Práctica 5a. Colecciones ArrayList

Cuestiones Previas

En esta práctica modificamos el código desarrollado en la práctica anterior, introduciendo una colección dinámica tipo `ArrayList` para sustituir al array de los coches de la ciudad, e incorporamos a la aplicación los cambios necesarios para poder gestionar la hora de entrada y salida del parking. Para ello, deberías leer antes el documento *"Introducción a la clase ArrayList"* que se encuentra antes que éste.

Copia todo tu código generado en el paquete P4 de la práctica anterior a un nuevo paquete P5a, y realiza en éste todas las modificaciones necesarias para la práctica.

1 Actividad 1. Gestionando la clase CarDB con colecciones

1.1 Subactividad 1. Incorporando colecciones

En esta actividad tienes que modificar el código desarrollado en la práctica 4 en la clase CarDB, para modelar y gestionar los coches leídos del fichero de coches como una colección `ArrayList` de objetos `Car`, en vez de como un array estático. Para ello:

- Modifica la declaración de `cityCars` como array, cambiándola por una colección `ArrayList`. En el constructor de `CarDB`, crea esta colección, sin asignarle una dimensión inicial.
- Incorpora los cambios necesarios en el código de los métodos de la clase `CarDB` que hagan referencia a `cityCars`. Por ejemplo, en `readCityCarsFile`, al ir leyendo coches, crea los correspondientes objetos del tipo de cada coche (`CombustionCar`, `ElectricCar` y `HybridCar`) y añádelos a la colección.
- Crea el método `saveCarsToFile` para guardar en un fichero los datos de la colección `cityCars`, según:

  ```
  public void saveCarsToFile (String filename) {...}
  ```

  Guarda los coches en el fichero en el mismo formato de línea de coche que el del fichero de coches con el cual hemos venido trabajando hasta ahora (sin líneas de comentario), y en el mismo orden que en ese momento tenga la colección `cityCars`. Recuerda que puedes emplear el operador `instanceof` para saber de qué tipo es cada coche que leas del `ArrayList cityCars`.

2 Actividad 2. Incorporando el tiempo en las entradas y salidas

En esta actividad realizamos las modificaciones necesarias para incorporar el instante temporal en el que tienen lugar tanto las entradas como las salidas del parking.

2.1 Especificación del fichero de entradas y salidas

Añadimos a este fichero la hora a la que se produce la entrada o salida. Así, cada línea del fichero tendrá ahora el siguiente formato (además de comentarios):

```
<typeES>;<plate>;time
```

- **typeES*:*** un carácter que indica si es una entrada (I) o una salida (O).
- **plate**: matrícula del coche que entra/sale.
- **time:** la hora de entrada o salida.

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

```
I;1700CGS;12:00
I;2408FKL;12:15
I;1234LGP;12:30
O;2408FKL;13:30
```

2.2 Clase `CarSpace`

Añade a la clase `CarSpace` el atributo `entryTime`, como cadena de caracteres. Añade también el getter y setter correspondiente.

2.3 Creación del parking a partir del fichero

Las líneas del fichero de parking correspondientes a los coches aparcados deberán llevar a mayores la hora de entrada al parking, según se indica debajo (resaltado en negrita):

```
# zonas 'A'-'F', cada una con 10 plazas. Las zonas 'E' y 'F' para eléctricos
F;10;E
# coches aparcados con la hora de entrada
A4;1234BCD;12:30
C9;5678FGH;11:10
```

Modifica el constructor de la clase `Parking` para que procese la hora de entrada del coche de cada línea y se la asigne al atributo `entryTime` de la plaza en la que vas a aparcar el coche.

2.4 Almacenar el parking en un fichero

En el método `saveParking` de la clase `Parking`, añade el código para guardar, al final de cada línea correspondiente a los coches aparcados, la hora de entrada del coche al parking, tomándola del atributo `entryTime` de la plaza, según el formato descrito en el subapartado anterior. Por ejemplo:

```
A4;1234BCD;12:30
```

Las líneas del fichero del parking que representan a los coches aparcados deben mantener el mismo orden de las prácticas anteriores, es decir, deben ir por orden ascendente de coordenada de la plaza.

2.5 Entradas al parking

Modifica el método de entrada al parking añadiéndole el tiempo, según:

```
public void carEntry (Car car, String time) {...}
```

donde `car` es el objeto de clase `Car` que representa al coche que entra (obtenido, por ejemplo, mediante el método `getCarFromPlate` del objeto `cdb`) y `time` la hora de entrada al parking en formato *hh:mm* (hora y minuto). Con el objeto `car` recibido, puedes conocer el tipo del coche para saber dónde aparcarlo (otra vez, el operador `instanceof` puede ayudarte a saber de qué tipo es el coche).

En el código del método `carEntry`, copia el valor del argumento `time` al atributo `entryTime` de la plaza donde aparques el coche.

2.6 Salidas del parking

Modifica el método de salida del parking según:

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

Ahora, el valor que retorna este método es el valor del atributo `entryTime` correspondiente a la plaza donde está aparcado el coche que sale. Cuando liberes la plaza, reinicializa a `null` sus atributos `plate` y `entryTime`.

3 Actividad 3. Clase principal

Crea la clase `P5a`, que contiene el método `main` y es la clase principal de la práctica.

Modifica el método `processIO` de la práctica 4, de forma que tenga en cuenta los cambios descritos en la actividad 2. Es decir, al proceder a actualizar el parking con cada entrada o salida, debe llamar a carEntry y carDeparture con los argumentos apropiados.

En el método `main`:

- Recibe y lee cinco argumentos, los nombres de cinco ficheros:
  - `file1`: un fichero existente con la estructura y contenido de un parking.
  - `file2`: un fichero existente de entradas y salidas.
  - `file3`: el nombre del fichero en el que guardar el resultado de la actualización del parking.
  - `file4`: un fichero existente con todos los coches de ciudad.
  - `file5`: el nombre del fichero donde guardar la colección de coches `cityCars` del objeto `cdb`.
- Crea un objeto `cdb` (como variable global) de la clase `CarDB` e invoca su método `readCityCarsFile` para leer el fichero `file4` con los coches, y que estos se almacenen en el `ArrayList cityCars` del objeto creado `cdb`.
- Crea un objeto de la clase `Parking` (asígnalo a la variable global `miParking`) a partir del fichero `file1`.
- Actualiza `miParking` a partir del fichero de entradas/salidas `file2` (llamando a `processIO`).
- Guarda `miParking`, invocando su método `saveParking` con `file3`.
- Guarda la colección `cityCars` en el fichero `file5,` invocando su método `saveCarsToFile.`