PEC 2

Clases y objetos

Universitat Oberta de Catalunya

Información relevante:

- Fecha límite de entrega: 18 de octubre.
- Peso en la nota de FC: 10%.

Contenido

Información docente

Esta PEC está vinculada con el módulo teórico “Abstracción y encapsulación” de los apuntes de la asignatura.

Prerrequisitos

Para hacer esta PEC, necesitas:

- Tener adquiridos los conocimientos de la PEC 1. Para ello te recomendamos que mires la solución que se publicó en el aula y la compares con la tuya.

Objetivos

Con esta PEC el equipo docente de la asignatura busca que:

- Sepas diferenciar entre objeto y clase, siendo capaz de abstraer la clase a partir de un conjunto de objetos/entidades reales que están relacionados entre sí.
- Veas el potencial de la encapsulación.
- Uses Java como un lenguaje de programación orientado a objetos.
- Crees programas bien documentados que faciliten su mantenimiento posterior.
- Entiendas qué es una excepción y cómo se utiliza en Java.
- Conozcas las diferencias entre las clases `String`, `StringBuilder` y `StringBuffer`.
- Acabes de familiarizarte con la dinámica de la asignatura.

Resultados de aprendizaje

Con esta PEC debes demostrar que eres capaz de:

- Codificar una clase a partir de unos requisitos y su representación como diagrama de clases.
- Entender la diferencia entre los diferentes modificadores de acceso.
- Comprender qué consecuencias tiene declarar un miembro como `static`.
- Gestionar casos anómalos, incorrectos e indeseados mediante excepciones.
- Documentar con Javadoc un programa.
- Saber utilizar las clases `String` y `StringBuilder`.
- Utilizar test unitarios para determinar que un programa es correcto.
- Usar con cierta soltura un entorno de desarrollo integrado (IDE) como Eclipse.

Enunciado

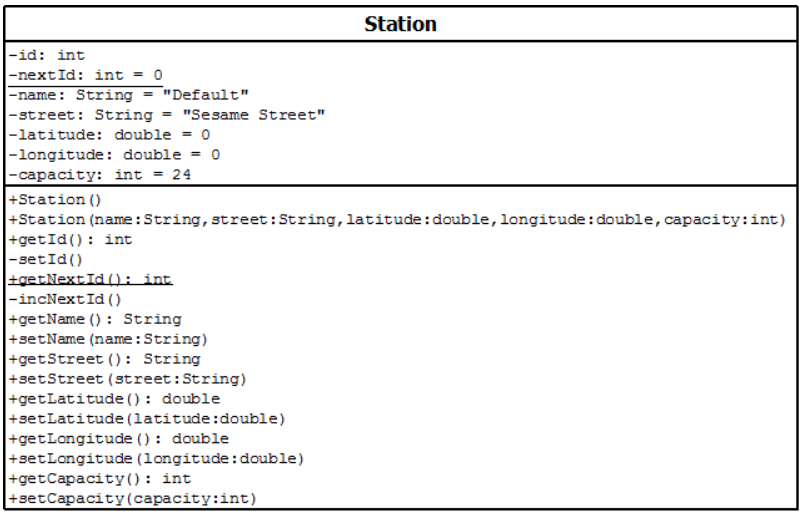
Esta PEC contiene 4 ejercicios evaluables. Debes entregar tu solución de los 4 ejercicios (ver el último apartado).

Ejercicio 1 (5 puntos)

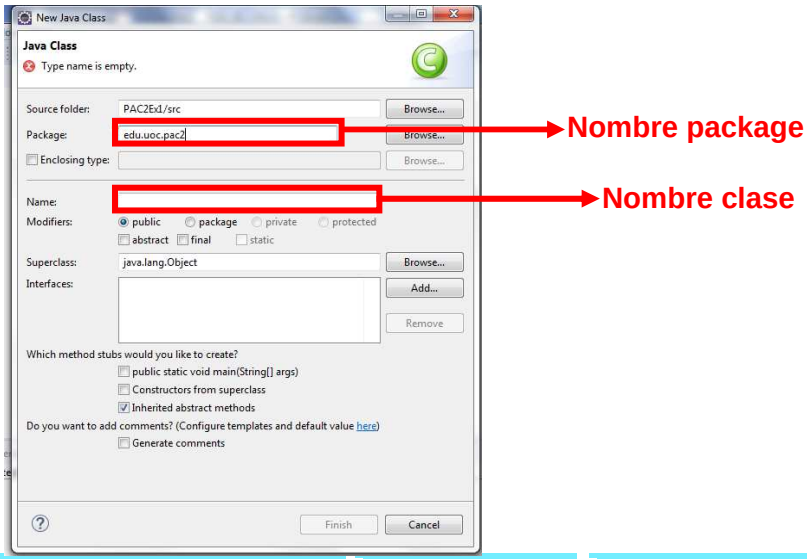
Antes de empezar debes:

- Leer los apartados del 3.1 al 3.6 de la Guía de Java.

Una vez realizada la tarea anterior **abre o importa el proyecto PAC2Ex1 en Eclipse**. En el *package* `edu.uoc.pac2.tests` está el fichero con los test unitarios que te proporcionamos. En el *package* `edu.uoc.pac2` tienes que codificar la clase `Station`, cuya representación en diagrama de clases UML es:



Para crear una nueva clase selecciona el directorio `src` en el menú izquierdo de Eclipse –llamado *Package Explorer*– y haz botón derecho con el ratón y selecciona en el menú contextual `New` → `Class`. En Java los nombres de los ficheros deben llamarse exactamente igual que la clase que contienen. Así pues, en el caso de este ejercicio debes crear el fichero `Station.java` que contendrá la clase `Station`. En la misma ventana donde pones el nombre de la clase (campo `Name`) puedes indicar el *package* al que pertenece dicha clase (ver recuadro rojo en la siguiente imagen).



La clase `Station` representa una estación de bicicletas urbanas ubicada en un lugar de una ciudad. Para la codificación de la clase `Station` debes tener en cuenta las siguientes especificaciones/consideraciones:

- El constructor por defecto debe inicializar todos los atributos con el valor por defecto que se indica en el diagrama de clases UML.

- El valor del atributo `id` se obtiene asignándole el valor que en ese momento tiene `nextId`. Por ejemplo, si `nextId` es igual a `5`, entonces la estación que estamos instanciando (i.e. el objeto de tipo `Station` que estamos creando) tendrá un `id` igual a `5`. Una vez asignado el valor al atributo `id`, el valor de `nextId` se tiene que incrementar una unidad.

- **Siempre que exista conflicto entre nombres**, sobre todo en los métodos *setter* (i.e. `setXXX`), **debes usar la palabra reservada `this`**. Si te fijas, el nombre del parámetro de todos los métodos *setter* se llama exactamente igual que el atributo de instancia que modifican. Si, por ejemplo, hacemos esto:

```
private void setName(String name){

   name = name;

}
```

¿Cómo sabe el compilador (y nosotros) si el `name` de la izquierda de la asignación es el atributo de la clase (o instancia/objeto) o es el parámetro del método `setName`? Y lo mismo ocurre con el `name` del lado derecho. Así pues, para evitar problemas muchos programadores llaman al parámetro del método utilizando el nombre del atributo a modificar y añadiéndole una coletilla, p.ej. `New`:

```
private void setName(String nameNew){

   name = nameNew;

}
```

Con esta solución no hay duda ni para el compilador ni para nosotros de qué es qué. A pesar de que podemos usar la solución anterior, lo más habitual (y mejor) es utilizar como nombres de los parámetros de los métodos exactamente los mismos que los de los atributos de la clase. Para resolver el conflicto de qué es qué se usa la palabra reservada `this`:

```
private void setName(String name){

   this.name = name;

}
```

La palabra reservada `this` hace referencia al objeto (o instancia) que es del tipo de la clase que estamos codificando. Por lo tanto, gracias a `this`, estamos diciendo que en la asignación el `name` de la izquierda es del objeto/clase y el `name` de la derecha es el parámetro del método `setName`. Podemos decir que el

`this` hace de coletilla e indica que aquello que lleva `this` pertenece al objeto/clase y no es ni un parámetro ni una variable declarados dentro del método. Por ejemplo, si tenemos un objeto de tipo `Station` asignado a la variable norte, entonces cuando llamamos a `setName` haciendo `norte.setName("Barcelona Norte")`, el this que utilizamos dentro del método setName se refiere al objeto `norte`.

- Si el nombre que se quiere asignar a la estación (i.e. atributo `name`) tiene más de `50` caracteres, entonces `setName` no debe asignar dicho nombre a la estación y debe imprimir por pantalla el mensaje `"[ERROR] Station's name cannot be longer than 50 characters"`.

- El valor de la latitud debe estar comprendido dentro del intervalo `[-90,+90]`. Si el valor que se quiere asignar está fuera de este rango, entonces `setLatitude` no debe asignar dicho valor y, en su lugar, debe imprimir por pantalla el mensaje `"[ERROR] Station's latitude must be in range [-90,+90]"`.

- El valor de la longitud debe estar comprendido dentro del intervalo `[-180,+180]`. Si el valor que se quiere asignar está fuera de este rango, entonces `setLongitude` no debe asignar dicho valor y, en su lugar, debe imprimir por pantalla el mensaje `"[ERROR] Station's longitude must be in range [-180,+180]"`.

- El valor de la capacidad (`capacity`; es decir, el número máximo de bicicletas que puede almacenar una estación) debe ser superior a cero. Si el valor que se quiere asignar es incorrecto, entonces no se debe asignar dicho valor y, en consecuencia, se debe imprimir por pantalla el mensaje `"[ERROR] Station's capacity must be greater than 0"`.

Requisito mínimo para evaluar este ejercicio: la clase `Station` debe pasar todos los test proporcionados.

(5 puntos: 3 pts. test proporcionados; 2 pts. otros aspectos)

Ejercicio 2 (1 punto)

Antes de empezar debes:

- Leer el apartado 3.12 de la Guía de Java que habla sobre Excepciones.

Una vez realizada la tarea anterior debes:

- Abrir el proyecto `PAC2Ex2` con Eclipse.

- Copiar todo el *package* `edu.uoc.pac2` del Ejercicio 1 en el proyecto `PAC2Ex2`.

- Modificar los métodos `setName`, `setLatitude`, `setLongitude` y `setCapacity` de la clase `Station` para que en vez de imprimir por pantalla el mensaje de error, lance una `Exception` con el mensaje correspondiente.

- Hacer las modificaciones que creas convenientes/necesarias para trabajar con excepciones de la manera más eficiente dentro de la clase `Station`.

Requisito mínimo para evaluar este ejercicio: la clase `Station` debe pasar todos los test proporcionados.

(1 punto: 0.5 pts. test proporcionados; 0.5 pts. otros aspectos)

Ejercicio 3 (1 punto)

Antes de empezar debes:

- Leer el apartado 5.1 de la Guía de Java que habla sobre Javadoc.

En el proyecto `PAC2Ex2`, documenta con comentarios Javadoc la clase `Station` que has codificado en el Ejercicio 2 de manera que describan los elementos de dicha clase: la clase en sí, sus atributos (independientemente de su modificador de acceso) y sus métodos. Los comentarios pueden ser en español o inglés, pero te recomendamos que sean en inglés.

Una vez documentada la clase, **genera un directorio llamado `doc` dentro del proyecto `PAC2Ex2` que incluya la documentación generada por el comando `javadoc`**. La documentación que generes debe incluir la descripción de la clase `Station` junto con la descripción de todos sus atributos y métodos (independientemente de su modificador de acceso).

(1 punto: 0.5 pts. comentarios Javadoc; 0.5 pts. documentación)

Requisito mínimo para evaluar este ejercicio: debe existir el directorio `doc` con la documentación en formato web generada por el comando `javadoc`.

Ejercicio 4 (3 puntos)

Antes de empezar debes:

- Leer el apartado 4.1 de la Guía de Java que habla sobre la clase `String` y las clases relacionadas `StringBuilder` y `StringBuffer`.

Abre el proyecto `PAC2Ex4` en Eclipse. Utilizando el potencial de las clases `String` y `StringBuilder` codifica los `TODO` del *package* `edu.uoc.pac2.strings`:

- `reverseText`: este método recibe un texto en formato `String` y lo gira. Por ejemplo, si recibe `"hola"`, este método devuelve el `String` `"aloh"`. Si el texto de entrada es `null`, entonces debe devolver `null`.

  (0.5 puntos: 0.25 pts. test proporcionados; 0.25 pts. otros aspectos)

- `countM`: este método recibe un texto en formato `String` y devuelve el número de letras 'm' (ya sea en mayúscula o en minúscula) que contiene el texto. Por ejemplo, si recibe `"Mola"`, este método devuelve `1`. Si el texto de entrada es `null`, entonces este método debe devolver `0`.

  (1 punto: 0.25 pts. test proporcionados; 0.75 pts. otros aspectos)

- `capitalize`: este método recibe un texto en formato `String` y un índice de tipo `int`. Este método convierte a mayúsculas los caracteres ubicados en todas las posiciones anteriores al índice indicado como parámetro. El resto de posiciones se quedan en el formato original. Por ejemplo, si recibe `"hola"` y el índice `2`, entonces este método devuelve `"HOla"`; si recibe `"hoLA"` y el índice `2`, entonces este método devuelve `"HOLA"`.

  Para este método ten en cuenta también que:

- El índice más pequeño posible es `0`, el cual deja el texto de entrada con el mismo formato.
- Si el índice que se indica es igual o más grande que la longitud del texto, devuelve el texto de entrada en mayúsculas.
- Si el índice indicado es negativo, devuelve el texto de entrada con el mismo formato.
- Si el texto de entrada es `null`, entonces debe devolver `null`.

(1.5 puntos: 0.5 pts. test proporcionados; 1 pt. otros aspectos)

Requisito mínimo para evaluar este ejercicio: el programa debe pasar todos los test proporcionados para los métodos `reverseText` y `countM`.

Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: `loginUOC_PEC2.zip`. Por ejemplo: `dgarciaso_PEC2.zip`. Este fichero comprimido tiene que incluir los siguientes elementos:

- El proyecto de Eclipse `PAC2Ex1` completado siguiendo las peticiones y especificaciones del Ejercicio 1.
- El proyecto de Eclipse `PAC2Ex2` completado siguiendo las peticiones y especificaciones del Ejercicio 2, así como la documentación en formato Javadoc cumpliendo los requisitos indicados en el Ejercicio 3.
- El proyecto de Eclipse `PAC2Ex4` completado siguiendo las peticiones y especificaciones del Ejercicio 4.

El último día para entregar esta PEC es el **18 de octubre de 2020** antes de las 23:59. Cualquier PEC entregada más tarde será considerada como no presentada..