Programación II

Curso 21-22

Práctica 3 – Parte b

1 Cuestiones previas

En esta práctica crearemos varias clases para comenzar a desarrollar un parking, que hará uso de las clases definidas en la práctica anterior para modelar los coches. Guarda en P3 todos los ficheros fuente y **.txt** de esta práctica y los **.class** generados, dentro del paquete P3 que creaste en la práctica anterior.

2 Actividad 1: crear las clases de un parking

Crea las siguientes clases, con sus atributos, constructores, y getters/setters:

- Clase `Coordinate`, que representa las coordenadas de una plaza (A2, B12…), con dos atributos privados:
  - **zone**: un carácter, entre la 'A' y la 'Z'.
  - **number**: un número entero estrictamente positivo.

  Añade a esta clase los siguientes métodos:
  - `public boolean isEqualTo (Coordinate c) {...}`

    comprueba si una coordenada es igual a otra (devuelve `true/false` según sea el caso).
  - `public String toText () {...}`

    devuelve una cadena con la coordenada ("A2", "B12"…).

- Clase `CarSpace`, para modelar una plaza del parking, con los siguientes atributos privados:
  - **coordinate**: un objeto de clase `Coordinate`, que indica las coordenadas de la plaza.
  - **plate**: Una cadena de tamaño 7, que representa la matrícula del coche aparcado en esa plaza. Si la plaza está libre, este campo se pondrá a `null`.

  Añade a esta clase el siguiente método:
  - `public String toText () {...}`

    devuelve una cadena con la forma "coordenada;matrícula" (por ejemplo "B4;1234CDF"). Si la plaza está vacía, devolverá sólo la coordenada ("B4"). Fíjate en que la coordenada la devuelve directamente el método `toText` de la clase `Coordinate`.

- Clase `Parking`, para representar la estructura y contenido del parking, y para realizar la gestión de la entrada y salida de coches del mismo. Contiene los siguientes atributos privados:
  - **maxZone**: un carácter de la 'A' a la 'Z' para indicar el máximo número de zonas del parking (las zonas del parking serán las del rango `'A'-'maxZone'`.
  - **sizeZone**: un entero para indicar el numero de plazas por zona (el mismo para todas las zonas).

    Si, por ejemplo, **maxZone**='E' y **sizeZone** =6, el parking tiene 5x6=30 plazas (A1, A2,...A6, ...,E1, E2,...,E6).
  - **lowerElectricZone**: un carácter que indicará una zona, que separa el parking en dos partes:
    - De la zona 'A' a la zona `lowerElectricZone` (no incluida), las plazas son para los coches de combustión.
    - De la zona `lowerElectricZone` (incluida) a la zona `maxZone`, las plazas son para coches híbridos y/o eléctricos.
  - **carSpaces**: un array bidimensional de plazas (objetos de la clase `CarSpace`), cuyo tamaño es el correspondiente a los atributos anteriores **maxZone** y **sizeZone**.

    Pista: el carácter 'A' es el número 65 en el código ASCII, por lo que 'A'-64 es el número 1. De esta forma puedes obtener directamente el número de zonas a partir del carácter **maxZone**.

3 Actividad 2: crear y guardar el parking

En esta actividad vamos a definir el formato de un fichero que almacena el estado del parking, y vamos a implementar los procedimientos para crear un parking a partir de un fichero, y para guardarlo en un fichero.

3.1 Formato de un fichero de parking

El fichero contiene una línea con las dimensiones del parking, y un conjunto de líneas representando los coches actualmente aparcados en el parking (un coche en cada línea).

La primera línea especificará el número de zonas (`maxZone`), el número de plazas por zona (`sizeZone`), y la indicación de las zonas para coches eléctricos e híbridos, según el formato siguiente:

```
maxZone;sizeZone;lowerElectricZone
```

Las siguientes líneas del fichero, que identifican cada coche aparcado en el parking, consistirán en la coordenada de la plaza del parking y la matrícula del coche aparcado en ella. Además, podrá haber líneas con comentarios (comienzan por '#').

Un ejemplo de este fichero podría ser:

```
# zonas 'A'-'F', cada una con 10 plazas. Las zonas 'E' y 'F' para eléctricos
F;10;E
# coches aparcados
A4;1234BCD
C9;5678FGH
```

3.2 Crear un parking a partir de un fichero

Para crear el parking a partir del fichero, añade el siguiente constructor a la clase `Parking`:

```
public Parking (String fileName) {...}
```

donde `fileName` indicará el nombre del fichero del parking. El código de este constructor tiene que:

1. Abrir el fichero, leer la línea de dimensionamiento del parking, e inicializar los atributos de la clase.
2. Crear el array bidimensional de plazas (**carSpaces**) según las dimensiones leídas, e inicializarlo con plazas (objetos `CarSpace`) con su coordenada correspondiente y la matrícula a `null`.
3. Leer el resto de líneas del fichero, que representan coches aparcados. Para cada coche, debe construir el objeto `Coordinate` de las coordenadas de la plaza donde está aparcado, y recorrer luego el array **carSpaces** buscando la plaza (objeto `CarSpace`) que corresponde a esa coordenada. Al encontrarlo, actualizar la matrícula de esa plaza para reflejar que está ocupada.

   Pista: usa el método `isEqualTo` de la clase `Coordinate` para comparar la coordenada de cada plaza leída del fichero con las coordenadas de las plazas del array **carSpaces**.

3.3 Almacenar un parking en un fichero

Para almacenar en un fichero el estado del parking debes crear el siguiente método en la clase `Parking`:

```
public void saveParking (String fileName) {...}
```

Este método creará el fichero y guardará en la primera línea las dimensiones del parking, y, a continuación, la información sobre todas las plazas ocupadas, una en cada línea, de acuerdo al formato descrito anteriormente. Las plazas ocupadas se guardarán de forma ordenada (zonas de menor a mayor, y, dentro de cada zona, primero las de número de plaza más bajo).

Pista: fíjate en que la información que hay que guardar sobre cada plaza ocupada es exactamente la que devuelve el método `toText` de la clase `CarSpace`.

Recuerda que, si no capturas las excepciones que se pueden producir en el manejo de los ficheros, debes declararlas en la definición de los métodos (con el correspondiente `throws`).

4 Actividad 3. Entradas y salidas del parking

4.1 Especificación del fichero de entradas y salidas

La secuencia de las entradas y salidas del parking se leerá de un fichero, que contendrá una o más líneas con el siguiente formato (además de comentarios en cualquier línea):

```
<typeES>;<plate>;<carType>
```

- **typeES*:*** un carácter que indica si es una entrada ('I') o una salida ('O').
- **plate**: matrícula del coche que entra/sale.
- **carType:** un carácter que indica si el coche es de combustión (C), eléctrico (E) o híbrido (H). Este campo sólo aparecerá en las entradas.

Un ejemplo, con 3 entradas y una salida, sería:

```
I;1700CGS;C
I;2408FKL;C
I;1234LGP;E
O;2408FKL
```

4.2 Entradas al parking.

Si el coche que entra es de combustión, sólo puede aparcar en las plazas para coches de combustión, es decir, en las zonas por debajo de `lowerElectricZone` (no incluida). Si el coche es eléctrico o híbrido, sólo puede aparcar en las plazas reservadas para estos coches, es decir, de la zona `lowerElectricZone` en adelante.

Para implementar la entrada de un coche al parking, crea en la clase `Parking` el siguiente método:

```
public void carEntry (String plate, char carType) {...}
```

donde *plate* es la matrícula del coche que entra, y *carType* el tipo de coche.

Este método debe acceder al array de plazas **carSpaces**, y localizar la plaza libre más baja por orden de coordenada (según el tipo de coche). Debe almacenar la matrícula de este coche en el campo de la plaza, marcándola así como ocupada.

Pista: si es un coche de combustión, debes recorrer las zonas desde la 'A' hasta la `lowerElectricZone` (no incluida) buscando la primera plaza libre. Si es de otro tipo, desde la zona `lowerElectricZone` hasta la última. En cada caso, se empezará por la zona más baja, y, dentro de cada zona, por el número más bajo, y se irá subiendo progresivamente hasta encontrar una plaza libre.

4.3 Salidas del parking.

Para implementar la salida de un coche del parking, crea en la clase `Parking` el siguiente método:

```
public void carDeparture (String plate) {...}
```

Este método debe buscar en el array **carSpaces** la plaza en la cual estaba aparcado este coche. Tras encontrarla, debe liberar la plaza, poniendo su campo **plate** a `null`.

5 Actividad 4: crear la clase principal

Crea la clase principal P3b que:

- Reciba y lea tres argumentos, los nombres de tres ficheros (`file1`, `file2` y `file3`).
- Cree el parking (crea un objeto de la clase `Parking`) a partir del fichero `file1`, que contiene el estado de un parking.
- Lea el fichero `file2`, que contiene entradas y salidas, y actualice el parking de acuerdo a su contenido.
- Guarde el estado del parking en el fichero `file3`.

En todos los casos, se puede suponer que los ficheros serán siempre correctos de acuerdo al formato correspondiente, por lo que no es necesario realizar ningún tipo de comprobación sobre su contenido.

6 Actividad 5. Representar gráficamente el parking

Representaremos el parking completo según indica la siguiente figura (para un parking [F;6;E]).

```
A1   2408FKL|A2          |A3          |A4          |A5          |A6   6523GTD|
B1           |B2   1700CGS|B3          |B4          |B5          |B6          |
C1           |C2          |C3          |C4          |C5          |C6          |
D1           |D2          |D3          |D4          |D5          |D6          |
E1 E         |E2 E        |E3 E        |E4 E        |E5 E        |E6 E        |
F1 E         |F2 E        |F3 E 1234LGP|F4 E        |F5 E        |F6 E        |
```

Es decir, ponemos las plazas de cada zona de forma consecutiva y en la misma línea, y cada zona en una línea diferente, representando la información de cada plaza de parking de la forma siguiente:

<Coordenada>ESPACIO<E>ESPACIO<Matricula>|

- <Coordenada>: la coordenada de la plaza (recuerda que la devuelve el método `toText`).
- <E>: Si la plaza es para coches eléctricos o híbridos, se pone una E y, si no, un espacio.
- <Matricula>: La matrícula del coche aparcado en la plaza, o 7 espacios en blanco si está libre (es decir, si la matrícula es `null`).
- Acabamos con el separador '|'.

6.1 Crear el dibujo de un parking

Añade a la clase `Parking` el método `toMap`, que genera el dibujo del parking tal como se ha descrito, y lo devuelve como una cadena.

```
public String toMap () {...}
```

6.2 Guardar el dibujo del parking

Añade a la clase P3b el código necesario para obtener el dibujo del parking llamando al método `toMap` de la clase `Parking` y guardarlo en el fichero *ParkingMap.txt*.