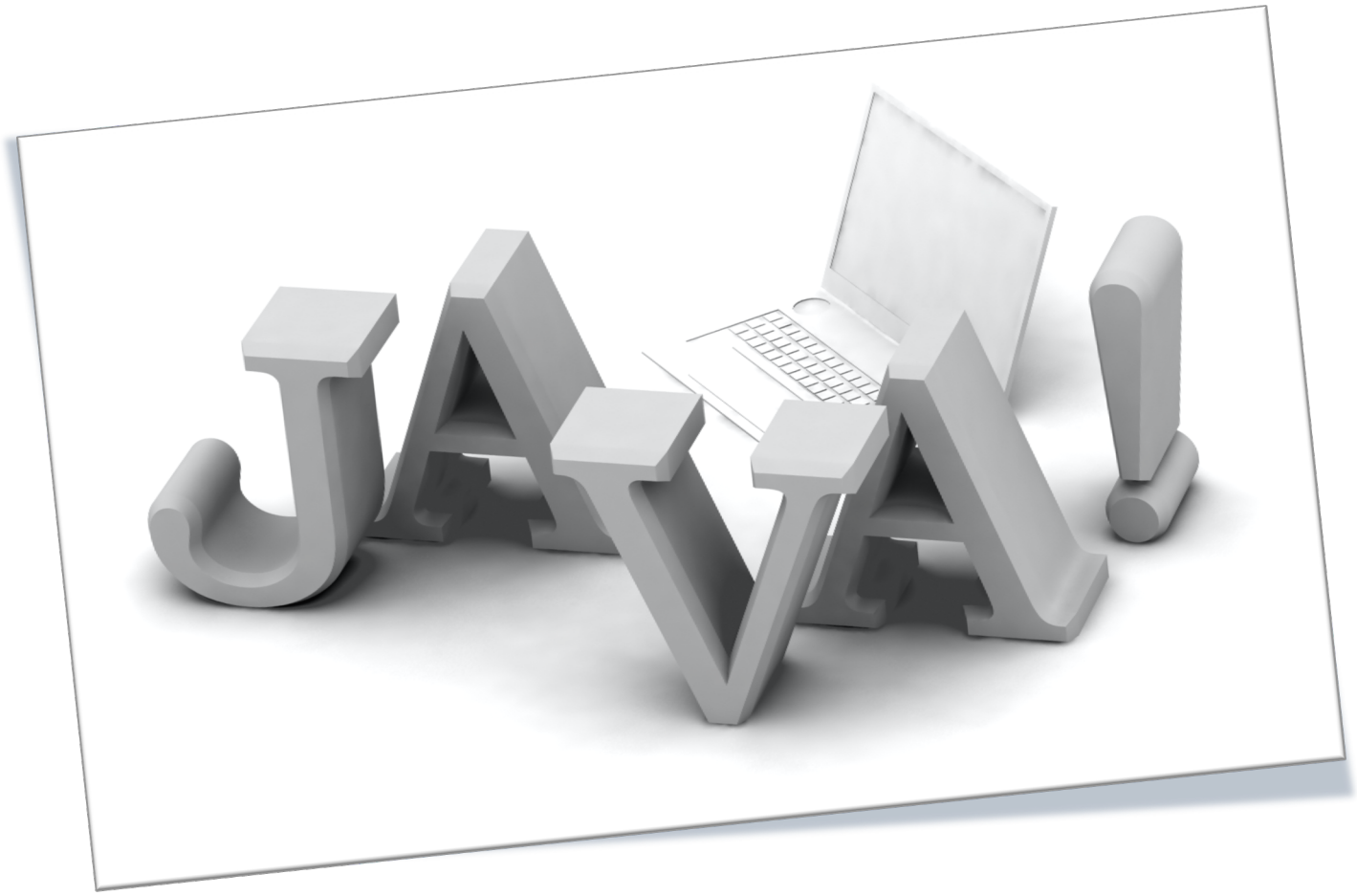
PRÁCTICAS

Capítulo 6: Programación gráfica y ficheros



Unidad 6. Práctica 6.2 Ficheros

Crea un proyecto que se llame Capitulo6
El proyecto debe estar en el paquete org.iesjaroso.daw.capitulo6.
Organiza los ejercicios en paquetes si lo consideras necesario.

Debemos considerar en realizar las prácticas en diferentes proyectos

Lectura y escritura de ficheros en modo texto y en modo binario. Ficheros .csv

Ejercicio 1.- Se desea crear una aplicación que permita generar datos de Personas de forma aleatoria. Para ello disponemos de varios ficheros:

- listaapellidos.txt
- listadnombresmasculinos.txt
- listadonombresfemeninos.txt

Crea una clase llamada Contacto con los atributos siguientes
enum Sexo{MUJER, HOMBRE, NOSELECCIONADO}
enum Grupo{FAMILIARES, AMIGOS, TRABAJO, EMERGENCIA, OTROS}
Atributos:

- String nombre
- String apellido1
- String apellido2
- String telefono
- Grupo grupo
- LocalDate fechaNacimiento
- Sexo sexo
- double latitud
- double longitud
- boolean destacado
- int nivel //Nivel de importancia del 1-10. 1 poco importante, 10 muy importante
- public static List<Contacto> contactos

Constructor:

- **Parametrizados.** Todos los parámetros.
- **Refactoriza el constructor con el patrón Builder.** En el constructor del Builder solo son importantes nombre, apellido1, apellido2 y telefono. Por defecto

Métodos:

- Crea los métodos setters y setters
- toString() Todos los campos
- equals. Dos objetos son iguales si apellido1, apellido2 y nombre son iguales
- Implementa la interfaz Comparable y Serializable

Más Métodos:

- public boolean insertar(Contacto contacto). Se almacena en contactos.
- public Optional<Contacto> buscar(int index)
- public void listar(int desde, int hasta): Solo se muestra nombre, apellido1, apellido2, telefono
- public void listaDetalle(int desde, int hasta). Muestra todos los campos
- public Contacto generarContacto(). Se leerá el fichero listaapellidos.txt, listadonombresfemeninos.txt y listadnombresmasculinos.txt. Para el resto de campos crea métodos que permitan generar datos de forma aleatoria pero cono sentido.
- public void generarDatos(int cantidad). Crea un número total de contactos indicado por cantidad, utilizando el método anterior, y lo almacena en contactos.
- public void guardarTexto(String fichero). Guarda todos los contactos en un fichero en modo texto. Lo más fácil es crear un .csv, este tipo de fichero almacena un registro por línea y cada campo es separado usando ';' La primera línea suele traer el nombre de los ficheros.
- public void cargarTexto(String fichero). Ídem al anterior pero lee los datos.
- public void guardarBinario(String fichero). Almacena los datos de contactos en binario.
- public void cargarBinario(String fichero). Carga los datos de un fichero codificado en binario

- public void guardarObjeto(String fichero). Guarda objetos de datos.
- public void cargarObjeto(String fichero). Carga objetos de datos.
- public void exportarHTML(String fichero). Exporta a html. Los datos puedes ponerlos en una tabla HTML
- public void ordenar()

Crea las excepciones que te parezcan adecuadas para dar robustez a la aplicación.

Ejercicio 2.- Escribe una clase llamada Sudoku.

Atributo:
int[][] sudoku

Constructor:

- Sudoku(int[][] sudoku)

Métodos:
Entre los métodos debe contar con:

- public void mostrarSudoku(int pos)
- public boolean comprobarSudoku()
- private boolean verificarFilas()
- private boolean verificarColumnas()
- private boolean verificarSectores()

Después escribe una clase llamada VerificadorSudoku. La clase debe ser capaz de leer datos del fichero del sudoku.txt y determinar los sudokus válidos.

Atributo:
List<Sudoku> sudokus

Constructor:

- Sudoku()

Métodos:
Entre los métodos debe contar con:

- public void leer(String fichero)
- public void escribir(String fichero)
- public void mostrarSudoku(int pos)

Ejercicio 3.-**Fracciones**. Tenemos un fichero de texto con la siguiente estructura interna:
2/3-25/3+1123/44+3/6-11/27+25/4...sigue
Fíjate que los elementos de tipo fracción están separados por los signos +,-,*,:
Se pide:

- Carga los datos y realiza las operaciones

Nota

- Crea una clase llamada Fracciones, que contenga un array de objetos Fraccion y añade los métodos que creas necesarios para realizar el ejercicio

Ejercicio 4.-**Diccionario definiciones**. Escribe
Crea una clase llamada Diccionario formada por:

Atributos:

- - Map<String, String> diccionario // Está formado por el par termino, definición

Métodos:

- public Diccionario(). Crea el mapa
- public String get(String termino): Obtiene la definición de una palabra.
- public boolean put(String termino, String definicion)
- public void importar(String nombreFichero): El fichero diccionario.txt contiene una lista de términos o palabras separadas por saltos de línea con su correspondiente definición. Cada

termino está separado de su definición por el símbolo "->" Lee el fichero para que se almacene en el mapa diccionario.

- public void guardar(String nombreFichero). Guarda en binario.
- public void abrir(String nombreFichero). Abre un fichero binario.
- public Optional<List<String>> buscar(Predicate<String> criterio). Este método nos permite buscar lista de palabras por un determinado criterio.
  - ✓ Úsalo para buscar términos con los críterios:
    Palabra más larga.
    Palabra más corta.
    Palabra de longitud x.
    10 palabras más largas.
    Palabra que contenga "subcadena"
    Palabra que comience por
    Palabra que termine por
    Palabra que coincida con una expresión regular
    3 primeras palabras que contengan todas las vocales
    10 últimas palabras de la letra "V"

  Se recomienda usar un stream para implementar esté método:

```
return diccionario.keySet()
                  .stream()
                  .filter(criterio)
                  .findAny()
                  …
```

- public void buscarDefiniciones(Predicate<String> criterio). Imprime tanto la palabra como la definición. Implementar de forma parecida al anterior pero teniendo en cuenta que se imprime por pantalla.
- public boolean contiene(String palabra). Nos muestra si existe la palabra
- public void reversa(String palabra). Busca una palabra y obtiene su definición. Después coge todas las palabras que tiene la definición y la convierte en una lista de palabras únicas. Por último busca cada una de estas palabras en el diccionario y muestra ambas. Termino y definición. Si no se encuentra escribe el termino y en la definición "No existe definición"
- public int getNumeroTerminos(): devuelve el número de palabras del diccionario
- public void buscarInversa(String palabra). Muestra todos los términos en donde *palabra* esté presente en la definición
- public void buscarInversa(Predicate<String> criterio). Ídem al anterior pero buscamos un criterio, en este caso en lugar de los términos, en las definiciones.

Gestión de carpetas y ficheros. Biblioteca NIO 2. Clases File, Path.

Ejercicio 5. Carpetas y ficheros.

Crear una clase UtilFichero con los siguientes métodos estáticos:

a) public static void crearRuta(Path ruta): un método que cree en todos los directorios de una ruta, usando el nombre de un fichero pasado como parámetro. Por ejemplo: crearRuta("peliculas/aventuras/cortos/peli01.avi")
b) lic static crearRutas(List<Path> rutas). Idém al anterior pero con todas las rutas. Creará las carpetas si no existen películas, aventuras, cortos y el fichero peli01.avi
c) public static int size(Path ruta). Calcula el espacio que ocupa todos los ficheros que hay en una ruta. Deberá recorrer recursivamente todos las subcarpetas de esa ruta.
d) public static borrar(Path ruta, Predicate<String> criterio). Borra todos los archivos recursivamente que cumpla una condición. Por ejemplo se le pasa una ruta "películas" y una condición "a*.avi"
e) public String info(Path fichero):
   <nombre> <Fichero|Carpeta> <FechaCreacion> <EspacioOcupado>
f) public static void listar(Path ruta). Muestar la info de forma recursiva de una ruta
g) public static void generar(Path ruta, int maxDir, int maxFicheros). Crea automáticamente en una ruta dada un número de directorios comprendido entre [1, maxDir] con nombres aleatorios y dentro de cada directorio crear un número de ficheros entre [0, maxFicheros] con nombres aleatorios.
h) public static Path getPathAzar(Path). Obtiene de un sistema de ficheros una ruta completa a un fichero o carpeta de forma aleatoria dado una ruta. Se debe hacer de forma recursiva.