

ACCÉS A DADES

(2n DAM)

Unitat 1. Exercicis sobre Fitxers



Contents

- 1. Sistema de fitxers.....3
 - 1.1. ls -la..... 3
- 2. Arxius de Text.....4
 - 2.1. Quantes vocals?..... 4
 - 2.2. Tail i Head.....4
- 3. Arxius XML.....4
- 4. Arxius JSON.....6

1. Sistema de fitxers

1.1. ls -la

En aquest exercici implementarem una versió del comandament **ls**. Llistarà un directori en un mode de visualització específic. Tant el directori com la vista s'indicaran quan iniciem el programa. El mode de visualització pot ser llista, columnes o taula:

- **List** → Llistarà el nom dels fitxers (i directoris), un en cada línia.
- **Columns** → El mateix que *List* però en diverses columnes.
- **Table** → Es mostrarà la informació del fitxer amb el patró: **DFRWH nom tamany permisos**:
 - **D** → it's a directory
 - **F** → it's a file
 - **R** → we can read the file
 - **W** → we can write the file
 - **H** → the file is hidden

Lliurament: Un arxiu `.java` amb el programa. Per ajudar-vos amb alguna tasca, ací teniu un codi per formatar una llista de cadenes en columnes. La constant `MAX_FILES_BY_COLUMN` ha d'inicialitzar-se a `4`.

```
public void listaColumnas(String[] filenames) {
    int columnas = (filenames.length / MAX_FILES_BY_COLUMN) + 1;

    String[][] salida = new String[MAX_FILES_BY_COLUMN][columnas];

    // Llenar la matriz 'salida' con los nombres de archivo
    for (int i = 0; i < filenames.length; i++) {
        salida[i % MAX_FILES_BY_COLUMN][i / MAX_FILES_BY_COLUMN] = filenames[i];
    }

    // Encontrar el nombre de archivo más largo para establecer el ancho de columna
    int maxLength = 0;
    for (String filename : filenames) {
        if (filename.length() > maxLength) {
            maxLength = filename.length();
        }
    }

    // Ajustar el formato para que cada columna tenga el mismo ancho
    String format = "%-" + (maxLength + 2) + "s";

    // Bucle para mostrar salida con formato simétrico
    for (int i = 0; i < MAX_FILES_BY_COLUMN; i++) {
        for (int j = 0; j < columnas; j++) {
            if (salida[i][j] != null) {
                System.out.printf(format, salida[i][j]); // printf para aplicar el formato
            }
        }
        System.out.println();
    }
}
```

2. Arxius de Text


2.1. Quantes vocals?

Escriu un programa que reba el nom d'un fitxer de text i una vocal. El programa mostrarà el nombre de vegades que apareix aquesta vocal en el fitxer.

2.2. Tail i Head

Escriu la teua pròpia versió de **tail** and **head** de *gnu utils*. Podeu trobar informació a:

- [tail](#)
- [head](#)

 Recordeu que aquests programes mostren les deu primeres i últimes línies per defecte. Si voleu modificar-lo, podeu ajustar amb **-nX** on X és el nombre de línies que voleu mostrar.

3. Arxius XML

A la plataforma trobaràs un fitxer anomenat [monaco2017.xml](#) conté molta informació sobre la cursa del campionat de Fórmula1 de l'any 2017 a Mònaco. Veiem un extracte:

```
1 <Result number="5" position="1" positionText="1" points="25">
2   <Driver driverId="vettel" code="VET" url="http://en.wikipedia.org/
   wiki/Sebastian_Vettel">
3     <PermanentNumber>5</PermanentNumber>
4     <GivenName>Sebastian</GivenName>
5     <FamilyName>Vettel</FamilyName>
6     <DateOfBirth>1987-07-03</DateOfBirth>
7     <Nationality>German</Nationality>
8   </Driver>
9   <Constructor constructorId="ferrari" url="http://en.wikipedia.org/
   wiki/Scuderia_Ferrari">
10     <Name>Ferrari</Name>
11     <Nationality>Italian</Nationality>
12   </Constructor>
13   <Grid>2</Grid>
14   <Laps>78</Laps>
15   <Status statusId="1">Finished</Status>
16   <Time millis="6284340">1:44:44.340</Time>
17   <FastestLap rank="2" lap="38">
18     <Time>1:15.238</Time>
19     <AverageSpeed units="kph">159.669</AverageSpeed>
20   </FastestLap>
21 </Result>
```

A banda d'altra informació, dins del [Result](#) trobarem com a atributs el número del pilot (atribut [number](#)) i la posició en què va acabar la cursa ([position](#)). A més, trobarem els següents elements:

- [Driver](#): informació sobre qui és el conductor
- [Manufacturer](#): informació sobre la marca o constructor del cotxe
- [Grid](#): posició en que el conductor ha començat la cursa
- [Laps](#): voltes completades

- **Status**: que indica amb l'atribut `statusID="1"` que el pilot va acabar la cursa. Altre valor indica que no l'ha acabada.
- **Time**: que indica en l'atribut de `millis` quant de temps va trigar a completar la cursa (en ms), i en el seu valor expressat en hores, minuts i segons.
- **FastesLap**: que indica la classificació respecte a la volta ràpida en el seu atribut de `rank`.

En aules pot trobar:

- classe **Driver**: implementada completament. Aquesta necessita la part de l'XML corresponent a un conductor (Element driver) per crear un objecte *Driver*.
- classe **ResultadoCarrera**: parcialment implementada, amb els següents atributs:

```
1 private Driver d;  
2 private String constructor;  
3 private int initialPos;  
4 private int finalPos;  
5 private long timeMillis;  
6 private int completedLaps;  
7 private int rankFastesLap;  
8 private boolean finisher;
```

Completa el programa, afegint:

- `Constructor(Element result)`, que rep un Element XML.
- `public ArrayList<ResultadoCarrera> carregaResultadosXML(String nomXML)` el qual rep el nom del fitxer XML i carrega els resultats.

4. Arxius JSON

A la plataforma trobareu un fitxer anomenat **StarWars.json**. Recupera informació sobre personatges de la famosa saga. Escriu un programa per obtenir:

- Personatges que no conduïen cap vehicle.
- Llista de personatges, ordenats per nombre de pel·lícules en les qual apareix.
- Crea un fitxer XML com aquest, amb una breu informació resumida:

```
<character films="4" vehicles="0">  
  <name>Luke Skywalker</name>  
  <mass>77</mass>  
  <url>https://swapi.dev/api/people/1/</url>  
</character>
```