 Proyecto Cupi2	ISIS-1205 Algorítmica y Programación II **Consideraciones adicionales de diseño**
Ejercicio:	n8_parqueadero
Autor:	Equipo Cupi2
Semestre:	2018-1

Manejo de excepciones propias

Para tratar los errores específicos que se presenten en la ejecución de la aplicación y poder asociar más información al error (no solo un mensaje de texto), clasificar errores en grupos y aplicar diferentes estrategias de manejo de error, se ha decidido utilizar clases propias de excepción. Estas son: PersistenciaException, FormatoArchivoException, EstadoParqueaderoException y PlacaException.

1. PersistenciaException

Es la clase de excepción que se lanza cuando se presenta un error de serialización/deserialización, al leer o escribir el archivo binario con la información del estado del mundo, o al leer o escribir un archivo de texto plano. El mensaje asociado con la excepción describe el problema que se presentó. Cuando esta excepción ocurre se registra en el archivo de log, acompañada de la fecha y hora en la que se generó la excepción (ver documento de descripción, sección “Persistencia”).

2. FormatoArchivoException

Es la clase de excepción que indica que hubo un problema procesando el archivo con la información del parqueadero. El mensaje asociado con la excepción describe el problema que se presentó. Cuando esta excepción ocurre se registra en el archivo de log, acompañada de la fecha y hora en la que se generó la excepción (ver documento de descripción, sección “Persistencia”).

3. PlacaException

Es la clase de excepción que se lanza cuando se intenta ingresar un carro por una placa ya existente en el parqueadero o se busca un carro por una placa no existente en el parqueadero. El mensaje asociado a la excepción indica cuál de estas dos situaciones fue la causante, para ello el mensaje y el atributo deben ser almacenados como atributos de la clase (Ver modelo conceptual). Al crear la excepción, se registra el error en el archivo de log con la fecha y hora de ocurrencia (ver documento de descripción, sección “Persistencia”).

4. EstadoParqueaderoException

Es la clase de excepción que se lanza cuando se intenta ingresar o sacar un carro del parqueadero pero este ya se encuentra cerrado, o se intenta ingresar un carro al parqueadero lleno. El mensaje asociado a la excepción indica la razón de la excepción (parqueadero cerrado o lleno), la acción asociada (ingresar o sacar un carro) y la placa del carro. Estos 3 elementos deben ser almacenados como atributos de la clase, utilizando las constantes de la misma para el estado y la acción (Ver modelo conceptual). Al crear la excepción, se registra el error en el archivo de log con la fecha y hora de ocurrencia (ver documento de descripción, sección “Persistencia”).

Escritura en el Archivo de Log

Dado que todas las excepciones se registran en el mismo log debe tenerse en cuenta la forma para extender un archivo de texto ya creado. Esto se hace enviando por parámetro al constructor del objeto PrintWriter un objeto BufferedWriter, que a la vez en su constructor por parámetro recibe un objeto de tipo FileWriter. Para que permita continuar el archivo de texto en vez de reemplazarlo, al objeto FileWriter se le envía por parámetro la ruta del archivo y `true`, para indicar que se desea anexar y no reescribir. De esta forma, siempre que se escriba en el archivo las líneas quedarán anexadas al final.

Manejo de fechas

Todas las fechas, tanto en el reporte como en el log de excepciones, deben tener el siguiente formato: <día de la semana> <mes> <día> <hora> COT <año> (ej: Thu Aug 17 14:45:29 COT 2017).

Para poder crear un String de la fecha con esta formato debe utilizar el método toString, como se muestra a continuación:

```
String cadenaFecha = fecha.toString();
```

Serialización

Se desea persistir la información de la lista de carros, la lista de puestos, la hora actual del parqueadero, la tarifa actualmente manejada y el valor que hay en la caja. Recuerde que el orden en el que usted almacena los objetos en el archivo es el mismo en el que los debe recuperar al momento de cargarlo. El orden a seguir es: puestos, carros, hora actual, tarifa y valor en caja.

Por otra parte, el cast para leer objetos de tipo int (como la hora o la tarifa) o double (como el valor en caja) de un archivo serializado se realiza con las clases **Integer** y **Double**, no utilizando los tipos int y double.