Examen de Programación
(Grados en Física y Matemáticas)

Septiembre 2018

Primera parte (1.25 puntos por cuestión, 50% nota del examen)

1) Escribir un método que, dado un texto en inglés (por tanto sin tildes) que se le pasa como parámetro, calcule y retorne el número de vocales (mayúsculas o minúsculas) que tiene.

   Pista: Recordar que la obtención de un carácter del string original se puede hacer con el método `charAt()`.

2) Se desea escribir un método Java para obtener el pago a realizar por hacer una reserva de una parcela en un camping que tiene la siguiente tarifa publicada:

PRECIO por día (IVA incluido)	TEMPORADA BAJA 23/03 – 29/03 03/04 – 21/06 10/09 – 23/09	TEMPORADA MEDIA (Semana Santa) 30/03 – 02/04 22/06 – 08/07 20/08 – 09/09	TEMPORADA ALTA 09/07 – 19/08
ADULTO	4,75 €	6,65 €	8,00 €
NIÑO (3 – 10 años)	2,65 €	2,65 €	4,20 €
PARCELA (1 caravana o tienda o campingcar, 1 coche y 1 conexión eléctrica)	9,55 €	16,35 €	37,00 €

   El método recibirá como parámetros el número de adultos, el número de niños, el número de días de estancia, y el texto `"Alta"`, `"Media"` o `"Baja"` que indica la temporada. El método retornará el pago a realizar como número real, o `Double.NaN` si el texto de la temporada es incorrecto.

   Se valora la eficiencia y el tamaño compacto del código.

3) Escribir un método iterativo que evalúa de forma aproximada la función integral senoidal usando el siguiente desarrollo en serie, válido para valores de x pequeños:

$$\int \frac{\sin x}{x} dx \cong \frac{x}{1} - \frac{x^3}{18} + \frac{x^5}{600} - \frac{x^7}{35280} + \frac{x^9}{3265920} - \frac{x^{11}}{439084800}$$

   El método recibe como parámetro el valor de x. Crear un array con los valores de los denominadores.

   Para hacer el cálculo más eficiente no utilizar la operación "*elevar a*". Basta observar que cada potencia de x se calcula como la anterior multiplicada por x^2 y que el signo se va alternando. Así, cada término i del desarrollo será `signo*potencia/denominador[i]`.

4) En un computador con sistema operativo Linux se desea escribir un *script* para copiar determinados archivos desde el disco duro a un *pen drive* situado en la carpeta `/mnt/disk1`. Los archivos a copiar se encuentran en dos directorios llamados `practica1` y `practica2`

situados dentro del directorio Proyectos que a su vez está en la carpeta del usuario. Cada uno de los dos directorios tiene los siguientes archivos:

- clases java con extensiones .java y .class
- documentos con extensión .html
- datos con extensión .xml

En primer lugar hacer un esquema de la distribución inicial de directorios y ficheros.

En segundo lugar escribir el *script*. Utilizar para los ficheros del disco duro rutas relativas desde la carpeta Proyectos, y para los ficheros del *pen drive* rutas absolutas. El *script* debe hacer los siguientes pasos:

- cambiar el directorio de trabajo poniéndolo en Proyectos
- crear en el *pen drive* dos carpetas llamadas programas y datos
- copiar en la nueva carpeta programas todos los ficheros .java y .class (de ambas prácticas)
- copiar en la nueva carpeta datos todos los ficheros (de ambas prácticas) cuyo nombre contenga la secuencia 201x, siendo x un solo carácter cualquiera, y que acabe en .xml
- mover a la nueva carpeta programas todos los ficheros (de ambas prácticas) cuyo nombre contenga la secuencia 201x, siendo x un solo carácter cualquiera, y que acabe en .html
- borrar de ambas prácticas todos los ficheros cuyo nombre contenga la secuencia 200x, siendo x un solo carácter cualquiera.

En tercer lugar hacer un esquema de la distribución final de carpetas y ficheros.

Examen de Programación
(Grados en Física y Matemáticas)

Septiembre 2018

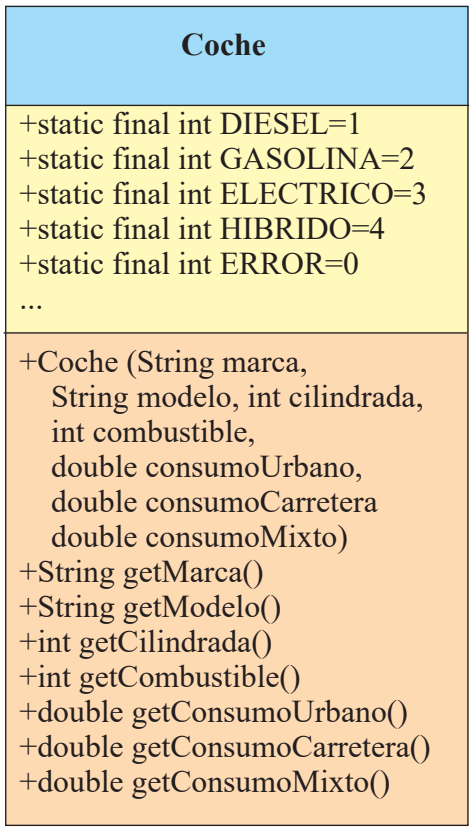
Segunda parte (5 puntos, 50% nota del examen)

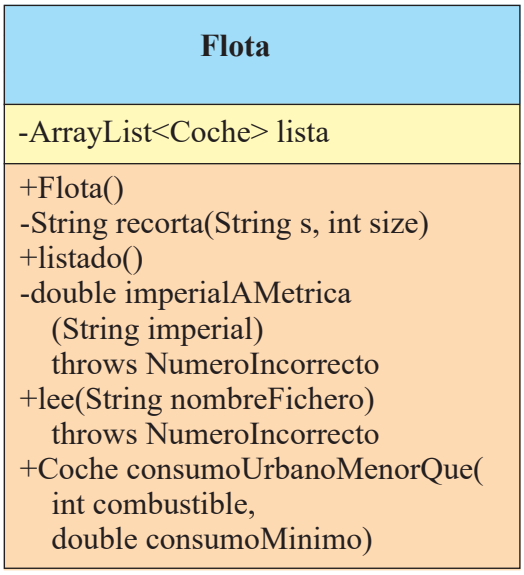
Se desea hacer parte del software para mantener los datos de una flota de coches. Se dispone para ello de un listado almacenado en un fichero de texto con formato "csv".

Se dispone de la clase Coche que ya está hecha y almacena los datos de un coche concreto. Se puede ver su diagrama de clase en la figura. La clase tiene un constructor al que se le pasan los valores iniciales de los atributos:

- la marca y modelo del coche
- la cilindrada (en cc)
- el combustible empleado (que debe ser una de las constantes DIESEL, GASOLINA, ELECTRICO o HIBRIDO, o puede ser ERROR si se ha producido un error al especificar el combustible)
- el consumo urbano, en carretera y mixto, en litros/100Km.

La clase dispone también de un método observador para cada atributo.





Se pide crear la clase Flota que almacena en el ArrayList lista el listado de coches, siendo cada uno un objeto de la clase Coche. Se pide también escribir un programa principal de prueba. La descripción de los métodos de la clase Flota es:

- *constructor*: Crea la lista vacía.

- recorta(): Si el String s es de tamaño menor o igual que size se retorna ese string. En otro caso se retornan los primeros caracteres de s, en número igual a size, reemplazando los tres últimos por tres puntos suspensivos: "...". Este método se supone que ya está hecho por lo que **no se pide** en este examen.
- listado(): Muestra en pantalla un listado de los coches. Antes de los datos se escribe un encabezamiento explicativo. Los datos aparecen en columnas, con un coche por línea. Las líneas están limitadas a 80 caracteres, por lo que si la marca o el modelo exceden 15 caracteres se recortarán en pantalla a 15 caracteres cada uno. Para ello contamos con el método recorta().

  Al final del listado se pondrá un texto explicando las posibles abreviaturas usadas en el encabezamiento así como las unidades de aquellas columnas que las tengan (cc, litros/100Km)
- imperialAMetrica(): Convierte un dato de consumo guardado en un String en formato español expresando medidas imperiales (millas por galón imperial) a número real de tipo double con la medida de consumo métrica (litros por 100 Km). Si el String imperial está vacío se retorna 0. Para esta conversión se tendrá en cuenta que una milla son 1.609344Km y que un galón imperial son 4.5461 litros.

  Pista: Para convertir el número en castellano de String a double puede crearse un Scanner con el texto como parámetro del constructor (*Ej*: Scanner scan=new Scanner(texto)). Luego puede leerse el número de este Scanner con nextDouble().

  La conversión puede fallar lanzando InputMismatchException si el texto no describe bien un número. Debe tratarse esta excepción lanzando otra excepción para avisar del error: NumeroIncorrecto. La nueva excepción debe lanzarse con un mensaje de error como parámetro (un String).

  Pseudocódigo de este método sin tener en cuenta el tratamiento de la excepción:

```
si imperial es igual a String vacío entonces
    retorna 0.0
si no
    Scanner scan=nuevo Scanner(imperial)
    configurar scan en castellano
    real consumo=scan.nextDouble() // en millas por galón
    retorna el consumo en litros/100Km
fin si
```

- lee(): Lee la lista de coches de un fichero de texto en formato "csv" cuyo nombre se pasa como parámetro. La primera línea del fichero es un encabezamiento explicativo que debe ignorarse. Luego hay un coche por línea, con sus datos separados por ";". Los datos son:
  - Marca;Modelo;Descripción;Cilindrada(cc);
  - Tipo de combustible: "Diesel", "Petrol", "Electricity", "Petrol Hybrid";
  - consumo urbano(imperial); consumo en carretera(imperial); consumo mixto(imperial);
  - emisiones de CO2(g/Km); emisiones de CO(mg/Km); emisiones de NOx(mg/Km); emisiones de partículas(mg/Km)

  Los datos de cilindrada y emisiones son números enteros. Los datos de cilindrada, consumo y emisiones pueden estar ausentes, en cuyo caso son strings vacíos (de cero caracteres). Si la cilindrada o un consumo están vacíos se pondrán a cero en la lista.

  La descripción y los datos de emisiones se deben ignorar pues no se usan.

El combustible debe traducirse de su texto en inglés a su constante entera definida en la clase Coche. Si hay un error en el texto se pondrá el valor ERROR.

Los datos de consumo son números reales en notación en español (con coma decimal) en unidades imperiales: millas por galón imperial. Hay que transformarlos a litros/100Km. Para ello disponemos del método imperialAMetrica(), que ya tiene en cuenta el caso de que el consumo sea un String vacío. Este método puede lanzar NumeroIncorrecto si el número real es incorrecto. Esta excepción se debe propagar sin ser tratada aquí.

La conversión de un String s con la cilindrada a número entero se puede hacer con el método Integer.parseInt(String s) que retorna el número entero. Este método puede lanzar NumberFormatException si el número es incorrecto. En el tratamiento de esta excepción debe lanzarse NumeroIncorrecto con un mensaje de error (String) como parámetro.

Ejemplo de fichero:

```
Manuf.;Model;Descr.;cc;Fuel;Imperial Urban;Imperial Extra-Urban;Imperial ...
ALFA ROMEO;Series 3;1.3 JTDm-2;1248;Diesel;65,7;97,4;83,1;89;234;67;0,27;;;;;;
BMW;1 Series;116d ED Plus 16'' tyres;1496;Diesel;72,4;91,1;83,1;89;119;37;0,14;;;;;;
BMW;1 Series;116d ED Plus 16'' tyres;1496;Diesel;72,4;91,1;83,1;89;119;37;0,14;;;;;;
CITROEN;C4;BlueHDi 100 S&S;1560;Diesel;76,3;91,1;85,6;86;272;55;0,42;;;;;;
...
```

Pista: recordar que si se dispone de un String s, el método s.split(";") retorna un array de Strings con todos los trozos de s que estén separados por ";".

- consumoUrbanoMenorQue(): Busca en la lista el primer coche del tipo de combustible indicado, cuyo consumo urbano es menor que el parámetro consumoMinimo y lo retorna. Si no se encuentra ninguno con esas condiciones se retorna null.

La excepción NumeroIncorrecto ya está definida en el proyecto en una clase aparte.

Finalmente, se pide hacer un programa principal en una clase aparte que haga lo siguiente:

a. Crea un objeto de la clase Flota.

b. Lee el fichero CSV llamado datos-coches-nuevos.csv.

c. Hace un listado de los coches, con el método listado.

d. Muestra en pantalla la marca, modelo y cilindrada del primer coche DIESEL cuyo consumo urbano es menor que 4.0 litros/100Km o un mensaje de error si no hay ninguno.

Tratamiento de errores:

- Si en el paso b) se lanzase NumeroIncorrecto se pondrá un mensaje en pantalla y se continúa con los pasos c) y d).

Valoración:

- encabezamiento de la clase, atributos y constructor: 0.5 puntos
- métodos listado(), imperialAMetrica(), consumoUrbanoMenorQue() y programa principal: 0.75 puntos cada uno
- método lee(): 1.5 puntos