la clase `TarjetaMonedero`: `TarjetaMonederoTest`[7].

Se aclara que si la solución del equipo 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.

[2] https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps
[3] http://www.movable-type.co.uk/scripts/gis-faq-5.1.html
[4] http://www.lexjansen.com/nesug/nesug06/dm/da15.pdf
[5] http://www.infor.uva.es/~yania/pub/poo/TarjetaMonedero/doc/
[6] https://aulas.inf.uva.es/mod/resource/view.php?id=22463
[7] https://aulas.inf.uva.es/mod/resource/view.php?id=22497

2. Condiciones de entrega

- La entrega consistirá en un único archivo `.tar`, `.tgz`, `.tar.gz`, `zip`, `rar`.
- El archivo contendrá el proyecto ECLIPSE compatible con Eclipse Mars-2.
- El proyecto ECLIPSE deberá nombrarse con `entrega1-idAlumno1-idAlumno2`.
- El `idAlumno` se refiere al identificador de la cuenta de laboratorio de cada alumno.
- Ejemplo: `entrega1-javper-margar`
- El proyecto deberá compilar.
- La entrega se realizará mediante la subida del archivo así obtenido en una tarea habilitada al respecto en el aula virtual. Se entrega una sola vez por cada equipo. El equipo decide uno de sus integrantes encargado de subir la práctica.
- La fecha y hora límites para la entrega se establecen en la tarea del aula virtual y son: 10 de Noviembre de 2017 a las 23:55.
- **No se admitirán entregas que incumplan estas condiciones**.
- **En caso de incumplimiento** de las condiciones anteriores, se considerará la práctica como **no entregada**.
- Es recomendable que el código cumpla las convenciones de código JAVA[8].
- Será necesario documentar las clases mediante comentarios JavaDoc[9].
- 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 posterior a la entrega, es decir, en la semana del 13-17 de Noviembre. Se dispondrá de las dos horas de laboratorio para realizar esas defensas y se podrá acordar otro día y hora con los equipos para este fin. Para ello, cada equipo que desee acordar 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 salvo causa debidamente justificada.

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

[8] http://www.oracle.com/technetwork/java/codeconvtoc-136057.html
[9] http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html