Curs 2020-2021

Exercicis Tema 1. Fitxers

Joan Gerard Camarena Estruch

Accés a Dades

Ejercicios

IES Jaume II El Just
Tavernes de la Valldigna

Continguts

1 Sistemas de Ficheros. 3

2 Persistencia en `WarShips` 3

3 Ficheros de acceso aleatorio. Empleados 5

4 Serialización. Empleados 7

5 XML 8

6 JSON 8

1 Sistemas de Ficheros.

La aplicación debe listar un directorio en un modo de visualización concreto. Tanto el directorio como el modo de visualización se pasarán como parámetros:

- Los modos serán: lista, columnas y tabla:
  - Lista: los nombres de archivos uno debajo del otro
  - Columnas: Las columnas serán de 5 filas.
  - Tabla: Se mostrará información de cada fichero: `DFRWH nombre tamaño fecha-modificación`, donde:
    * D → Se trata de un directorio
    * F → Se trata de un fichero
    * R → Se puede leer sobre dicho fichero
    * W → Se puede escribir sobre dicho fichero
    * H → El fichero está oculto (Hidden)

Se entregará un único fichero java. Como ayuda os paso procedimiento que muestra una lista de cadenas en formato de columnas. Como ayuda para el modo de vista de columnas (la constante `MAX_FILES_BY_COLUMN = 4`):

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

    for (int i=0;i<filenames.length;i++){
        salida[i % MAX_FILES_BY_COLUMN][i / MAX_FILES_BY_COLUMN]=filenames
            [i];
    }

    //bucle para mostrar salidals
    for (int i=0;i<MAX_FILES_BY_COLUMN;i++){
        for (int j=0; j<columnas;j++)
            System.out.print(salida[i][j] + " - ");
        System.out.println(" /");
    }
}
```

2 Persistencia en `WarShips`

Aprovechando la aplicación guerra de barcos creada anteriormente, le añadiremos las funcionalidades necesarias para que tenga persistencia. Para ellos utilizaremos las clases de flujo de caracteres `FileReader`, `FileWriter` y sus descendientes. Deberemos:

1. Realizar los cambios necesarios para guardar (y cargar) la configuración del tablero en un fichero de propiedades con el siguiente formato. El fichero debe llamarse `warship.propierties`:

```
board_tam=10
num_boats=5
max_jugadas=50
```

2. Realiza los cambios necesarios para que la aplicación guarde los barcos en un fichero de texto: `boat_out.txt`. El formato será: `boat;dimension;direccion;fila;columna` donde la dimensión será un entero que indicará la dimensión. La dirección será 0 horizontal y 1 vertical. La fila y la columna enteros que indican la posición. Por ejemplo seria

```
0;5;0;8;0
1;4;0;4;1
2;3;0;2;5
3;2;1;4;9
4;2;0;9;6
5;2;0;2;0
```

3. Realiza los cambios necesarios para que las jugadas se guarden en un fichero: `moviments_out.txt`. El formato será: `numero_mov;fila;columna;resultado` donde número será el número de movimiento, la fila y la columna de la celda donde se tira la bomba y el resultado un entero que indicará 0 si agua, 1 tocado, 2 hundido, 3 error (fuera del tablero). Por ejemplo seria:

```
1;0;3;2
2;0;2;3
3;9;5;2
4;9;6;2
5;9;7;2
...
18;9;2;3
19;5;5;2
20;5;6;2
21;5;7;2
22;5;8;3
```

4. Una vez terminados a y b. Renombar los ficheros por `boat_in.txt` y `moviments_in.txt` respectivamente. Crea una segunda aplicación a partir de la primera que situe los barcos segun el fichero `boat_in.txt` y realice los movimientos según el contenido del fichero `moviments_in.txt`.

3 Ficheros de acceso aleatorio. Empleados

Este bloque está basado en los ejercicios de las páginas 21 y 22 del libro *Acceso a Datos* de la Editorial **Garceta**, en los cuales se estudia el acceso aleatorio a datos.

A partir de los programas `generarFicheroAleatorio.java` y `mostrarFicheroAleatorio.java` :

```
public static void generarFicheroAleatorio() {
        try {
            File fichero = new File("AleatorioEmple.dat");
            //declara el fichero de acceso aleatorio
            RandomAccessFile file = new RandomAccessFile(fichero, "rw");
            //arrays con los datos
            String apellido[] = {"FERNANDEZ", "GIL", "LOPEZ", "RAMOS",
                "SEVILLA", "CASILLA", "REY"};//apellidos
            int dep[] = {10, 20, 10, 10, 30, 30, 20};       //
               departamentos
            Double salario[] = {1000.45, 2400.60, 3000.0, 1500.56,
                2200.0, 1435.87, 2000.0};//salarios

            StringBuffer buffer = null;//buffer para almacenar apellido
            int n = apellido.length;//numero de elementos del array

            for (int i = 0; i < n; i++) { //recorro los arrays
                file.writeInt(i + 1); //uso i+1 para identificar empleado
                buffer = new StringBuffer(apellido[i]);
                buffer.setLength(10); //10 caracteres para el apellido
                file.writeChars(buffer.toString());//insertar apellido
                file.writeInt(dep[i]);       //insertar departamento
                file.writeDouble(salario[i]);//insertar salario
            }
            file.close();  //cerrar fichero
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Exe3.class.getName()).log(Level.SEVERE, null,
                ex);
        } catch (IOException ex) {
            Logger.getLogger(Exe3.class.getName()).log(Level.SEVERE, null,
                ex);
        }
    }
```

```
  public static void mostrarFicheroAleatorio() {
        try {
            File fichero = new File("AleatorioEmple.dat");
            //declara el fichero de acceso aleatorio
```

```
            RandomAccessFile file = new RandomAccessFile(fichero, "r");
            int id, dep, posicion;
            Double salario;
            char apellido[] = new char[10], aux;
            posicion = 0;  //para situarnos al principio
            for (;;) {  //recorro el fichero
                file.seek(posicion); //nos posicionamos en posicion
                id = file.readInt();   // obtengo id de empleado

                //recorro uno a uno los caracteres del apellido
                for (int i = 0; i < apellido.length; i++) {
                    aux = file.readChar();
                    apellido[i] = aux;    //los voy guardando en el array
                }

                //convierto a String el array
                String apellidos = new String(apellido);
                dep = file.readInt();        //obtengo dep
                salario = file.readDouble(); //obtengo salario

                if (id > 0) {
                    System.out.printf("ID: %s, Apellido: %10s,
                       Departamento: %d, Salario: %.2f %n",
                            id, apellidos.trim(), dep, salario);
                }

                //me posiciono para el sig empleado, cada empleado ocupa
                   36 bytes
                posicion = posicion + 36;

                //Si he recorrido todos los bytes salgo del for
                if (file.getFilePointer() == file.length()) {
                    break;
                }

            }//fin bucle for
            file.close();  //cerrar fichero
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Exe3.class.getName()).log(Level.SEVERE, null,
                ex);
        } catch (IOException ex) {
            Logger.getLogger(Exe3.class.getName()).log(Level.SEVERE, null,
                ex);
        }
    }
```

se pide crear un sólo programa para las Actividades 3 y 4 de las páginas 23 y 24 respectivamente. El programa presentará un menú con las siguientes opciones:

1. Búsqueda de datos por ID (actividad 3):

```
Dime el ID del empleado a buscar: 6
ID: 6, Apellido:    CASILLA, Departamento: 30, Salario: 1435,87
```

2. Actualización de salario por ID (actividad 4):

```
Dime el ID del empleado a modificar el salario: 4
Dime la modificación del salario: 100
ID: 4, Apellido:      RAMOS, Departamento: 10, Salario: 1800,56
```

3. Calcular salarios por departamento:

```
Dime el número del departamento: 10
ID: 1, Apellido:  FERNANDEZ, Departamento: 10, Salario: 1000,45
ID: 3, Apellido:      LOPEZ, Departamento: 10, Salario: 3000,00
ID: 4, Apellido:      RAMOS, Departamento: 10, Salario: 1700,56
El salario total del departamento 10 es 5701,01
```

La opción 3 del menú (no está en el libro). Mostrará los empleados de dicho departamento y calculará la suma de los salarios de un departamento que se pide.

4 Serialización. Empleados

Vamos a aprovechar el código del ejercicio anterior para utilizar la POO y la serialización de objetos. Para ello se pide:

1. Crear una clase `Empleado` para almacenar los datos anteriores. Modificaremos un poco la estructura para almacenar el apellido como un String.
2. Realizar una función denominada `importarFichero` que lee del fichero aleatorio anterior y lo carga en un array de `Empleado`.
3. Realizar una función que muestre por pantalla los empleados (en memoria).
4. Realizar un método que permita añadir un empleado a la colección de empleados que están en memoria. No repetir el identificador !!
5. Método para eliminar un empleado, pidiendo el ID al usuario
6. Método que guarde los objetos en memoria en un fichero de objetos, de nombre `ObjectEmple.dat`
7. Método que cargue dicho fichero de objetos en memoria.

Integrar todo lo pedido anteriormente en un menú.

5 XML

Añadir al proyecto la funcionalidad de:

1. A partir de tener el listado de empleados en un Array, generar un fichero `Empleados.xml` con el contenido en formato XML

```
<Empleados>
    <empleado>
        <id>1</id>
        <apellido>FERNANDEZ</apellido>
        <dep>10</dep>
        <salario>1000.45</salario>
    </empleado>
    <empleado>
        <id>2</id>
        <apellido>GIL</apellido>
        <dep>20</dep>
        <salario>2400.6</salario>
    </empleado>
    ...
</Empleados>
```

2. A partir del fichero XML obtenido en el apartado anterior, cargarlo en un array de `Empleado`.

6 JSON

1. A partir de tener el listado de empleados en un Array, generar un fichero `Empleados.json` con el contenido en formato JSON:

```
{"empleados": [
    {
        "apellido": "FERNANDEZ",
        "salario": 1000.45,
        "id": 1,
        "dept": 10
    },
    {
        "apellido": "GIL",
        "salario": 2400.6,
        "id": 2,
        "dept": 20
    },
 ...
]}
```

2. A partir del fichero JSON obtenido en el apartado anterior, cargarlo en un array de `Empleado`.