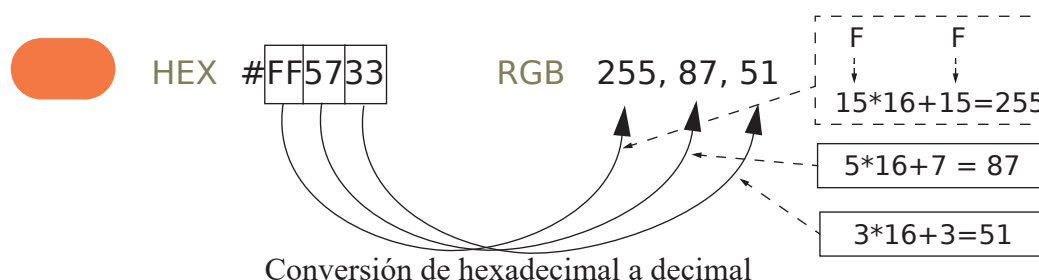


Examen de Programación (Grados en Física y Matemáticas)

Junio 2018

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

- 1) Los colores se pueden representar mediante una combinación de tres números que representan la cantidad de luz roja, verde o azul. Es habitual usar para ello tres números enteros entre 0 y 255 (formato RGB). También es posible representar un color mediante números hexadecimales de 6 cifras (formato HEX), donde las dos primeras cifras representan el rojo, las dos siguientes el verde y las dos últimas el azul.



Se pide escribir un método al que le pasamos un String con un color en su representación HEX, con un símbolo "#" seguido de 6 cifras hexadecimales representadas por valores de 0 a 9 y letras de la A a la F, respectivamente para los valores 10 a 15. El método debe obtener tres números enteros con la conversión a decimal de cada grupo de dos cifras, y retornarlos en un array.

La conversión de un número de dos cifras se hace multiplicando el valor numérico de la primera cifra (que está entre 0 y 15) por 16 y sumando el valor numérico de la segunda cifra.

La conversión de una cifra hexadecimal a valor numérico se puede hacer con este método ya disponible:

```
public static int toDecimal(char cifra) // No se pide este método
```

La cabecera del método que se pide será:

```
public static int[] hexToRgb(String hex)
```

En resumen, este método debe convertir las dos primeras cifras hexadecimales en un número entero (multiplicando la primera cifra convertida a decimal por 16 y sumando la segunda cifra en decimal), y repetir esto para las dos siguientes cifras y las dos últimas. Con ello se obtendrán tres números enteros que hay que retornar en un array.

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

- 2) La siguiente tabla muestra la tarifa de agua vigente en Santander:

Consumo	Abastecimiento	Alcantarillado
Cuota fija	11,35 €	5,68 €
Desde 0 hasta 15 m ³	0,1445 €/m ³	0,0723 €/m ³

Consumo	Abastecimiento	Alcantarillado
Desde 16 hasta 30 m ³	0,1858 €/m ³	0,0929 €/m ³
Desde 31 hasta 45 m ³	0,8256 €/m ³	0,4129 €/m ³
Desde 46 hasta 100 m ³	1,1869 €/m ³	0,5935 €/m ³
Más de 100 m ³	1,2901 €/m ³	0,6450 €/m ³

Las tarifas de abastecimiento y alcantarillado se hallan guardadas en sendos arrays estáticos:

```
static double[] abastecimiento={11.35, 0.1445, 0.1858, 0.8256, 1.1869, 1.2901};
static double[] alcantarillado={5.68, 0.0723, 0.0929, 0.4129, 0.5935, 0.6450};
```

Escribir un método estático al que se le pase el consumo de agua realizado por un hogar en m³ (número entero) y retorne el pago a realizar (número real), que será la suma del abastecimiento más el alcantarillado (cuota fija + gasto por consumo en ambos), añadiendo además el 10% del IVA vigente. Se puede suponer que el consumo nunca es negativo.

- 3) Se dispone de un ArrayList de nombres (Strings) que se considera muy grande, por lo que se desea obtener dos ArrayList de tamaño mitad. Se pide escribir un método al que se le pasan como parámetros la lista original y dos listas vacías. El método rellenará la primera lista vacía con los elementos de la primera mitad de la lista original y la segunda lista con los de la segunda mitad. Si el número de valores de la lista original es impar el valor central se incluirá en la primera mitad.
- 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 proyecto1 y proyecto2 situados dentro del directorio Documentos que a su vez está en la carpeta del usuario. Cada uno de los dos directorios tiene a su vez los siguientes directorios:
 - src: contiene archivos con clases java acabados en .java y .class
 - doc: contiene documentos con extensión .html
 - data: contiene datos con extensión .csv

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 Documentos, y para los ficheros del *pen drive* rutas absolutas. El *script* debe hacer los siguientes pasos:

- cambiar el directorio de trabajo poniéndolo en Documentos
- crear en el *pen drive* dos carpetas llamadas programas y datos
- copiar en la nueva carpeta programas todos los ficheros .java y .class (de ambos proyectos) cuyo nombre comience por h
- copiar en la nueva carpeta datos todos los ficheros (de ambos proyectos) cuyo nombre contenga la secuencia 2018 y acabe en .csv
- mover a la nueva carpeta programas todos los ficheros de las carpetas doc (de ambos proyectos) cuyo nombre contenga la secuencia 2018 y acabe en .html
- borrar de las carpetas data y doc (de ambos proyectos) todos los ficheros cuyo nombre contenga la secuencia 2017

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

Examen de Programación (Grados en Física y Matemáticas)

Junio 2018

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

Se desea hacer parte del software para mantener un catálogo de árboles singulares. Se dispone para ello del catálogo almacenado en un fichero de texto con formato "csv".

Se dispone de la clase Arbol que ya está hecha y almacena los datos de un árbol 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: lugar, circunferencia del tronco a 1.30m de altura (en m), altura (en m), especie, nombre, e identificador en el catálogo de árboles singulares de Cantabria. Hay datos que pueden ser desconocidos, en cuyo caso aparecen almacenados de esta forma:

- circunferencia o altura: Double.NaN
- nombre: "?"
- identificador: -1

La clase dispone también de un observador para cada atributo.

Arbol	Catalogo
-String lugar -double circunferencia, altura -String especie -String nombre -int id	-ArrayList<Arbol> lista
+Arbol (String lugar double circunferencia, double altura, String especie, String nombre, int id) +String getLugar() +double getCircunferencia() +double getAltura() +String getEspecie() +String getNombre() +int getId()	+Catalogo() -double convierteNumero(String texto) throws NumeroIncorrecto -String recorta(String s, int size) +lee(String nombreFichero) throws NumeroIncorrecto +listado() +muestraArbol(String nombre)

Se pide crear la clase Catalogo que almacena en el ArrayList lista el catálogo de árboles, siendo cada uno un objeto de la clase Arbol. Se pide también escribir un programa principal de prueba. La descripción de los métodos de la clase Catalogo es:

- *constructor*: Crea la lista vacía.
- *convierteNumero()*: Convierte un número real guardado en el parámetro texto a número real. El texto puede ser igual a "?" para indicar un dato desconocido, y en ese caso se retornará Double.NaN. En otro caso el texto contiene dos palabras, una con el número en formato castellano (con coma decimal) y otra con las unidades. Por ejemplo "13,4 m". En este caso se retornará la primera palabra del texto convertida a número (las unidades se ignoran).

Pista: Para resolver el caso en el que el texto no es "?" 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, poniendo en pantalla un mensaje de error que incluya el texto incorrecto y posteriormente lanzando la excepción NumeroIncorrecto, para avisar del error.

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

```

si texto es igual a "?" entonces
    retorna Double.NaN
si no
    Scanner scan=nuevo Scanner(texto)
    configurar scan en castellano
    retorna scan.nextDouble()
fin si

```

- 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, seguidos de tres puntos suspensivos: "..."
- lee(): Lee la lista de árboles 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 árbol por línea, con sus datos separados por ";". Los datos son: Localización; Circunferencia (a 1,3 m); Altura; Especie; Nombre; Número.

Algunos datos pueden ser desconocidos, en cuyo caso aparecen como:

- circunferencia, altura o nombre: "?"
- identificador: "-"

Los datos de circunferencia y altura, cuando son conocidos, son números reales en notación en español (con coma decimal) seguidos de un espacio en blanco y las unidades. Ejemplo: "13,4 m". Para convertir un texto de este tipo a número real disponemos del método convierteNumero(), que ya tiene en cuenta el caso de que el número sea desconocido. Este método puede lanzar NumeroIncorrecto si el número real es incorrecto. Esta excepción se debe propagar sin ser tratada aquí.

Ejemplo de fichero:

```

Localización;Circunferencia (a 1,3 m);Altura;Especie;Nombre;Número
Ojedo, Pista a San Tirso;13,75 m;11,5 m;Castaño (Castanea sativa);La Narezona;3
Ruento, Monte Aa, nº 10;12,3 m;26 m;Roble, cagiga (Quercus robur L.);?;26
Ruento (Ucieda);12,20 m;13 m;Roble común (Quercus robur);El roble gordo;36
La Parte (Pesaguero);10,1 m;25 m;Castaño (Castanea sativa Miller);El Abuelo;213
...

```

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 ";".

- listado(): Muestra en pantalla un listado de los árboles. Antes de los datos se escribe un encabezamiento explicativo. Los datos aparecen en columnas, con un árbol por línea. Las líneas están limitadas a 80 caracteres, por los que si el lugar, la especie o el nombre exceden de 16 caracteres se recortarán en pantalla a 16 caracteres cada uno, añadiendo tres puntos suspensivos si se han recortado (total 19 caracteres cada uno). Para ello contamos con el método recorta().

- `muestraArbol()`: Muestra en pantalla los datos del árbol cuyo nombre se indica. Primero busca en la lista el árbol cuyo nombre se indica. Si no se encuentra, pone "Árbol no encontrado". Si se encuentra, pone todos los datos del árbol, uno por línea, cada uno precedido de una etiqueta explicativa.

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

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

- a. Crear un objeto de la clase `Catalogo`.
- b. Lee el fichero CSV llamado `arboles.csv`.
- c. Hace un listado de los árboles, con el método `listado`.
- d. Muestra en pantalla los datos del árbol "La Narezona".

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
- método `recorta()`: 0.5 puntos
- métodos `convierteNumero()`, `listado()`, `muestraArbol()` y programa principal: 0.75 puntos cada uno
- método `lee()`: 1 punto