

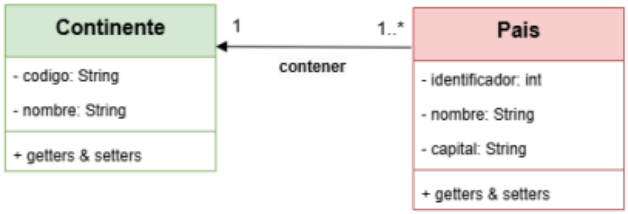
ACTIVIDAD 1.2 - Tratamiento fichero txt con orientación a objetos

El objetivo de la actividad es realizar labores de tratamiento de un fichero con extensión txt haciendo uso de las clases del paquete java.io de JAVA almacenando la información en objetos y tratándolos.

▼ Tratamiento de un fichero con extensión “txt” - objetos

Dado el fichero **informacion_paises.txt**, situado en el directorio descrito, se pide crear un proyecto **Ficheros_02** JAVA. Un programa que incluya, además del método main, un método llamado **imprimirInformacionPaises,** que recorra el fichero, tratando únicamente la información referente a los países. Las líneas del fichero que comienzan por # no se imprimen.

Para realizar el tratamiento, será necesario que trabajes con la siguiente estructura de clases “tipo bean” asociada a las entidades que se describen en el siguiente diagrama de clases:



El resultado a imprimir, para cada país, será algo similar a esto:

```
Pais [identificador=93, nombre=Arabia Saudita, capital=Riad,
continente=Continente [codigo=03, nombre=null]]
INSERT INTO T_PAIS (cod_continente, identificador, capital,
nombre_pais) VALUES (93, '03', 'Riad', 'Arabia Saudita');
```

▼ Resolución

```
package org.palomafp.aadd.rmr;

import java.io.FileNotFoundException;
import java.io.IOException;

import org.palomafp.aadd.rmr.vo.Continente;
import org.palomafp.aadd.rmr.vo.Pais;

import java.io.FileReader;
import java.io.BufferedReader;

public class Main {

    // este método se ejecuta al arrancar el programa
    public static void main (String[] args) {

        // llamo al método que va a procesar mi fichero
        imprimirInformacionPaises();
    }


    public static void imprimirInformacionPaises() {
        // el codiguito va aquí
        try {

            // preparo para leer el archivo de la ruta
            BufferedReader br = new BufferedReader (new FileReader("C:/User
s/Rocío/aadd/fic02/informacion_paises.txt"));
            String linea = ""; // guardo cada línea aquí

            // bucle donde leo línea a línea hasta que no haya más
            while ((linea = br.readLine()) != null) {

                // si la lóinea mepieza por # o - la ignoro y no hace nada
                if (linea.startsWith("#") || linea.startsWith("-")) {

                } else {
                    // paso 1. limpio los datos
                    linea = linea.replace("\"", ""); // quito todas las com
illas dobles
                    String[] trozos = linea.split(","); // parto la línea p
or las comas para tener lso 4 datos
```

```
                    // paso 2. meto los datos en objetos
                    Continente c = new Continente(); // creo el continente
                    c.setCodigo(trozos[0].trim()); // guardo el code quitán
dole los espacios

                    Pais p = new Pais();
                    p.setIdentificador(Integer.parseInt(trozos[1].trim()));
// convierto a num el identificador
                    p.setNombre(trozos[2].trim()); // meto el nombre
                    p.setCapital(trozos[3].trim()); // meto la capital
                    p.setContinente(c); // mteo el objeto continente dentro
de país

                    // paso 3. imprimo l oque se ipde
                    System.out.println(p.toString());

                    // paso 4. monto la frase sql sacando los datos de los
objetos que he creado arriba
                    System.out.println("INSERT INTO T_PAIS (cod_continente,
identificador, capital, nombre_pais) VALUES (" + p.getIdentificador() + ",
'" + c.getCodigo() + "', '" + p.getCapital() + "', '" + p.getNombre() +
"');");
                }
            }

            br.close();

        } catch (FileNotFoundException e) {
            System.out.println("El archivo no se encuentra.");
        } catch (IOException e) {
            System.out.println("Ha ocurrido un error.");
        }
    }

}
```

Estructura de paquetes y clases

Para mantener el código ordenado y seguir las buenas prácticas, he dividido el proyecto en dos paquetes:

- **Paquete base** (`org.palomafp.aadd.rmr`): aquí se encuentra la clase `Main.java` que contiene la lógica de la lectura y ejecución,
- **Paquete VO** (`org.palomafp.aadd.rmr.vo`): contiene las clases de datos o *beans* (`Continente.java` y `Pais.java`)

`.vo` es "Value Object". Es un estándar para organizar las clases cuya única responsabilidad es almacenar y transportar datos. Estas clases deben ser sencillas y tener únicamente los atributos, métodos constructores, getters y setters.

Diseño de objetos

- **Continente:** tiene los atributos código y nombre.
- **País:** tiene los atributos idefntificador, nombre y capital. Además, para representar la relación de composición, contiene un atributo de tipo `Continente`.

A ambas clases les he añadido el método `toString()` para facilitar la impresión de su estado.

La inclusión del objeto `Continente` como atributo dentro de `Pais` es fundamental (por la relación de composición). Gracias a esto cuando llamo al `toString()` de `Pais`, este llama internamente al `toString()` de `Continente`, generando de forma automática la salida anidada exacta que pide el enunciado.

Lectura del fichero de texto

En el main, he creado el método `imprimirInformacionPaises()`. Para acceder al fichero, he utilizado las clases `FileReader` y `BufferedReader`, dentro de un bloque try-catch para gestionar las excepciones que pueda disparar.

La lectura la he hecho con un bucle `while` que comprueba si la línea leída es distinta de nulo. Para ignorar las cabeceras, he puesto un condicional `if` que ignora cualquier línea que comience por `#` o `-`.

El bloque de error es obligatorio porque las clases de `java.io` generan excepciones comprobadas. En cuanto a la lectura, `readLine()` devuelve `null` cuando alcanza el final del fichero, lo que sirve como condición para detener el bucle.

Tratamiento y extracción de datos

Al tener una línea válida, el proceso de tratamiento de datos es el siguiente:

1. **Limpio:** elimino todas las comillas dobles usando `linea.replace(\"", "")`.
2. **Fragmento:** divido la línea utilizando la coma como separador mediante `linea.split(",")`, obteniendo un array de 4 posiciones.
3. **Instancio:** creo un objeto `Continente` (dándole la pos 0 del array) y un objeto `Pais` (dándole las posiciones 1, 2 y 3). He utilizado `Integer.parseInt()` para convertir el identificador a número entero, y el método `.trim()` para eliminar los espacios en blanco que haya.

Con el replace antes del split limpio todos los caracteres de una.

Salida por consola

Con los objetos ya instanciados y relacionados (`p.setContinente(c)`), imprimo los dos resultados:

1. **El estado del objeto:** usando un Syso que llama al toString de país.
2. **La instrucción SQL:** concatenando el texto fijo con las propiedades de los objetos (getters), añadiendo las comillas simples de forma manual en la cadena.