Learning goals

Before the next day, you should have achieved the following learning goals:

- Read from text files
- Write to text files

1 ls

Write a program that shows on screen the names of the files in the current directory. (Hint: look at methods from class File).

2 mkdir

Write a program that takes a name from the user at the command line and creates a directory with that name. Remember that the only argument of method `main` is an array of Strings, each of them an argument written after the name of the class. For example, if you write `java myClass this That 0`, the elements in `args` will be three strings of values "this", "That", and "0".

3 cat

a)

Write a program that takes a name from the user at the command line. If a file with that name exists, the program must show its contents on screen. If it does not, the program must say so.

b) (*)

Modify the program so that it takes many file names at the command line, and then shows them all one after the other.

4 cp

a)

Write a program that takes two names from the user at the command line. If a file with the first name exists, the program must copy it into a file with the second name.

If the first file does not exist, the program must say so. If the second file does exists, the program must ask the user whether to overwrite it or not, and act accordingly.

b) (*)

Modify the program so that it takes many file names at the command line. When this happens, the last name must be a directory (otherwise, your program should complain). If it is a directory, your program has to copy all files (i.e. the other arguments) into that directory.

5 tr (*)

Write a program that takes a name and two strings from the user at the command line. If a file with that name exists, the program must show its contents on screen, but substituting any occurrence of the first string by the second string. If the file does not exist, the program must say so.

6 sort (*)

Write a program that takes a name from the user at the command line. If a file with that name exists, the program must show its contents on screen, but with the lines shown alphabetically. If the does not exist, the program must say so.

Hint: Remember that Strings in Java implement the interface `Comparable<String>`.

7 uniq (*)

Write a program that takes a name from the user at the command line. If a file with that name exists, the program must show its contents on screen, but removing duplicates lines (showing on screen only one line for each set of duplicated lines). If the does not exist, the program must say so.

8 Temperature averages

Write a program that reads a file from disk in comma-separated format (CSV). Every line must contain a list of number separated by commas.

The program must output the average for every line plus the average for the whole file. A file may look like the following:

```
25, 24, 20, 18, 15, 13, 14, 13, 15, 17, 19, 21
25, 25, 24, 20, 18, 15, 13, 14, 13, 15, 17, 19
21, 25, 25, 24, 20, 18, 15, 13, 14, 17, 19, 21
25, 25, 24, 20, 18, 15, 13, 14, 13, 15, 17, 19
21, 25, 25, 24, 20, 18, 15, 13, 14, 13, 15, 17
21, 25, 25, 24, 20, 18, 15, 13, 14, 13, 15, 17
19, 21, 25, 25, 24, 20, 18, 15, 13, 14, 13, 15
17, 19, 21, 25, 25, 24, 20, 18, 15, 13, 14, 13
...
```

9 Binary cp (**)

Write a program that takes two names from the user at the command line. If a file with the first name exists, the program must copy it into a file with the second name. If the first file does not exist, the program must say so. If the second file does exists, the program must ask the user whether to overwrite it or not, and act accordingly.

This is the same exercise as above with an important difference: it must be able to copy binary files (use `InputStream` instead of `Reader`, etc). Try it with `.class` and `.exe` files and check that the copies work exactly like the originals.