EJERCICIOS PROGRAMACION III
Práctica SEMANA 8

CONTENIDOS: Introducción MVC y Persistencia: Ficheros en Java I - texto delimitado.

Ejercicios obligatorios:

Revisad la documentación de biblioteca.jar sobre todo las clases OpMat (para operaciones con matrices y mostrar tablas por pantalla), Rutas (acceso a ficheros) y Esdia (entrada y salida por consola robusta). Utilizad siempre rutas compatibles con todos los sistemas operativos con los mecanismos que ofrece Java.

1. **Ejercicio de la Quiniela**. Se deberá implementar el ejercicio ya tratado anteriormente de la quiniela, pero en esta ocasión se deberá aplicar el patrón MVC simple. Implementa la solución empleando colecciones y clases como podrían ser Quiniela, Partido y Equipo. Realiza las siguientes versiones de este ejercicio.

   v1:MVC simple: con opciones para solicitar los partidos, resultados y mostrar la quiniela.

   v2:MVC simple similar al anterior pero **con carga desde fichero .txt**: en la inicialización del programa los datos fijos de la jornada (partidos entre equipos) se encontrarán en líneas de un fichero de texto llamado `equipos.txt` situado en una carpeta llamada datos en el escritorio del usuario. En esta versión del programa los partidos se crean a partir de cada una de las líneas del fichero de texto que son de la siguiente forma:

   ```
   Zamora CF-CD Rioseco
   ```

   El programa deberá seguir un estilo arquitectónico MVC simple (sin herencia ni interfaces) con clases con las responsabilidades correspondientes cuando se sigue esta arquitectura. Es posible utilizar Arrays o Colecciones y métodos de utilidad incluidos Biblioteca.jar, en concreto las clases Rutas.java y OpMat.java.
2. **Una compañía tiene un archivo de Facturas**, con formato de texto delimitado por tabulaciones (TSV). Cada línea del archivo de texto delimitado contiene en este orden:

   - Concepto de la factura.
   - Descuento aplicado que puede ser 0 y se expresa como un número menor que 1 y con 2 decimales, (e.g. 0.05).
   - Fecha como cadena.
   - Importe de la factura.
   - NIF.
   - Nombre del cliente.
   - Dirección.
   - IVA (e.g: 0.21)

Es necesario construir un programa que proporcione las siguientes opciones:

- **Opción que permita proporcionar un valor mínimo de factura**, que se calculará de la forma: importe*(1-descuento)*(1+IVA) para cada factura, y que proporcione el listado de clientes que tengan facturas con importe mayor que el proporcionado por consola.
- **Opción que permita mostrar por pantalla el listado total de facturas**.
- **Opción que permite exportarlas en un archivo .html simple**. Se deberá crear el texto del HTML y la tabla en lenguaje HTML generada dinámicamente en función de las facturas disponibles en el lugar correcto. Utiliza funciones como String.replace y marcadores dentro del HTML (e.g. %%TABLA%%) para que te sea más sencillo introducir la tabla en String del HTML. Abre el fichero con el navegador para comprobar que funciona correctamente.
- **Opción que permita exportar las facturas a un archivo .csv (delimitado por comas)**.

Resolver el problema empleando colecciones, clases e incluso bibliotecas (como biblioteca.jar). Además, se deberá emplear la arquitectura MVC ya conocida.

3. Crear un programa basado en la clase Persona (nombre, teléfono y peso), colecciones y MVC. El programa deberá poder realizar las siguientes acciones:
   - **Cargar en el arranque de la aplicación** los datos que residen en un archivo en el escritorio llamado datos.txt si estuviera disponible e informar por consola al usuario (tanto si se han cargado como si no). El archivo datos.txt tiene formato delimitado por tabuladores.
   - **Opción de menú que permita leer de un fichero TSV (importar) los datos**. Se le proporcionará el nombre de fichero por consola y este fichero debe residir en el escritorio.
   - **Opción que permita mostrarlos en forma tabular**. Implementa métodos en la clase Persona que faciliten este proceso para representar el estado del objeto como una fila y para obtener una cabecera.
   - **Opción salir del programa** tras confirmar con el usuario si está seguro de que quiere salir.
4. **Programa ListaDeClase**. Está basado en la clase DatosDeAlumno, que tiene un atributo de tipo Dirección y otro de tipo DatosPersonales. La clase Dirección tiene los siguientes atributos: calle (String), número (int), piso (int), letra (String). La clase DatosPersonales tiene como atributos: nombre (String), apellidos (String), edad (int), NIF (atributo alfanumérico y **no repetido**). Tanto la clase DatosPersonales como Dirección poseen un factory method que puede crear ejemplares de la clase con valores aleatorios razonables para los atributos. Revisa la clase **`Random`** en Java y cómo obtener un entero entre dos valores enteros. El programa creará en el inicio una colección de 20 alumnos con valores razonables aleatorios y mostrará un menú con las siguientes opciones:

   1 Volver a crear la lista de alumnos
   2 Mostrar tabla de alumnos
   3 Exportar datos personales (CSV)
   4 Exportar direcciones (CSV)
   q Salir

   La tabla mostrada en la opción 2 debe contener tanto los datos personales como los de la dirección.