COMP2021 Project User Manual

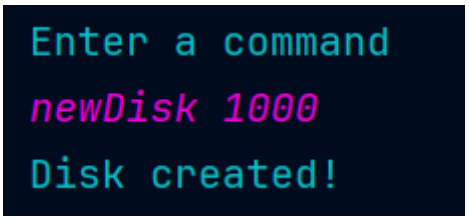
Introduction

The **CVFS** is an **in-memory Virtual File System (VFS)** in Java. It allows **uniform access to files** located in memory. The CVFS provides a **command line interface** tool to facilitate the use of virtual disks. Users can create virtual disks, directories, and documents, then perform multiple functions on them.

Commands Description

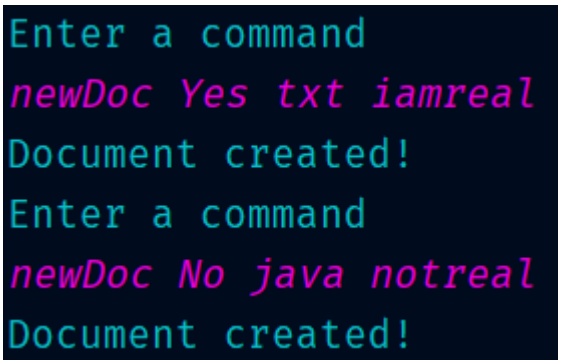
newDisk:

When the "***newDisk***" command is implemented, it **creates a new disk** and **parses the disk size** from users' input. Successful disk creation sets the new disk as the working disk, and the working directory is set to be the root directory of the disk.



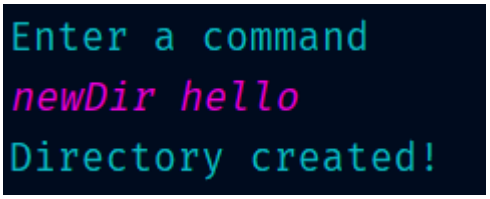
newDoc:

When the "***newDoc***" command is implemented, it **creates a new doc** in the working directory. It parses the document name, constructs the document with specified name, type, content and checks if the current disk has enough space. If success, the system updates the disk size and manages command history.



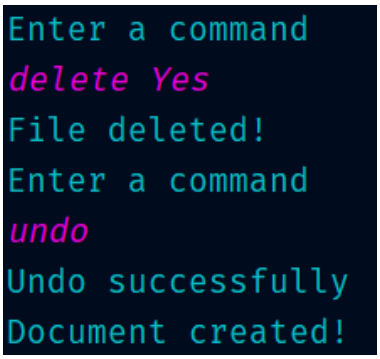
newDir:

When the "***newDir***" command is implemented, it **creates a new directory** in the working directory. It checks the validity of directory name and if input is valid, it successfully creates a Directory object and verifies availability of disk space. When creation is successful, it updates the disk size and manages command history.



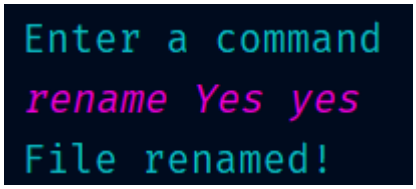
delete:

When the "***delete***" command is implemented, it **removes an existing file** from the working directory. It checks if the given file exists and if found, it retrieves the file's details for potential undo purposes. Once success, it updates the disk size and lists the remaining files in the directory.



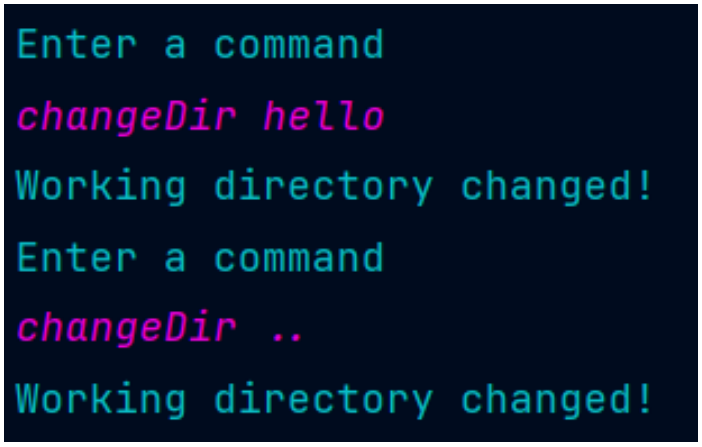
rename:

When the "***rename***" command is implemented, **renaming an existing file** in the working directory is allowed. It checks if the name is valid and verifies the existence of of the given old file. If both checks pass, the rename is proceed and updates the file list accordingly.



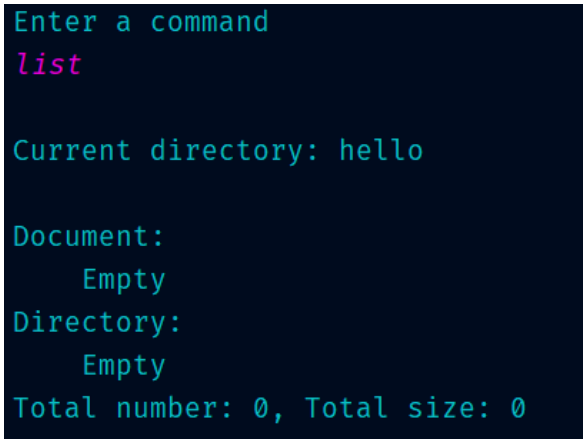
changeDir:

When the "***changeDir***" command is implemented, user can **navigate between directories** in the working directory. It checks if the given directory exists. If the input is "..", user can go the parent directory of the file. It checks if the current directory is already the root of the disk, if not, it moves user to the parent directory. When change is successful, it updates the working directory and displays the new directory contents. It also manages the command history for potential undo operations.



list:

When the "***list***" command is implemented, the system **displays all files and directories** in the working directory. It counts the number of files listed and calculate the total size of the listed files. It displays the name, type and size of each document along with directory's name and size. If there are no documents or directories, it states that the directory is "Empty". It finally reports the total number of files and total size of those files.



rList:

When the "***rList***" command is implemented, the system provides the **listing function in a recursive way**. It formats the output with indentation to indicate the hierarchy level of each intern. For each document, its name, type and size are displayed, and their respective directory's name and size would be displayed too. If no documents or directory exist, it states that the directory is "Empty". It returns the total number of files displayed and the total size of those files.

```
Enter a command
changeDir ..
Working directory changed!
Enter a command
rlist

Directory: Disk 1
    Document:
        Name: No, Type: java, Size: 54 bytes
        Name: yes, Type: txt, Size: 54 bytes
    Directory:
        Directory: hello, size: 40
            Document:
                Empty
            Directory:
                Empty
Total number: 3, total size: 148
```

newSimpleCri:

When the "***newSimpleCri***" command is implemented, it **creates a simple criterion** for filtering files. It checks the criterion name, attribute, operator and value based on restrictions for name, type and size. Once the checking is passed, it adds the criterion to the list, while error prompt appropriate messages, managing history potential undo.

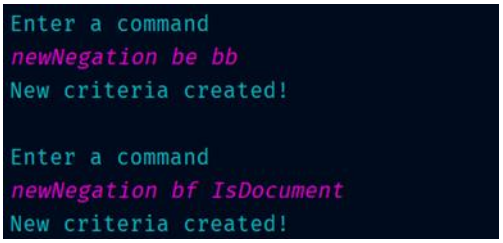
```
Enter a command
newSimpleCri ba size > 46
New criteria created!

Enter a command
newSimpleCri bb type equals "css"
New criteria created!

Enter a command
newSimpleCri bc name contains "poc"
New criteria created!
```

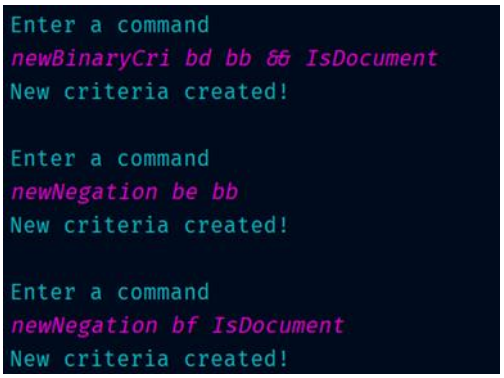
newNegation:

When the "***newNegation***" command is implemented, it **creates a composite criterion** that negates an existing criterion. This command and the "newBinaryCri" command both checks the criterion names and existence before adding the new criteria to the list.



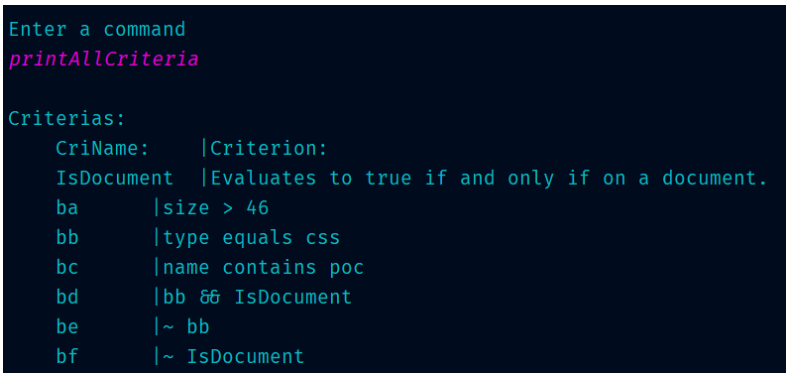
newBinaryCri:

When the "***newBinaryCri***" command is implemented, it constructs a criterion combining two existing criteria using a logical operator (&& or ||).



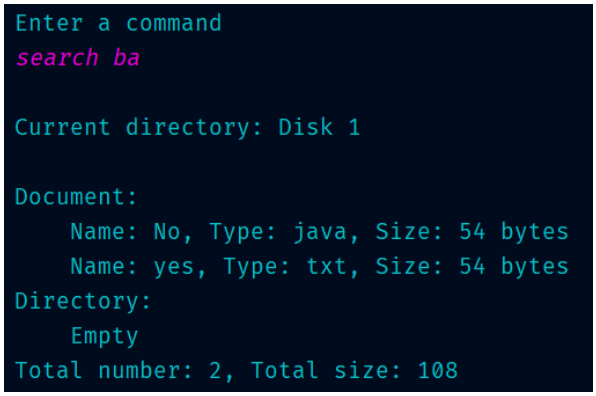
printAllCriteria:

When the "***pritnAllCriteria***" command is implemented, it **displays all defined criteria** in a structured format. It iterates through the criteria list, showing all criterion's name with its components, such as attribute name, operator, value, logical operator, or IsDocument. This shows a clear overview of of all criteria available for file filtering.



search:

When the "***search***" command is implemented, it helps users to **find files in the working directory** based on an existing criterion. It first checks if the given criterion exists. If the criterion is found, it retrieves the corresponding criterion and uses it for file filtering in the working directory. It then lists all matching files and reports the total number of files and total size of file listed. It shows a clear result of the search operation.



rSearch:

When the "***rSearch***" command is implemented, it functions as a **recursive search for files** in the working directory based on an existing criterion. It checks the existence of the given criterion before retrieving it. Then it calls a method to perform the recursive search, listing all matching files and reporting the total number of files and their total size.

save:

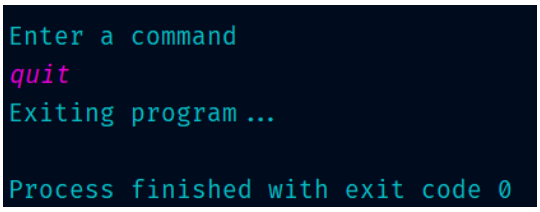
When the "***save***" command is implemented, it enables users to **save the current working virtual disk** and its **files** to a **given file path** on the local file system. It checks if a search criterion is applied, if so, it gets the relevant files based on the criterion and saves them to a temporary directory before writing them to the given path. If no criteria are found, it directly saves the whole working directory.

load:

When the "***load***" command is implemented, it enables user to **load a virtual disk** from a **given file path**. It uses a method called "loadfold" to recursively read the directory structure, adding subdirectories and documents. Each document is created by reading its content and type, allowing users to restore their previous work.

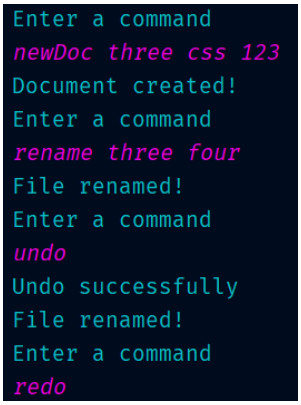
quit:

When the "***quit***" command is implemented, it enables users to **terminate the system's execution**. Once invoking the command, the program prints a message stating that it is exiting and then calls System.exit(0), effectively shutting down the application. Valid inputs include "quit", "Quit", "QUIT".



undo:

When the "***undo***" command is implemented, it supports "newDoc", "newDir", "delete", "rename", "changeDir", "newSimpleCri", "newNegation" commands to **perform undoing actions**. Each command pushes its detail onto command history and reverse the command effect.



redo:

When the "***redo***" command is implemented, it supports "newDoc", "newDir", "delete", "rename", "changeDir", "newSimpleCri", "newNegation" commands to **perform redoing actions**. Each command pushes its detail onto command history and re-apply them.

Step-by-Step Instructions

All commands should be executed **with correct upper/lower cases, except the "quit" command.**

For example, "newDisk 10" will function, but "newdisk 10" won't function.

For all commands, except "quit", must be input after creating **at least one** new disk.

The following are how the command should be inputted, where **command name** is in underline and **required inputs** are in *italics*.

newDisk *diskSize*

- use this command before any other commands except "quit"
- The input disk size is the maximum size of the virtual disk
- the size should not be negative
- Both newDoc and newDir requires disk size to be created, so the disk size is recommended to be around at least few hundred

newDoc *name type content*

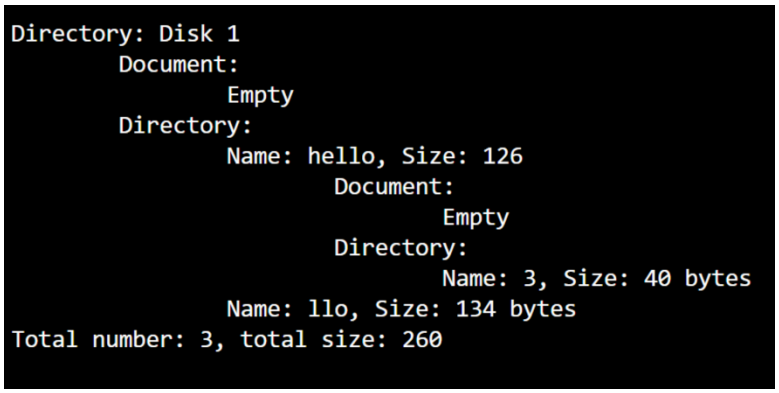
- name should contain only 10 English letters or numbers
- Different capitalization with same name (i.e. "doc1", "Doc1") is considered as different name
- type can only be "txt", "java", "html", "css"
- documents without content is allowed
- The size of a document is calculated as 40 + content.length*2

newDir *name*

- name should contain only 10 English letters or numbers
- Different capitalization with same name (i.e. “dir1”, “Dir1”) is considered as different name
- The size of a directory is calculated as 40 plus the total size of its contained files

delete *fileName*

- the file name should be a document’s or a directory’s name that already existed in the current directory
- However, if the file name is inside of a directory that is in the current directory, this command will not work except change the current directory to that directory.
- For example,
  Assume the situation is like the picture on the right. If the directory named “3” wanted to be removed, this action can only be executed correctly if changed to the directory “hello” first. Then, use “delete 3” to remove that directory.



rename *oldFileName newFileName*

- the *oldFileName* should be a document’s or a directory’s name that is existing in the current directory
- the *newFileName* should satisfy the name requirement as mentioned in the newDoc part: Different capitalization with same name (i.e. “doc1”, “Doc1”) is considered as different name

changeDir *dirName*

- the *dirName* should be an existing directory that is in the current directory (I.e. the name should be totally same)
- If want to change the current directory be the parent directory, e.g. "disk 1/hello/3" to "disk 1/hello", then *dirName* should be ".." (changeDir ..)
- changeDir is needed if want to create files in current directory's directories

list

- list out the documents' name and size, directories' name and size, total file numbers (sum of documents and directories numbers), and the total file size

```
Current directory: Disk 2

Document:
        Name: hi, Type: txt, Size: 52 bytes
Directory:
        Name: 10, Size: 80 bytes
Total number: 2, Total size: 132
```

rList

- list out all the files that are inside the current directory

example →

```
Directory: Disk 2
        Document:
                Name: hi, Type: txt, Size: 52 bytes
        Directory:
                Directory: 10, size: 80
                        Document:
                                Empty
                        Directory:
                                Directory: 11, size: 40
                                        Document:
                                                Empty
                                        Directory:
                                                Empty
Total number: 3, total size: 132
```

newSimpleCri *criName attrName op val*

- *criName* can only be exactly two English characters
- *attrName* can be **type** (op: **equals**) (val: string in double quote)
  , **name** (op: **contains**) (val: string in double quote)
  , or **size** (op: **>, <, >=, <=, ==, !=**) (val: integer)

```
newSimpleCri ba size > 47
newSimpleCri bb type equals "css"
newSimpleCri bc name contains "lo"
```

newNegation *criName1 criName2*

- existing criteria (either isDocument or those created by newSimpleCri) is required before saving it as a new negation criterion.
- For instance, *"*newNegation *bf IsDocument"* is a negation criterion that excludes all results which are documents.

newBinaryCri *criName1 criName3 logicOp criName4*

- Similar to the "newNegation", this command requires existing criteria in order to perform logical operations on themselves as a new binary criterion.
- *logicOp* is either && or ||
- For example, "newBinaryCri *bd bb && IsDocument*" performs logical operation AND, where the results of bd should satisfy both bb and isDocument.

printAllCriteria

- All existing criteria would be printed out, including user-defined criteria and isDocument.

search *criName*

- Search all the files directly contained in the working directory that satisfy criterion *criName*.
- total number and size of files will also be listed.
- Note that the working directory of the resulting document(s) will also be listed, as well as the total count and size printed.

rSearch *criName*

- This works similarly to rList, but the output will be the files that meet the criteria with name *criName*
- If the file that meets the criteria is inside of a directory, that directory will be considered as meeting the criteria as well
- For example;

Original ->

```
rlist

Directory: Disk 1
        Document:
                Name: hi, Type: txt, Size: 64 bytes
        Directory:
                Directory: 10, size: 128
                        Document:
                                Name: hey, Type: java, Size: 48 bytes
                        Directory:
                                Directory: ab, size: 40
                                        Document:
                                                Empty
                                        Directory:
                                                Empty
                Directory: 20, size: 40
                        Document:
                                Empty
                        Directory:
                                Empty
Total number: 5, total size: 232
```

With the criteria of IsDocument ->

The directory "10" will be involved also as it the document "hey" is inside of it.

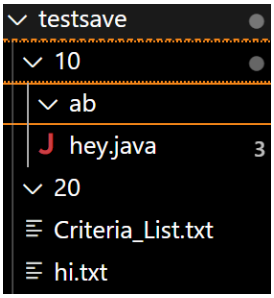
```
Directory: Disk 1
        Document:
                Name: hi, Type: txt, Size: 64 bytes
        Directory:
                Name: 10, Size: 128
                        Document:
                                Name: hey, Type: java, Size: 48 bytes
                        Directory:
                                Empty
Total number: 3, total size: 192
```

The size of the directory involved will be the original size of that as the rSearch can be used with size, which if the size changes during the rSearch process, then the output will be shown wrongly

save *path*

- This can save the current directory files into a real folder with *path*
- the path should be a real existing folder that is inside the folder that was chose in the compiler
  For example, if the name of folder that was opened in the compiler is “abc”, and there is a folder named “forsave” just inside “abc”, the path will be **forsave**
- the *path* can be forsave/abc/a/b/c, which the files are wanted to be saved inside the folder “c”
- the criteria that was made can be saved as well

- For example:



load *path*

- This can load the real existing folders to CVFS with *path*
- This works similarly to save
- This can load the saved criteria as well

quit

- **Stop** the terminal and exiting the system

undo

- Allow undo certain commands
- can be used multiple times in a row
- Only the following commands can be undone: newDoc, newDir, delete, rename, changeDir, newSimpleCri and newNegation

redo

- Allow redoing the undo command that was executed before
  can be used multiple times in a row
- Only the following commands can be redone: newDoc, newDir, delete, rename, changeDir, newSimpleCri and newNegation

Troubleshooting

newDisk

- **Non-integer disk size: “fifty” / Negative disk size**

```
Enter a command
newDisk fifty
Invalid disk size. Please enter a valid integer.
Enter a command
```

```
Enter a command
newDisk -100
The disk size should not be negative!
Enter a command
```

Re-enter the command with non-negative integer disk size.

newDoc

- **Same document name**

```
Enter a command
newDoc doc1 txt This is doc1.
Document created!
Enter a command
newDoc doc1 css This is also doc1.
There is another file with the same name, please choose another name for this file
Enter a command
```

Duplicated document names are not allowed even if they have different document type. Re-enter the command to create the document.

- **Invalid document name**

```
newDoc invalid! txt This is invalid
Only digits and English letters are allowed in file names
Enter a command
newDoc ThisIsInvalid css This is too long
File name may have at most 10 characters and not empty
Enter a command
```

- **No document name/type (or “ ”)**

```
Enter a command
newDoc  txt No doc name.
File name may have at most 10 characters and not empty
Enter a command
newDoc doc1  No doc type!
Only types txt, java, html, and css are allowed
Enter a command
```

Make sure all commands are input correctly.

- **Out of size**

```
Enter a command
newDisk 40
Disk created!
Enter a command
newDoc noContent java out of size!!!
The disk is out of space to add this file.
Enter a command
```

Make sure the disk size is large enough.

newDir

- **Same directory name**

```
Enter a command
newDir dir1
Directory created!
Enter a command
newDir dir1
There is another file with the same name, please choose another name for this file
Enter a command
```

Duplicated directory names are not allowed. Re-enter the command to create the directory.

- **Invalid directory name**

```
Enter a command
newDir invalid!
Only digits and English letters are allowed in file names
Enter a command
newDir ThisisInvalid
File name may have at most 10 characters and not empty
Enter a command
```

- **Out of size**

```
Enter a command
newDisk 40
Disk created!
Enter a command
newDir Hello
Directory created!
Enter a command
newDir noContent
The disk is out of space to add this file.
Enter a command
```

Make sure the disk size is large enough.

delete

- **When deleting a file does not exist**

```
Enter a command
newDisk 1000
Disk created!
Enter a command
delete doc1
File not found!
Enter a command
```

```
Enter a command
newDoc doc1 html this is doc1.
Document created!
Enter a command
delete DOC1
File not found!
Enter a command
```

Make sure the file name is correct. Different capitalization is also considered as different document.

rename

- **Old file name not found**

```
Enter a command
newDoc doc2 css This is doc2.
Document created!
Enter a command
rename Doc2 docTwo
File not found!
Enter a command
```

Make sure the old file name is correct. Different capitalization is also considered as different documents.

- **Wrong new file name**

```
Enter a command
newDoc doc2 css This is doc2.
Document created!
Enter a command
rename doc2 doc_2
Only digits and English letters are allowed in file names
Enter a command
```

Similar to creating new file, the new file name should contain only digits and English letters, and each file name may have at most 10 characters.

newSimpleCri

- **Criteria name already exist**

```
Enter a command
newSimpleCri bc name contains "doc"
New criteria created!

Enter a command
newSimpleCri bc name contains "same"
Criteria name already exists!
Enter a command
```

Different attrName, type but same criName are also considered as same.

- **Invalid criteria name/ attribute name/ type**

```
Enter a command
newSimpleCri abc name contains "doc"
Please enter a two English letters for criteria name!
Enter a command
newSimpleCri bc name NOTcontains "doc"
If attribute is name, operator can only be "contains"
Enter a command
newSimpleCri bc name contains doc
If attribute is name, val can only be a string in the double quote!
Enter a command
newSimpleCri bb type equal "css"
If attribute is type, operator can only be "equals"
Enter a command
```

newNegation

- **Criteria name already exist**

```
Enter a command
newSimpleCri bc name contains "doc"
New criteria created!

Enter a command
newSimpleCri bb type equals "css"
New criteria created!
```

```
Enter a command
newNegation cb bc
New criteria created!

Enter a command
newNegation cb bb
Criteria name already exists!
Enter a command
```

Different criName2, type but same criName1 are also considered as same.

- **Invalid criteria name/ criteria not found**

```
Enter a command
newNegation cb bc
Criteria not found!
Enter a command
```

```
Enter a command
newSimpleCri bc name contains "doc"
New criteria created!

Enter a command
newNegation acb bc
Please enter a two English letters for criteria name!
Enter a command
```

Make sure a simple criteria is created before doing negation.

newBinaryCri

- **Criteria name already exist**

```
Enter a command
newSimpleCri bb type equals "css"
New criteria created!

Enter a command
newSimpleCri ab name contains "abc"
New criteria created!

Enter a command
newSimpleCri ac type equals "java"
New criteria created!

Enter a command
newNegation ae ab
New criteria created!

Enter a command
newBinaryCri af ae || ac
New criteria created!
```

```
Enter a command
newBinaryCri af ae && bb
Criteria name already exists!
Enter a command
```

Different criName3 and criName4, type but same criName1 are also considered as same.

- **Invalid criteria name/ logic operator**

```
Enter a command
newBinaryCri bd bb && IsDocument
Criteria not found!
Enter a command
```

Make sure a simple criteria is created before creating a new binary criteria.

```
Enter a command
newSimpleCri ab name contains "abc"
New criteria created!

Enter a command
newSimpleCri ac type equals "java"
New criteria created!

Enter a command
newNegation ae ab
New criteria created!
```

```
Enter a command
newBinaryCri ace ae || ac
Please enter a two English letters for criteria name!
Enter a command
```

```
Enter a command
newBinaryCri af ae & ac
Logic operator can only be && or || !
Enter a command
newBinaryCri af ae || ae
The two selected criteria can not be the same
Enter a command
```

search

- **Criteria not found**

```
Enter a command
search Disk1
Criteria not found!
Enter a command
```

Make sure the input criteria is created.

rSearch

- **Criteria not found**

```
Enter a command
rSearch criName
Criteria not found!
Enter a command
```

Make sure the input criteria is created.

redo

- **Redo without undo**

```
Enter a command
newDisk 1000
Disk created!
Enter a command
newDoc doc1 css testdoc!
Document created!
Enter a command
redo
Cannot redo without undo.
Enter a command
```

```
Enter a command
newDisk 1000
Disk created!
Enter a command
newDoc doc1 css testdoc!
Document created!
Enter a command
undo
Undo successfully
File deleted!
Enter a command
redo
redo successfully
Document created!
Enter a command
redo
Cannot redo without undo.
Enter a command
```

undo

- **Undoing unspecified commands**

```
Enter a command
newDisk 1000
Disk created!
Enter a command
undo
Previous step cannot be undo.
Enter a command
```

Only the following commands can be undo/redo: newDoc, newDir, delete, rename, changeDir, newSimpleCri, newNegation

other issues

- **Inputting any commands (except “quit”) without creating a new disk**

```
Enter a command
list
Enter a command
rList
Enter a command
printAllCriteria
Enter a command
newDoc doc1 txt cannot be created.
Enter a command
```

The system would ignore all the commands before creating a new disk. Make sure to create a new disk before inputting any commands.

- **Inputting any commands (except “quit”) with incorrect capitalization/ unknown commands**

```
Enter a command
newDISK 1000
Enter a command
new
Enter a command
```

The system would ignore all the commands. Make sure the commands are correct.

- **Insufficient number of commands inputted**

```
Enter a command
newDir
Error: please enter "newDir [dirName]".
Enter a command
```

```
Enter a command
newDisk
Error: please enter "newDisk [diskSize]".
Enter a command
```

```
Enter a command
rename doc1
Error: please enter "rename [oldFileName] [newFileName]".
Enter a command
```

The system would ignore the commands and return an error message. Make sure the number of input commands are correct.

Additional Resources

References:

https://coderanch.com/t/645585/java/Virtual-File-System