Práctica 10: Archivos Eda 2

Oscar Daniel Rosario Morales, Mejía Alba Israel Hipólito

6 de noviembre de 2018

1. Objetivo

El estudiante conocerá e identificará aspectos sobre los archivos, como las operaciones, el tipo de acceso y organización lógica.

2. Desarrollo

2.1. Bibliotecas en Java

Las bibliotecas usadas para el manejo de archivos fueron la .io y .nio , a continuación se dará una breve descripción de cada una así como de las clases usadas:

2.1.1. Java.io:

Proporciona las clases necesarias para la creación de objetos que tengan los medios necesarios para realizar operaciones de entrada y salida del sistema mediante el flujo de datos, serialización y el sistema de archivos.

import java.io.BufferedReader :

es una clase de Java para leer el texto de una secuencia de entrada (como un archivo) almacenando en el búfer caracteres que leen a la perfección caracteres, matrices o líneas.

import java.io.File:

La clase File se usa para obtener información sobre archivos y directorios. Además la clase File permite crear y eliminar archivos y directorios. Un objeto de la clase Java File representa un archivo o directorio. Durante la práctica fue de utilidad para crear objetos de tipo File con el mismo directorio que el archivo idealmente creado que nos permitan hacer uso del método existe y delete, ambos para los métodos creados para eliminar el archivo creado anteriormente

import java.io.FileReader:

Esta clase es usada para leer archivos de caracteres. Su método read() es usado a bajo nivel, permitiendo leer caracteres de manera singular, todo el stream de caracteres, o un número fijado de caracteres. Los FileReaders están usualmente envueltos en objetos de alto nivel como BufferedReaders, el cual incrementan el rendimiento y proveen una manera conveniente de trabajar con información.

import java.io.FileWriter:

Esta clase es usada para escribir caracteres en archivos. Su método write() permite escribir caracteres o strings a un fichero. Esta clase normalmente está envuelta en objetos Writer de mas alto nivel como BufferedWriter o PrintWriters, lo cual proveen un mayor rendimiento y un alto nivel, con métodos mas flexibles para escribir información. Se creo un objeto de Este tipo el cual necesitaba la ruta del archivo en el que iba a escribir con ayuda del otro objeto creado de tipo PrintWriter. import java.io.IOException;

import java.io.IOException:

Significa que se ha producido un error en la entrada/salida. Por ejemplo, cuando estamos leyendo de la consola, un fichero, etc. Es obligatorio tratar la excepción, ya sea en la cabeza del método con ”throws IOException.º con un bloque try/catch.

import java.io.PrintWriter:

Clase que nos permite escribir con formato texto tanto en la salida estándar, como en ficheros, en cadenas, o en streams.

Entre los métodos que contiene, se utilizaron a append y close, el primero recibe la cadena que será escrita en el archivo definido con el FileWrier y el segundo para cerrar el flujo de datos.

Hay que especificar, a través de los argumentos, dónde se va a producir la salida.

EJ. 1: PrintWriter pw1= new PrintWriter(System.out);

EJ. 2: PrintWriter pw2= new PrintWriter(new File(”fichero.txt”));

A partir de la versión 5, incluso se permite:

Ej. 3: PrintWriter pw2= new PrintWriter(”fichero.txt”);

Implementa todos los métodos de la clase PrintStream. A diferencia de esta clase, sus métodos de escritura no se vacían automáticamente al escribir una nueva línea.

2.1.2. Java.nio:

Java NIO (Not Blocking IO) es un nuevo API disponible desde Java7 que nos permite mejorar el rendimiento así como simplificar el manejo de muchas cosas. Define los buffers, que son contenedores para datos, y proporciona una visión general de los otros paquetes NIO.

import java.nio.file.Files:

Esta clase consiste exclusivamente en métodos estáticos que operan en archivos, directorios u otros tipos de archivos. Esta clase contiene a los métodos createDirectories y createFile, que como su nombre lo dice nos sirvieron para que con ayuda del path, crear tanto el directorio como el archivo deseado.

import java.nio.file.Paths:

Esta clase consta exclusivamente de métodos estáticos que devuelven una ruta convirtiendo una cadena de ruta o URI. Se usó en la práctica para usar una cadena como directorio, el cual nos ayudó a crear tanto la carpeta como el archivo deseado.

3. Conclusiones

Rosario Morales Oscar Daniel

Si bien algunos de estos métodos e implementaciones ya los habíamos utilizado con anterioridad en el proyecto 1. No se les dio la importancia verdadera, ya que surgieron mas problemas en la implementacion de los algoritmos de ordenamiento. Sin embargo en la resolución de esta practica 10, se pudo entender mejor el funcionamiento de las diferentes clases así como de los métodos que las conforman, para el correcto funcionamiento de archivos en el lenguaje de Java, tanto escritura como lectura dentro de los ficheros así como la creación de los mismos.

Mejía Alba Israel Hipólito

Esta práctica fue de gran provecho ya que aprendí como hacer uso de ficheros en java, incluso mejor que en semestres anteriores en lenguaje C y python. Comprendí cómo crear ficheros, agregar cadenas al mismo y eliminar el fichero por completo.Por otro lado tuve problemas con el buffer al mandar hacer distintos métodos creados en la clase Practica10. Por último me pareció interesante la clase IOExceptions para hacer el manejo de excepciones.

4. Bibliografia y Referencias

*https://docs.oracle.com/javase/7/docs/api/
* https://sekthdroid.wordpress.com/tag/java-filereader/
*http://zaguandelainformatica.blogspot.com/2014/11/la-clase-printwriter-de-java.html