Streams Cheat Sheet for Grace

The following instructions show how programmers can read and write text files in Grace.

1 Opening a file for reading.

Files to be read by Grace programs must first be loaded into the browser in the same way that programs are loaded, i.e. by clicking on the upload icon at the top of the left pane that holds the list of files and then selecting the file in the dialog box.

File operations are accessed from built-in library `io`. Thus a program using files will need to import the library, e.g.

```
import "io" as inout
```

A file that has been opened in a program has type `FileStream`, a type defined in `io`. We can access a file for reading by invoking method `open` of `io` with two parameters, the complete path to the file, and the string "r". Thus if file `words.txt` is in the folder `MyFolder`, then we open the file and give it name `input` as follows:

```
def input: inout.FileStream = inout.open("MyFolder/words.txt","r")
```

The three most important methods available on an object of type `FileStream` that has been opened for reading are `getline`, `eof`, and `close`. In the example above, `words.getline` would return the next line of the file (i.e., a string consisting of all the characters up to but not including the new line character). The expression `words.eof` would evaluate to `true` if and only if the there is nothing left unread in the file. When you are done reading a file, you should always close it so that if you wish to read and write to it later, you can open it.

If you would like to get the entire file as a single string, the `read` method will provide that. However, in most cases it will be easier to read a file a line at a time.

2 Opening a file for writing

We can write to existing files or create new ones to write to. In either case, when a file is opened for writing it is first erased. New data cannot be added to the end of an existing file.[1]

Files are opened for writing similarly to reading except that the second parameter is "w" rather than "r".

```
def output: inout.FileStream = inout.open("MyFolder/vals.txt","w")
```

[1] It is possible to open a file so that you can write new material at the end of the existing material. See the Grace documentation at
`http://web.cecs.pdx.edu/~grace/doc/modules/io/` for further information.

Data can be written to the file using method `write`. Unlike the `print` command, `write` does not move to the next line after the string is written. Thus if you want output to be on separate lines, you will need to tack on an `"\n"` at the end of the text to be output.

It is critical that you `close` a file after you have finished writing to it. If you don't, the data to be written may remain in a buffer and never end up written to the file.

3 Exceptions

Errors can occur when trying to access files. These can include a file that is not there (or in use by another program), trying to read from a file that has disappeared, trying to write to a file when no more space is available, etc. These errors will throw an `IOException`, so all code accessing files should surround the file commands with a try catch construct. See the example below

4 Example

In this section we show an example of a program that reads in all the lines of the file `words.txt` in folder `MyFolder` and copies them into file `copy.txt` in the same folder. After the program is complete, the two files should be identical.

```
try {
   def words: inout.FileStream = inout.open ("MyFolder/words.txt","r")
   def output: inout.FileStream = inout.open("MyFolder/copy.txt","w")
   while {!words.eof} do {
       def nextWord: String = words.getline
       output.write(nextWord ++ "\n")
   }
   words.close
   output.close
} catch {ex: IOException ->
    print "Exception {ex}"
}
```