TP2 : File Navigation and I/O

A summary of the I/O main classes is as follows:

File : creates a java object (File MetaData-descriptor serving as a java driver which negotiates with the corresponding filesystem (AIX/jfs, Solaris/ufs, Windows/NTFS, etc.).

Several methods can be invoked like :

```
delete()

renameTo()

exists()
```

FileReader This class is used to inform the filesystem that read operations would be requested. A no permission exception could be returned. Otherwise, the filesystem and JVM will prepare the suitable mechanism to serve future read access methods.

BufferedReader This class is used to make FileReader more simple and efficient to use. Designer will find in BufferedReader better methods in term of flexibility and performance.

FileWriter This class is used to inform the filesystem that write operations would be requested. It is used to write to character files. Its write() methods allow you to write character(s) or Strings to the target file.

BufferedWriter This class is similar to FileWriters and used to make some kind of write operations more efficient and easier to use.

PrintWriter This class has been created to enhance mechanism and related methods of writing to a file. New methods like format(), printf(), and append() make PrintWriters very flexible and powerful.

a. Exercise 1

- Write a java application called “CatFile.java” which prints on standard output a file (The file name is passed as argument).
- Rewrite “CatFile.java” into an interactive version called “CatFileFd.java” which invokes a FileDialogue object if no argument is passed.

In the two versions all exception catches should be controlled and taken. For each case, give a solution with FileInputStream, an other with FileReader, and a third version with BufferedReader.

b. Exercise 2

- Write a java application called “WriteFile.java” which permits the construction and the writing of a new file (The file name is passed as argument). Data are written using keyboard. You can use a Scanner(System.in) instance to handle with keyboard input.

All exceptions should be catched. Give a solution with FileOutputStream, an other with FileWriter, a third version with BufferedWriter and a fourth version with PrintWriter.

- Rewrite “WriteFile.java” into an interactive version called “WriteFileFd.java” which invokes a FileDialogue object if no argument is passed.
- Write a java application called “AppendFile.java” which permits to add new data to an existing file (The file name is passed as argument).
- Rewrite “AppendFile.java” into an interactive version called “AppendFileFd.java” which invokes a FileDialogue object if no argument is passed.
- Rewrite all the previous Java applications into new versions using Frames and TextAreas. The frame should be splitted into two Panels. The top panel (Refer to Panel class in javadoc) contains the file proprieties and the bottom panel contains the TextArea. This latter (TextArea) will contains the data of the file. In other hand, the Menubar should be used to interact with the system (for example an open MenuItem will invokes the FileDialogue to select the desired file. I Addition, a “Format” Menu attached to The Menubar with help the user to customize displayed data in TextArea space (for example a Size MenuItem will help to adjust police size “12” or “14”).

In the versions all exception catches should be controlled and taken.