PRÁCTICA 3:

Pablo Martínez Ubide y Edurne Abian Andres

Portal de Datos Públicos del Ayuntamiento de Zaragoza. Trabajando con ficheros .csv

OBJETIVOS:

En esta práctica se nos pide representar la información almacenada en los ficheros ".csv" y programar una colección de clases y funciones para facilitarnos el tratamiento de estos datos; finalmente, debemos diseñar un conjunto de programas que hagan uso de los ficheros de texto y creando otros nuevos.

TAREA 1:

Main: Creamos el ArrayList para almacenar los datos leídos. Pasamos los ficheros que se facilitaron para la práctica. Se llama a un método para leer el fichero y almacenarlo.

Método lineas: Recibe la ruta de un fichero, saca la información separada por puntos y comas y la almacena en un ArrayList de <UsoBizi> con las líneas para leer que se le manda como parámetro. Consta de *nombreFichero*, que recoge el nombre del fichero que lee, y *nombreCopia*, que es el nombre donde se va a guardar la información de la nueva copia reducida. *Totales*, que almacena la información, y *max*, que es el número de líneas que se van a leer del fichero original.

Se crea un scanner para leer el fichero ".csv", se hace un Formatter para guardar el archivo. Se crea un contador para no tener en cuenta la primera línea y comienza un bucle mientras siga habiendo líneas. En él se crea un string donde se almacena la información de una fila, y se realiza un Split para ir separando por punto y comas la información, y quedarnos con las columnas primera, tercera y quinta. Se añaden como objetos de la clase <UsoBizi> en cada iteracción y se va añadiendo a un ArrayList y finalmente se guarda en el disco otra vez como ".csv".

TAREA 2:

LEER FICHERO: Se crea un scanner para leer el fichero que se creó anteriormente, se van almacenando las filas y separándolas con un Split por ";", y se crea un objeto con la información de la fila, que a su vez se añade a un ArrayList.

TRASLADOS: Método que muestra por pantalla la información que se requiere en la práctica, por lo que se recibe el ArrayList que se ha creado en el método anterior al leer el archivo.

Se crea una variable para ir aumentando el número de recorridos circulares que hay en total, se recibe el ArrayList para registrar los traslados y si la estación de anclaje y recogida es la misma, el recorrido es circular; en caso contrario es traslado. Y se muestra por pantalla cada uno de los usos. Traslados, circulares y totales (estos corresponden al tamaño del Arraylist).

TAREA 3:

Se crea el scanner para recoger la ruta del fichero, creamos el ArrayList donde se almacena los datos leídos, pasamos el fichero elegido por el usuario, teniendo en cuenta que deberá seguir el modelo de los datos de la primera tarea.

Se llama al método para leer el fichero y almacenarlo. Se crea un ArrayList de objetos tipo <UsoBizi> para almacenar la información, y a su vez un ArrayList del objeto de <UsuariosBizi> en el que se irán almacenando el número de usos que ha hecho un usuario, después de analizar todo el archivo con los usos. Para ello se crea un bucle que recorre cada uso y llamando al método que recoge si es circular o no se irán sumando los usos para ese usuario, cuando coincida el ID de la fila de uso leída.

COMPROBAR ID: Recibe un id de usuario de un uso y comprueba si ya existe en el ArrayList que contiene los usuarios. En caso de que exista se devuelve el índice; si no, devuelve -1.

BUSCAR CIRCULAR: Compara fila por fila los id de las estaciones de recogida y retirada para devolver en cada caso 0 o 1.

USO BIZI: Compuesto por:

- *idUsuario*: Id del usuario que hace uso del sistema.
- *estRetiro*: Id de la estación de retirada de la bici.
- *estAnclaje*: Id de la estación de dejada de la bici.

También los constructores de la clase. El método toString para devolver la info sobre un uso. Los getter and setters

USUARIO BIZI: Compuesto por:

- *idUsuario*: Id del usuario que hace uso del sistema.
- *usosTraslados*: La cantidad de usos de traslados.
- *usosCirculares*: La cantidad de usos circulares.
- Método que suma un uso traslado.
- Método que suma un uso circular.
- Método toString para mostrar la información de un usuario.
- Getter y setter correspondientes.
- Método compareTo, heredado de la interfaz “comparable”.