Sesión 10. Trabajo con excepciones

Importa en Eclipse el Proyecto que puedes bajar del campus virtual para la sesión 10. Una parte de este proyecto ya está implementada.

Descripción

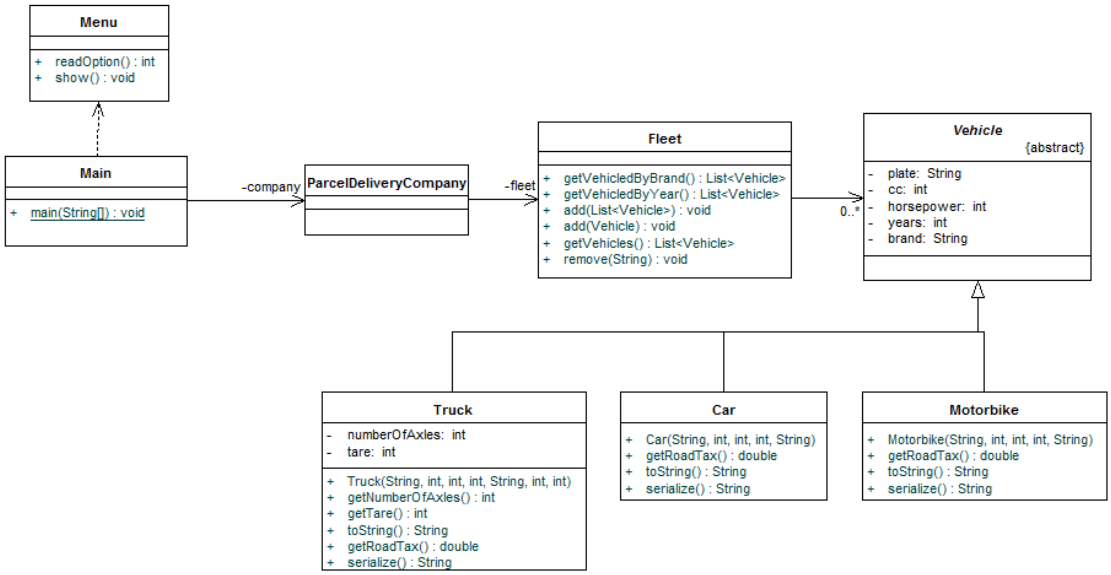
Una empresa de paquetería tiene una flota de vehículos. Puede haber tres tipos diferentes de vehículos: coches, motos y camiones. Todos tienen las siguientes propiedades que necesitan para su gestión:

- **plate**(string): el número de matrícula del vehículo.
- **cc** (integer): centímetros cúbicos del motor.
- **horsepower** (integer): potencia del motor.
- **years** (integer): número de años.
- **brand** (string): marca del vehículo (y algunas veces el modelo).

Los camiones pueden también tener:

- **numberOfAxles** (integer): número de ejes.
- **tare** (integer): peso.
-

El siguiente es el diagrama UML que representa el Proyecto



Se debe completar la aplicación para la empresa de paquetería. La aplicación ofrece un menú de usuario con las siguientes opciones:

```
1- Cargar un fichero
2- Añadir vehículo
3- Eliminar vehículo
4- Guardar en fichero
5- Importar de zip
6- Exportar a zip
7- Mostrar vehículos por marca
```

```
8- Mostrar vehículos por año
9- Calcular el impuesto de circulación total
0- Salir
```

1. Cargar un fichero: Carga vehículos desde un fichero de texto. El nombre del fichero se le pregunta al usuario.
2. Añadir un vehículo: Pregunta al usuario por toda la información necesaria para crear un nuevo vehículo.
3. Eliminar un vehículo. El programa pregunta al usuario por el número de matrícula. No pueden estar repetidas.
4. Guardar en fichero: Se le pide al usuario el nombre del fichero donde guardar.
5. Importar de zip: Similar a la opción 1, pero carga de un fichero comprimido .gz.
6. Exportar a zip: Similar a la opción 4 pero almacena la información en un fichero .gz
7. Mostar vehículos por marca: Muestra la lista de vehículos de la flota ordenados por marca.
8. Mostrar vehículos por año: muestra los vehículos de la flota ordenados por año.
9. La aplicación debe calcular el impuesto de circulación (el total) de todos los vehículos de la flota. Para ello se usa la siguiente fórmula:
   - Para motos, impuesto es 30 + cc × 0.5 + 10 × años
   - Para coches, impuesto is 60 + potencia + 7 × años + cc /12
   - Para camiones, impuesto es 100 + número de ejes × 10 + años × 20 + cc / 6

El fichero de texto de entrada contiene una línea por cada vehículo. Las líneas tienen una representación textual de los vehículos, separando cada campo con un carácter tabulador (`'\t'`). Se proporciona un ejemplo de fichero `vehicles.txt`. Este es el contenido:

Tipo, matrícula, centímetros cúbicos, potencia, años, marca (número de ejes y peso)

```
car	12345ABC	307	110	5	Ford-Focus
motorbike	2345BCD48	50	2	Honda
truck	3456CDE1900	310	4	Pegaso	2	9500
car	4567DEF307	140	3	Mercedes
motorbike	5678EFG50	75	1	Yamaha
car	6789FGH80	206	1	Renault-Twingo
motorbike	7890GHI48	80	4	Suzuki
truck	8901HIJ1300	150	7	Iveco	1	3500
```

Ejercicio

Se debe completar el programa en dos pasos:

1. Hacer que funcione la opción 1 de cargar los datos de un fichero.
2. Añadir control de errores y su tratamiento.

En esta sesión será una simulación de la carga del fichero. En la siguiente sesión de laboratorio añadiremos capacidades de entrada/salida.

Hacer que el programa funcione:

1- Revisa el código y ejecuta el programa. En primer lugar, debes familiarizarte con el código. El código compila y ejecuta, pero rompe la ejecución con excepciones NotYetImplementedExceptions.

2- Completa la clase VehicleParser para conseguir una lista de vehículos List<Vehicle> a partir de una lista de cadenas List<String> devuelta por la implementación actual (falsa) de FileUtil().

Añadir control de errores

Ahora ya tenemos una funcionalidad básica, pero no hay gestión de errores. Tenemos que preparar el código para manejar algunos errores relativos a cada operación. Cuando haya un error se debe escribir en la salida de log y si es posible se debe continuar.

- Añadir un vehículo cuyo número de matrícula ya existe.
  - El programa debe indicar al usuario el error y luego mostrar el menú.
- El usuario proporciona valores incorrectos cuando añade un nuevo vehículo.
  - Los campos numéricos (centímetros cúbicos, potencia, …) deben ser numéricos y no strings, los campos strings no deben estar vacíos, etc.
  - El programa debe dejar que el usuario lo intente de nuevo.
- Eliminar un vehículo cuya matrícula no existe.
  - El programa debe indicar al usuario el error y luego mostrar el menú.
- Analizar errores de carga del fichero (si una línea es inválida se ignora el vehículo):
  - Líneas en blanco
    - Se ignora la línea
  - Número de campos incorrecto
    - Se envía un mensaje al log y se ignora la línea
  - Campos en diferente orden
    - Se envía un mensaje al log y se ignora la línea
  - Formato del número erróneo para los campos numéricos
    - Se envía un mensaje al log y se ignora la línea

Tarea no presencial obligatoria

- Completa todo lo que no hayas acabado de lo que se pide para esta sesión 10 (opciones 1, 2, 3 y 9).
- Realiza la validación de datos para la entrada de usuario cuando se añade un nuevo vehículo (opción 2):
  - Todos los valores numéricos deben estar dentro de rango. En caso de error se reintenta.
  - Los valores cadena no pueden estar vacíos o null ni pueden tener más caracteres de un máximo para cada campo. En caso de error se reintenta.
  - El tipo del vehículo seleccionado debe ser uno de los tipos permitidos. En caso de error se reintenta.
- Añadir test para el método parse() de VehicleParser.
  - Comprobar casos positivos y negativos
  - Probar los casos en los que las excepciones (casos negativos) deben ser lanzadas.

- Las opciones 4, 5, 6 y 7, 8 (con ordenación) se realizarán en la próxima sesión.

Tarea no presencial optativa

Sustituir el uso de listas de Java por las propias listas implementadas en sesiones anteriores.