SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

Computer System Engineering

Week 6: Lab 4 (25 marks)

Objective: Implement file operations in Shell Interface using Java/C

Java Version:

In Lab 1 we have implemented a Shell Interface. In Lab 4 we will extend the Shell implementation with several file operation methods.

A collection of file operation functions are defined in **java.io.File** class. You can refer to the documentation of **java.io.File** class from the following link:

http://docs.oracle.com/javase/7/docs/api/java/io/File.html

In order to implement Q1 to Q4 (see below), you need to write code to handle different file operation commands, and also implement the following functions:

```
public static void Java_create(File dir, String name);

public static void Java_delete(File dir, String name);

public static void Java_cat(File dir, String name);

public static void Java_ls(File dir, String display_method, String
sort_method);

public static boolean Java_find(File dir, String name);

public static void Java_tree(File dir, int depth, String sort_method);
```

The first 3 functions are for Q1, and the last 3 functions are for Q2, Q3, Q4, respectively.

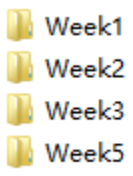
Q1. Implement functions to create, delete, and display a file (10 marks)

Use the starting code “FileOperation – starting code.java” to implement your solution.

For Q1, there are 3 operations for a file: *create*, *delete*, and *display*.

Create a file: New files can be created and added to current directory.

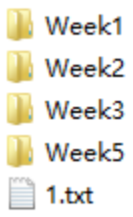
For example, let us say that we have 4 sub-directories in the current directory:



From our Shell Interface, when we type in the following command:

```
jsh>create 1.txt
```

A file called "1.txt" will be created in the current directory:

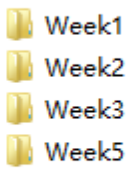


Delete a file: When a file is no longer needed, we want to remove it from the current directory.

For example, when we type in the following command:

```
jsh>delete 1.txt
```

The file "1.txt" will be deleted from the current directory:



Display a file: When we want to see the contents of a file, we use the display command by typing in the following command:

```
jsh>display test.txt
```

The contents of "test.txt" will be displayed on the screen:

```
Hello.
This is the content of the file test.txt.
```

Suggested Function:

Function	Class which function belongs to
.exists()	*File*
.createNewFile()	*File*
.readLine()	*BufferedReader*

Test Command:

jsh> create a.txt
jsh> display a.txt
jsh> delete a.txt

Q2. Implement function to list a directory (5 marks)

Continue to use the code you implemented in Q1.

Here we design a function to list files under the current directory.

When we type:

```
jsh>list
```

The files (including directories) will be listed as follows:

Week1
Week2
Week3
Week5

We also add the option to show properties of files, e.g. file size, last modified time.

When we type:

```
jsh>list property
```

The files (including directories) will be printed out as follows:

```
Week1        Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week2        Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week3        Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5        Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The list function should be able to sort the files according to different properties.

For example, when we type:

```
jsh>list property time
```

The files (including directories) will be printed out as follows:

```
Week2        Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week1        Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
```

```
Week3        Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5        Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The function for sorting a list of files and directories based on different properties (e.g. name, size, time) is provided in the starting code:

```
private static File[] sortFileList(File[] list, String sort_method);
```

Suggested Function:

Function	Class which function belongs to
.lastModified()	*File*
.getName()	*File*
Date()	*Date*

Test Command:

jsh> list
jsh> list property
jsh> list property time

Q3. Implement function to find files in the current directory and subdirectories (5 marks)

Continue to use the code you implemented in Q2.

Here we design a function to find files in the current directory and its subdirectories. For example, the current directory is **C:\CSE_Lab\src\**. When we type in the find command with an argument as follows:

```
jsh>find .java
```

The function will print out the names of all files with ".java" as a suffix in the filename. The output is as follows:

```
C:\CSE_Lab\src\Week1\SimpleShell.java
C:\CSE_Lab\src\Week2\MergeSortThreaded.java
C:\CSE_Lab\src\Week2\MultiThread.java
C:\CSE_Lab\src\Week3\Bank.java
C:\CSE_Lab\src\Week3\BankImpl.java
C:\CSE_Lab\src\Week3\TestBank.java
C:\CSE_Lab\src\Week5\FileOperation.java
```

Note that you need to find files not only in the current directory, but also in its subdirectories. So here you can implement a recursive function to find files at different levels of subdirectories.

Suggested Function:

Function	Class which function belongs to
.contains()	*String*
.listFiles()	*File*

Test Command:

jsh> find .java
jsh> find .xyz

Q4. Implement function to list subdirectories and files in a tree structure (5 marks)

Continue to use the code you implemented in Q3.

In order to efficiently show files at different levels of sub-directories, we need a function to show files in a tree structure. For example, when we type in the *tree* command under the directory **C:\CSE_Lab\src\**:

```
jsh>tree
```

The output should be as follows:

```
Week1
  |-SimpleShell.java
Week2
  |-data
    |-input_1.txt
    |-input_2.txt
  |-MergeSortThreaded.java
  |-MultiThread.java
Week3
  |-Bank.java
  |-BankImpl.java
  |-TestBank.java
Week5
  |-FileOperation.java
```

Sometimes we only want to see a few of the upper levels of a directory. We should be able to control the maximum level of subdirectories to be shown. For example, when we type in the *tree* command with the argument 1 in the directory **C:\CSE_Lab\src\**:

```
jsh>tree 1
```

The output should be as follows:

```
Week1
Week2
Week3
Week5
```

As you can see here, we set the maximum level of subdirectories to be 1, so only the first level of subdirectories is shown.

Similar to the list command, we can also decide which file property we use to sort the file order to be displayed. For example, when we type in the *tree* command in the directory **C:\CSE_Lab\src\**:

```
jsh>tree 2 time
```

The output should be as follows:

```
Week2
  |-MergeSortThreaded.java
  |-MultiThread.java
  |-data
Week1
  |-SimpleShell.java
Week3
  |-Bank.java
  |-TestBank.java
  |-BankImpl.java
Week5
  |-FileOperation.java
```

The tree structure is listed based on *time* order. So the sequence is different from the one that uses *name* order.

Suggested Function:

Function	Class which function belongs to
.isDirectory()	*File*
.listFiles()	*File*

Test Command:

jsh> tree
jsh> tree 2
jsh> tree 5

The starting codes for Lab 4 can be found on eDimension:

FileOperation - starting code.java

After you finish all 4 questions, submit the java file (modified from "FileOperation – starting code.java") to eDimension before 11:59 pm, 10 March 2017.

File operation functions in ***java.io.File*** *class:*

Create a file:

File file;

file.createNewFile();

Delete a file:

```
File file;
file.delete();
```

List files:

```
File dir;
File[] list = dir.listFiles();
```

Get file property:

```
File file;
file.getName();                        //file name
file.length();                         //file size
new Date(file.lastModified());         //file time
```

Get file path:

```
File file;
file.getAbsolutePath();
```

Check whether a file is a directory(folder):

```
File file;
file.isDirectory();
```

Lab 4: C version

Week 6: Lab 4 (25 marks)

Objective: Implement file operation in Shell Interface using C

Please note that for the C version of the lab, you need to implement the functions to (1) create, delete, display files; (2) list directories; (3) find files in a directory and (4) print the directory structure in a tree format on your own. **You should not make use of system calls to run existing system programs** (e.g. cat, tree) to provide the above functionality.

NOTE: In the C part, if you have any problem in any command, just type "man" followed by the command, the manual page for that command will help you to get more information and options about the command.

For C:

A collection of file operation functions are defined in **C**. You can refer to the documentation of file operations in the following link:

http://www.thegeekstuff.com/2012/07/c-file-handling/

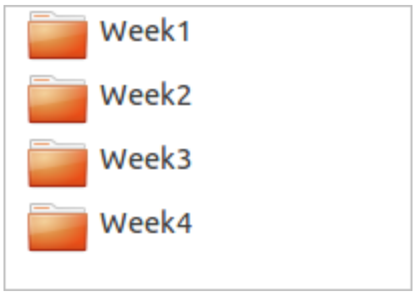
Q1. Implement functions to create, delete, and display a file (10 marks)

Use the starting code "StartingMain.c" to implement.

For Q1, there are 3 operations for single file: create, delete, and display.

Create a file: New files can be created and added to the current directory.

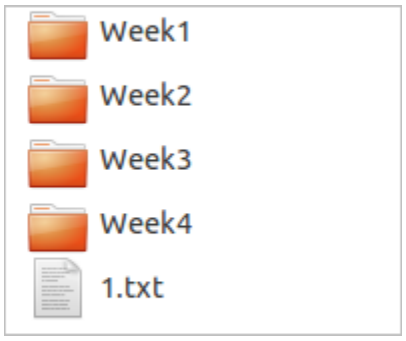
For example, now we have 4 subdirectories under the current directory:



From our Shell Interface, when we type in the following command:

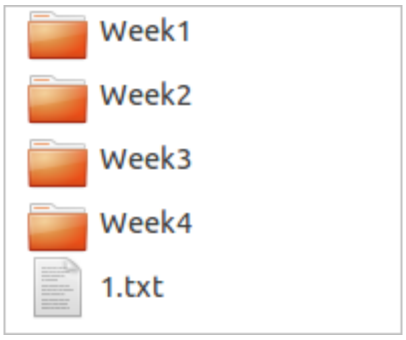
```
csh>create 1.txt
```

A file called “1.txt” will be created under the current directory:



Delete a file: When a file is no longer needed, we want to remove it from the current directory.

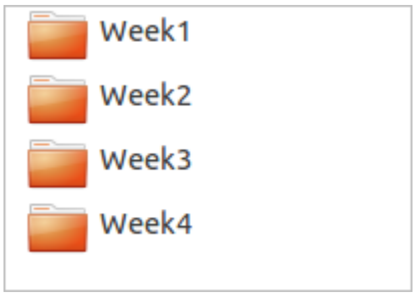
For example, now we have 4 folders and the file “1.txt” under the current directory:



When we type in the following command:

```
csh>delete 1.txt
```

The file “1.txt” will be deleted from the current directory:



Display a file: When we want to see the contents of the file “test.txt”, we use the display command by typing in the following command:

```
csh>display test.txt
```

The contents of the file “test.txt” will be displayed on the screen as follows:

```
Hello.
This is the content of the file test.txt.
```

Suggested Function:

Function	Usage	Needed headers
main()	*the core of every program and it is required in each c program*	`#include <stdio.h>` `#include<stdlib.h>`
fgetc	*To take input from user*	

Function	Usage	Header
*FILE *fp;* *fopen=(file.txt,"w+");*	*Create writable file with (file.txt) as a name*	`#include <stdio.h>`
*int remove(const char *filename);*	*Remove function*	
*int unlink(const char *filename);*	*Unlink function*	

Test Command:

csh> create a.txt
csh> display a.txt
csh> delete a.txt

Q2. Implement function to list a directory (5 marks)

Continue to use the code you implemented in Q1.

Here we design a function to list files under the current directory, it should work as the *ls* command in Linux.

When we type in:

```
csh>list
```

The files (including folders) will be listed as follows:

```
Week1
Week2
Week3
Week5
```

We also add in the option to show properties of files, e.g. file size, last modified time.

When we type in:

```
csh>list property
```

The files (including subdirectories) will be displayed on the screen as follows:

```
Week1        Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week2        Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week3        Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5        Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The list function should be able to sort the files according to different properties.

For example, when we type in:

```
csh>list property time
```

The files (including subdirectories) will be printed out as follows:

```
Week2        Size: 4096        Last Modified: Sun Jan 18 21:09:22 SGT 2015
Week1        Size: 4096        Last Modified: Mon Jan 26 13:10:47 SGT 2015
Week3        Size: 4096        Last Modified: Thu Feb 05 16:43:57 SGT 2015
Week5        Size: 0           Last Modified: Thu Feb 12 16:16:27 SGT 2015
```

The function for sorting a list of files based on different properties (e.g. name, size, time), to determine all the options, you need to use the opendir() and readdir() functions:

Suggested Function:

Command	Usage	Header
strtok	*Parsing the command*	`#include<string.h>`
chdir	*Change directory*	`#include<unistd.h>`
strcmp	*Compare two strings*	
strcpy	*Copy between strings*	
strstr	*Locate a substring in another string*	

*DIR * opendir(const char * dirname);*
 *struct dirent *readdir(DIR *drip);*
	Open the directory and read its files.	*#include<dirent.h>*

*int state(const char *restrict path, struct stat *restrict buf);*	*Get file properties*	*#include<sys/stat.h >*

*int alphasort(const struct dirent **d1, const struct dirent **d2);*	*Sort an array in an alphabetical order*	*#include<dirent.h>*

Test Command:

csh> list
csh> list property
csh> list property time

Q3. Implement function to find the files under the current directory and subdirectories (5 marks)

Continue to use the code you implemented in Q2.

Here we design a function to find files under the current directory and its subdirectories. For example, the current directory is **C:\CSE_Lab\src\**. When we type in the find command with an argument as follows:

```
csh>find .txt
```

The function will print out the entries of all files with ".txt" in its name (as a suffix in the name). The output is as follows:

```
C:\CSE_Lab\src\Week1\SimpleShell.txt
C:\CSE_Lab\src\Week2\MergeSortThreaded.txt
C:\CSE_Lab\src\Week2\MultiThread.txt
C:\CSE_Lab\src\Week3\Bank.txt
C:\CSE_Lab\src\Week3\BankImpl.txt
C:\CSE_Lab\src\Week3\TestBank.txt
C:\CSE_Lab\src\Week5\FileOperation.txt
```

Note that you need to find files not only in current directory, but also in its subdirectories. So here you can implement a recursive function to find files at different levels of subdirectories,

Suggested Function:

Function	Usage	header
strstr	*Locate a substring in another string*	`#include<string.h>`
atoi	*Convert string to integer*	`#include<ctype.h>`

Test Command:

csh> find .txt
csh> find .xyz

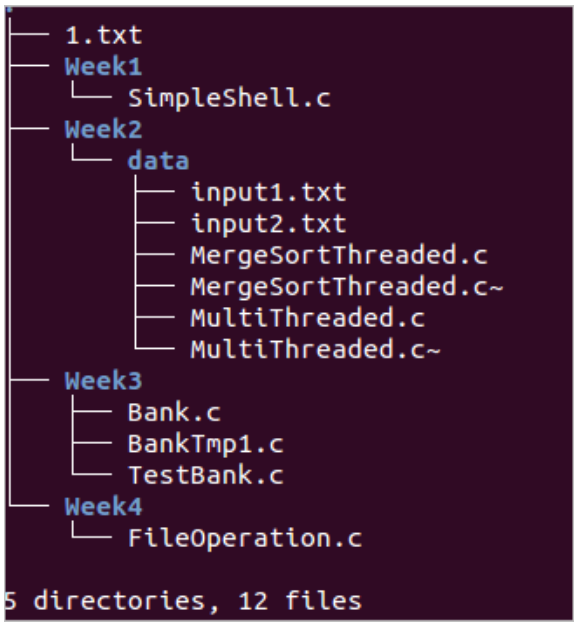
Q4. Implement function to list subdirectories and files in a tree structure (5 marks)

Continue to use the code you implemented in Q3.

In order to efficiently show files at different level of directories, we need a function to show files in a tree structure. For example, when we type in the *tree* command under the directory **C:\CSE_Lab\src\**:

```
csh>tree
```

The output should be as follows:



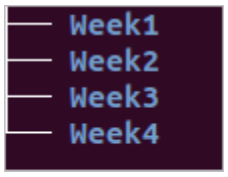
```
├── 1.txt
├── Week1
│   └── SimpleShell.c
├── Week2
│   └── data
│       ├── input1.txt
│       ├── input2.txt
│       ├── MergeSortThreaded.c
│       ├── MergeSortThreaded.c~
│       ├── MultiThreaded.c
│       └── MultiThreaded.c~
├── Week3
│   ├── Bank.c
│   ├── BankTmp1.c
│   └── TestBank.c
└── Week4
    └── FileOperation.c

5 directories, 12 files
```

- Sometimes we only want to see a few of the upper levels of a directory. We should be able to control the maximum level of subdirectories to be shown. For example, when we type in the *tree* command with the argument 1 in the directory **C:\CSE_Lab\src\**:

  ```
  jsh>tree 1
  ```

  The output should be as follows:



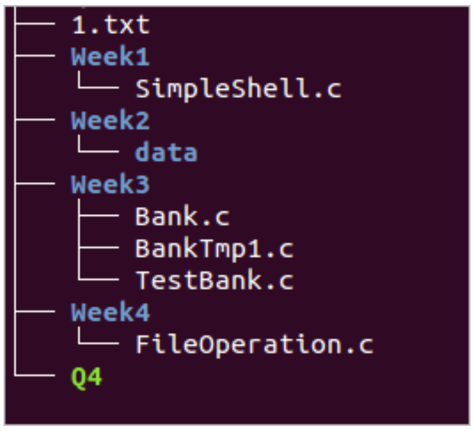
```
├── Week1
├── Week2
├── Week3
├── Week4
```

As you can see here, we set the maximum level of subdirectories to be 1, so only the first level of subdirectories is shown.

Similar to the list command, we can also decide which file property we use to sort the file order to print. For example, when we type in tree command under directory **C:\CSE_Lab\src\**:

```
jsh>tree 2 time
```

The output should be as follows:



```
├── 1.txt
├── Week1
│   └── SimpleShell.c
├── Week2
│   └── data
├── Week3
│   ├── Bank.c
│   ├── BankTmp1.c
│   └── TestBank.c
├── Week4
│   └── FileOperation.c
└── Q4
```

The tree structure is listed in *time* order. So the sequence is different from the one using *name* order.

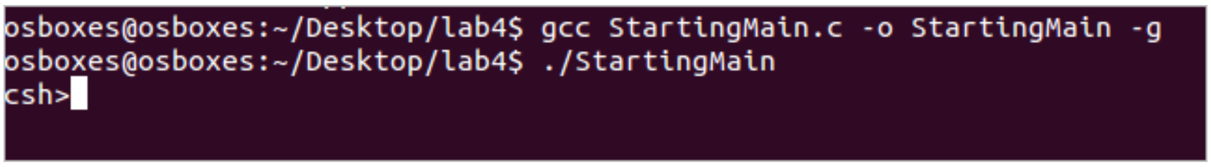
Suggested Function:

command	Class which function belongs to
.listFiles()	*File*
*int scandir(char *dirp, struct dirent ***namelist, int (*filter)(struct dirent *), int (*compare)(struct dirent **, struct dirent **));*	*#include<dirent.h>*

Test Command:

csh> tree
csh> tree 2
csh> tree 5

The starting codes in C for Lab 4 can be found in eDimension:



```
osboxes@osboxes:~/Desktop/lab4$ gcc StartingMain.c -o StartingMain -g
osboxes@osboxes:~/Desktop/lab4$ ./StartingMain
csh>
```