SIMPLILEARN PROJECT DOCUMENTATION

Alexandra Arseni, Vodafone

alexandra.arseni@vodafone.com

Contents

Personal Information

Name: Alexandra Arseni

Email: alexandra.arseni@vodafone.com

Github URL: alexarseni/PhaseEndProject (github.com)

About this document

This document contains a concise explanation of the organization and logic behind this application. No code or screenshots are presented in this document. For the screenshot document please find the “Alexandra-Arseni-Screenshots” document. For the code document please find the “Alexandra-Arseni-Code” document.

The app’s architecture

The code is organized in four different packages:

- virtualkey.main, which contains the main method of the application
- virtualkey.util, which contains the mainMenu and submenu user interfaces
- virtualkey.service, which contains all the business methods of the application

Note: the code has been tested in the provided lab’s environment. The code has not been tested in a windows environment.

virtualkey.main

This package contains the VitualKeyApp class, which contains the main method that starts the execution of the code.

VirtualKeyApp

The VirtualKeyApp class, has the main method that starts the execution of the application. The main method calls the class’ welcome method that is used only to display the application information and developer’s details. Then, the main method calls the mainMenu method of the MainMenu class.

vitualkey.util

This package contains the MainMenu and SubMenu classes.

MainMenu

Attributes:

No attributes

Methods:

void mainmenu();

Structure:

The MainMenu class is used to manage the first interaction with the user, through its mainMenu method. The method uses a do{}while() loop to display messages to the user and get the user input continuously. At first, the user is prompted with the possible choices:

1. Display all files currently present, by calling the getFiles() method. This method returns a sorted array of strings. We then use a for-each loop to traverse through the array and display each item.
2. Access the submenu and its business methods, by calling the submenu() method.
3. Exit the application. A boolean variable is used to exit the loop. When the user inputs the number 3, the boolean variable "exit" is assigned the true value, which is the condition the while loop is waiting for to finish.

The user must enter a number in the range 1-3. If the user enters a number that is not in the 1-3 range, he will be notified with the respective message.

Exceptions:

If the user inputs an alphanumerical word instead of a number, then an InputMismatch exception will be thrown. The exception will be handled internally and prompt the user with the message "What you typed was not a number".

SubMenu

Attributes:

No attributes

Methods:

void submenu(virtualkeyservice vks);

Parameters: VirtualKeyService object
Return Type: NA

Structure:

The submenu() method follows the same structure as the mainMenu() method, but also takes the VirtualKeyService object as an argument from the mainMenu method. The core of the submenu() method is the do{}while() loop with the switch command. The switch now has four possible options:

1. Add a user specified file in the directory, for which the addFile() method of the VirtualKeyService class is called. The user is prompted to input the name of the file. The method returns a string to inform the user whether the operation was successful. The method is case sensitive.
2. Delete a user specified file, for which the user is prompted to input the name of the file that is to be deleted and then the deleteFile() method is called. The method is case sensitive.
3. Search for a user specified file, for which the user is prompted to enter the name of the file that is to be searched, and then the searchFile() method is called. The method is case sensitive.
4. Exit the submenu interface and go back to mainMenu's interface.

Exceptions:

If the user inputs an alphanumerical word instead of a number, then an InputMismatch exception will be thrown. The exception will be handled internally and prompt the user with the message "What you typed was not a number".

virtualkey.service

VritualKeyService

This class contains all the business methods of the application.

Attributes:

`destination`: This attribute of type File determines the path where all file handling operations will take place. We have set the “./VirtualKeyFiles“ as the default directory.

```
File destination = new File("./VirtualKeyFiles");
```

Methods:

constructor:

The constructor checks whether the path declared by the ‘destination’ attribute already exists. If it does not, then it creates it.

void createSomeFiles();

Parameters: NA
Return Type: NA

Structure:
This method uses the createNewFile() method to create some initial files in the `destination`. This is an optional method created only to make to testing easier.

Exceptions:
This code catches the IOException, in the case that something goes wrong while creating the files.

string[] getFiles();

Parameters: NA
Return Type: Array of strings

Structure:
The method uses the String[] java.io.File.list() method, to get an array of strings, naming the files in the `destination` directory. If the returned array is not empty, we then use the Arrays.sort() method to sort the array in ascending order. Finally, we return the array. Displaying the elements of the array is done in the mainMenu() method.

Exceptions:
No exceptions are raised in the execution of this code.

string addFile(string filename);

Parameters: String filename
Return Type: String

Structure:
This method uses the boolean java.io.File.createNewFile() to create a new file with the specified name. If the return value of the createNewFile() is True, then the addFile() method returns the successful message to the user. If the return value is False, then the user is notified that a file with the same name already exists.

Exceptions:
The createNewFile() method throws an IOException if an I/O error occurred.

string deletefile(string filename);

Parameters: String filename
Return Type: String

Structure:
This method uses the java.io.File.exists() method to check whether a file with the user-specified name actually exists. If it does, then it proceeds to delete the file with the java.io.File.delete() method.

Exceptions:
No exceptions are thrown with the execution of this code.

string searchfile(string filename);

Parameters: String filename
Return Type: String

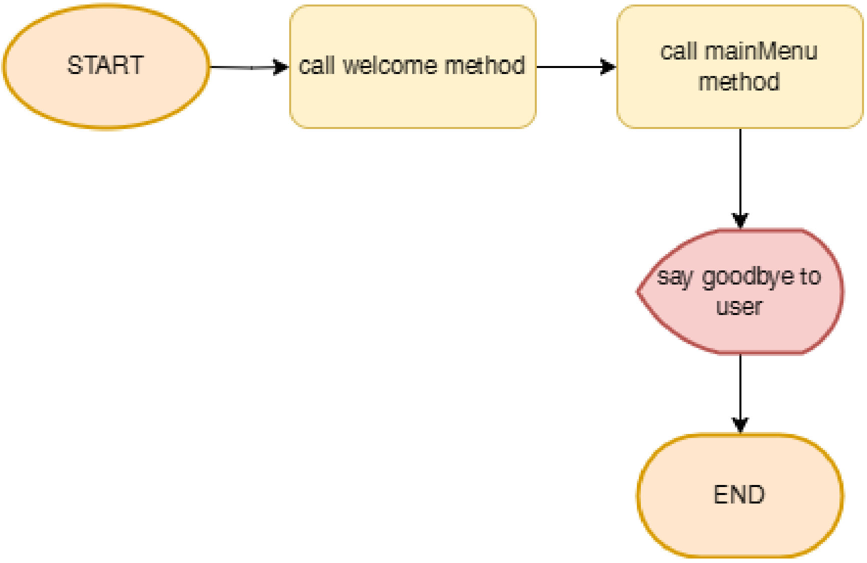
Structure:
This method uses the java.io.File.exists() method to notify the user whether the file they are searching for actually exists. The corresponding message is then returned to the user in a string.

Exceptions:
No exceptions are raised in the execution of this code.

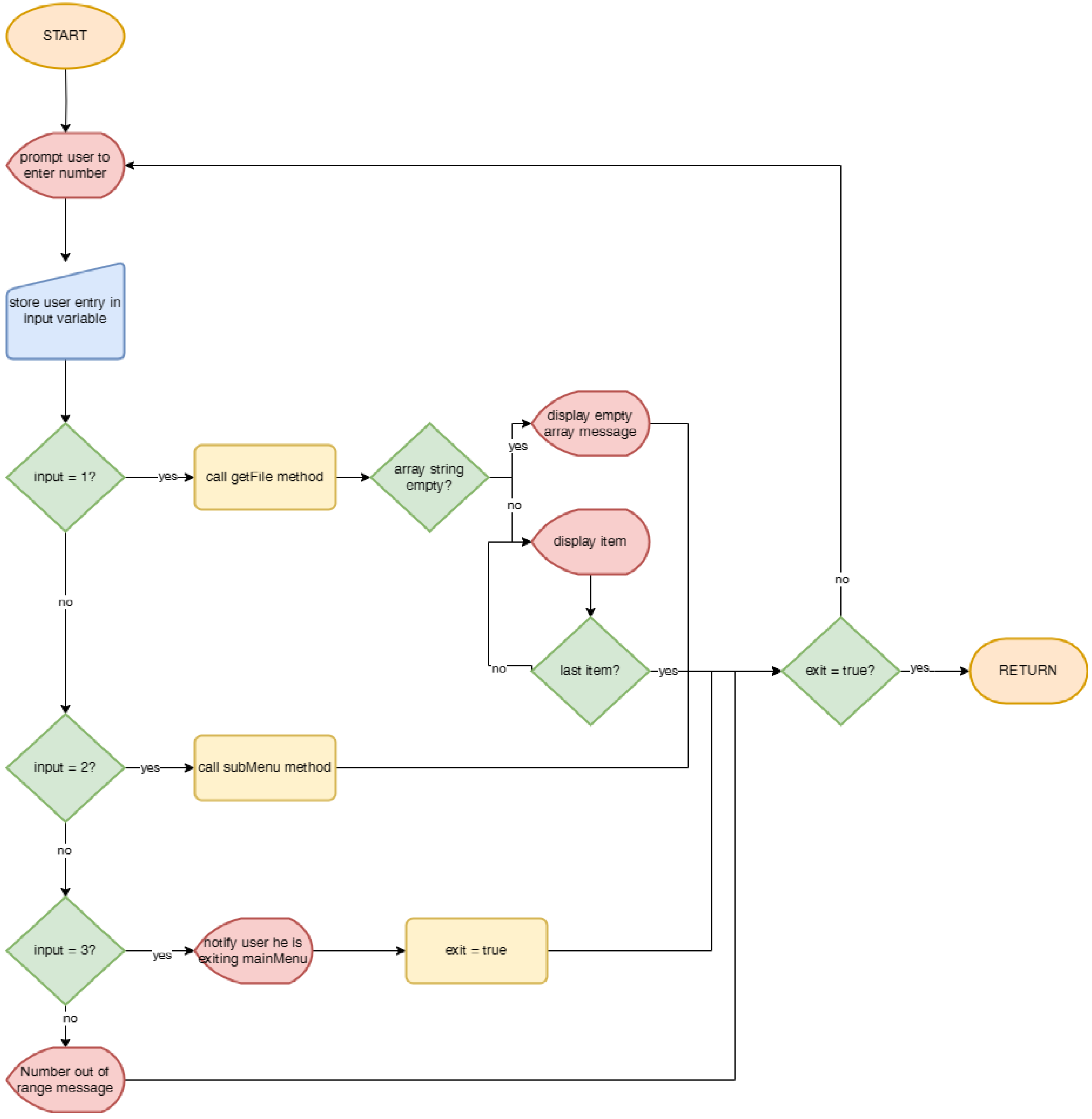
Flow charts

Note: The flow charts do not show the execution flow in case of an exception.

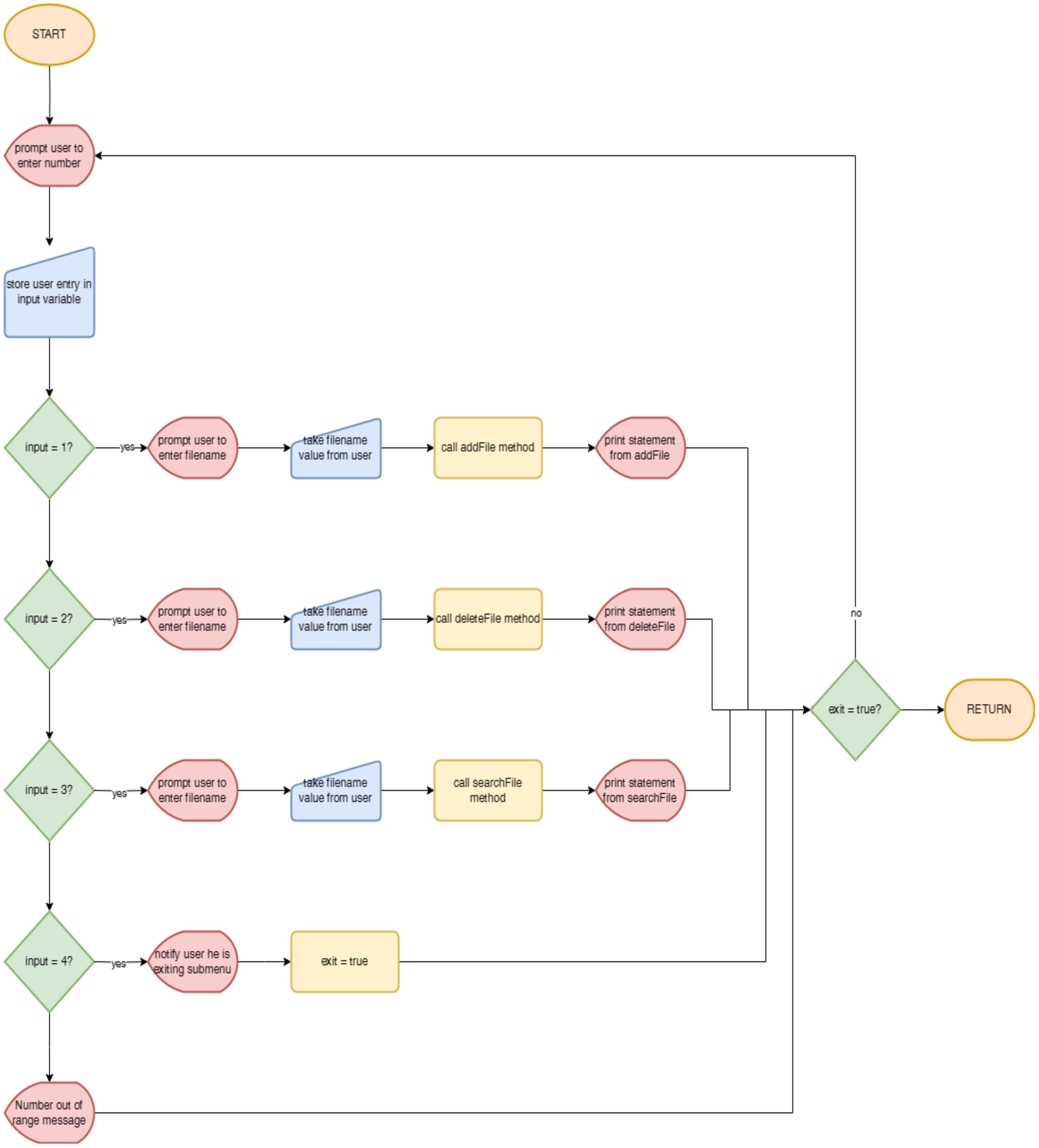
Main



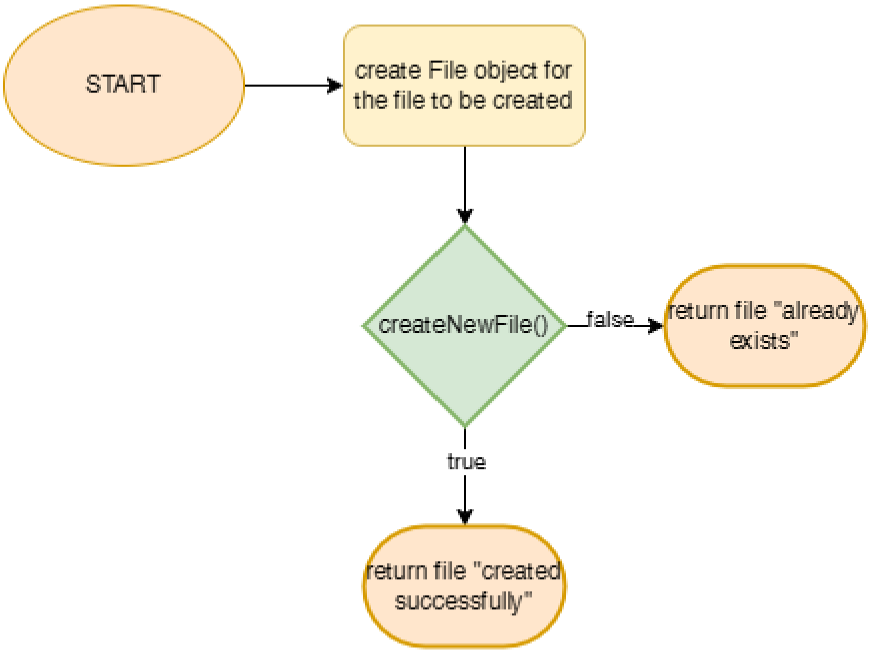
mainMenu


START
prompt user to enter number
store user entry in input variable
input = 1?
yes
call getFile method
array string empty?
yes
display empty array message
no
display item
last item?
no
yes
exit = true?
no
yes
RETURN
no
input = 2?
yes
call subMenu method
no
input = 3?
yes
notify user he is exiting mainMenu
exit = true
no
Number out of range message

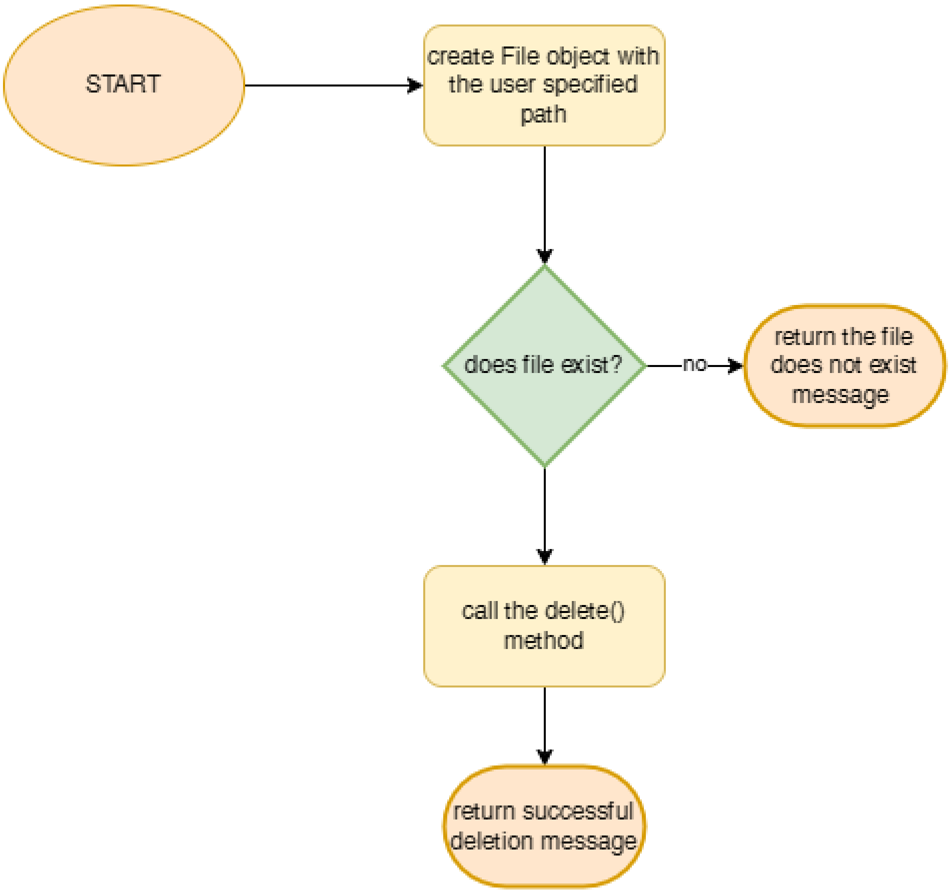
subMenu


START
prompt user to enter number
store user entry in input variable
input = 1?
yes
prompt user to enter filename
take filename value from user
call addFile method
print statement from addFile
input = 2?
yes
prompt user to enter filename
take filename value from user
call deleteFile method
print statement from deleteFile
input = 3?
yes
prompt user to enter filename
take filename value from user
call searchFile method
print statement from searchFile
input = 4?
yes
notify user he is exiting submenu
exit = true
Number out of range message
exit = true?
no
RETURN

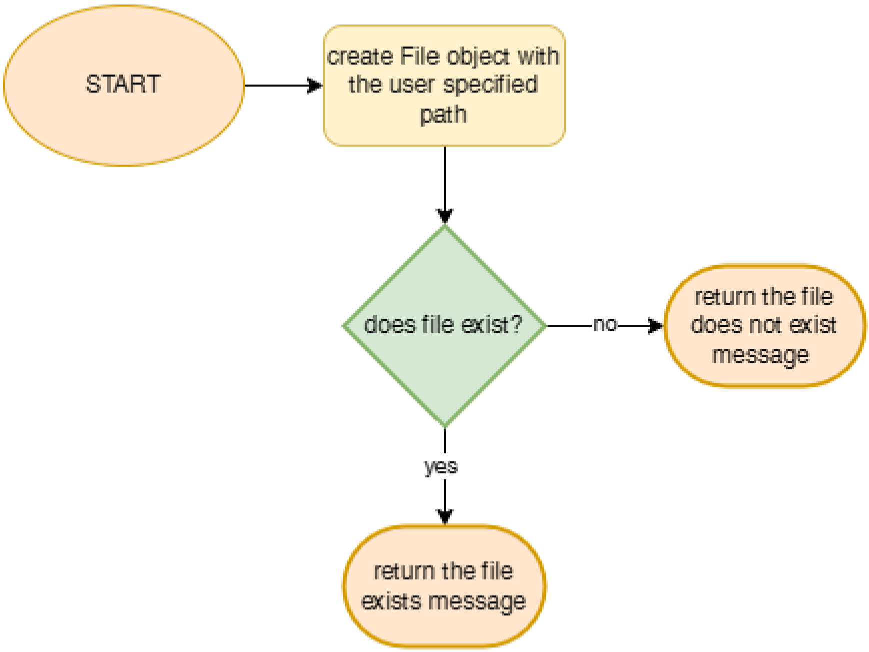
addFile


START
create File object for the file to be created
createNewFile()
false
return file "already exists"
true
return file "created successfully"

deleteFile


START
create File object with the user specified path
does file exist?
no
return the file does not exist message
call the delete() method
return successful deletion message

searchFile


START
create File object with the user specified path
does file exist?
no
return the file does not exist message
yes
return the file exists message