

cse15l-lab-reports

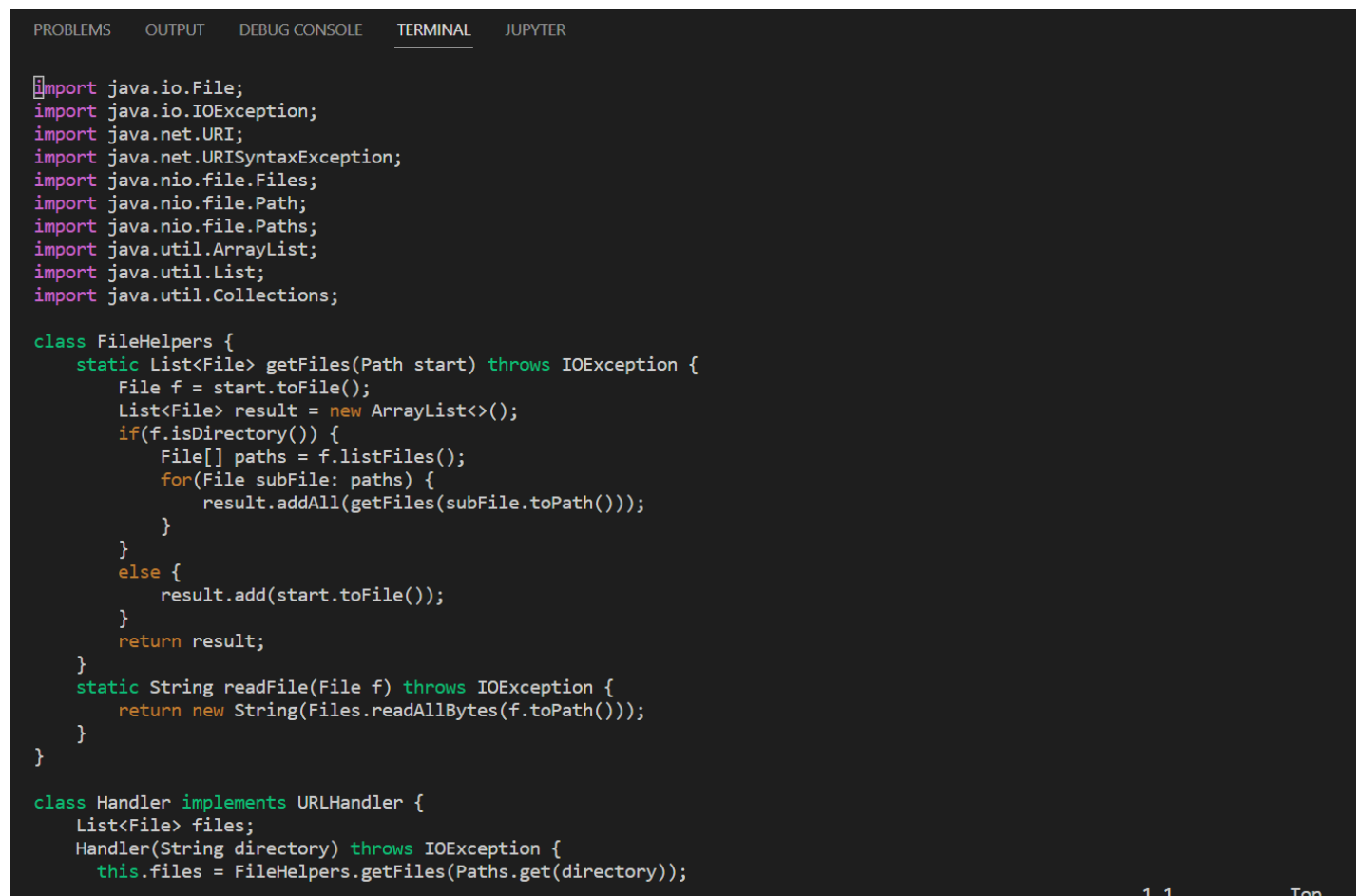
Lab Report 4

Part 1

Objective: Changing the `main` method to take a command-line argument. Solution:

```
/" <enter> NNCwArgs[1])); <escape> /FileH <enter> nnc$this.files; <escape> :wq <enter>
```

Start with the vim editor open on `DocSearchServer.java`



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

import java.io.File;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;

class FileHelpers {
    static List<File> getFiles(Path start) throws IOException {
        File f = start.toFile();
        List<File> result = new ArrayList<>();
        if(f.isDirectory()) {
            File[] paths = f.listFiles();
            for(File subFile: paths) {
                result.addAll(getFiles(subFile.toPath()));
            }
        }
        else {
            result.add(start.toFile());
        }
        return result;
    }
    static String readFile(File f) throws IOException {
        return new String(Files.readAllBytes(f.toPath()));
    }
}

class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.getFiles(Paths.get(directory));
    }
}
```

Typing in `/" <enter>` will move the cursor to the first instance of `"` in the file.

```

class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.GetFiles(Paths.get(directory));
    }
    public String handleRequest(URL url) throws IOException {
        List<File> paths = FileHelpers.GetFiles(Paths.get("./technical"));
        if (url.getPath().equals("/")) {
            return String.format("There are %d total files to search.", paths.size());
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {
                /"

```

38,58

19%

NNCW means find the 3rd instance of `/"` from bottom and then delete the next word and enter insert mode.

```

class DocSearchServer {
    public static void main(String[] args) throws IOException {
        if(args.length == 0){
            System.out.println("Missing port number! Try any number between 1024 to 49151");
            return;
        }

        int port = Integer.parseInt(args[0]);

        Server.start(port, new Handler("./technical/"));
    }
}
? "

```

c

74,40

Bot

```

class DocSearchServer {
    public static void main(String[] args) throws IOException {
        if(args.length == 0){
            System.out.println("Missing port number! Try any number between 1024 to 49151");
            return;
        }

        int port = Integer.parseInt(args[0]);

        Server.start(port, new Handler(
    }
}
-- INSERT --

```

74,40

Bot

Fill in the replaced word with `args[1]))`; should correctly have Handler read a command-line argument now.

```

class DocSearchServer {
    public static void main(String[] args) throws IOException {
        if(args.length == 0){
            System.out.println("Missing port number! Try any number between 1024 to 49151");
            return;
        }

        int port = Integer.parseInt(args[0]);

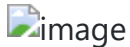
        Server.start(port, new Handler(args[1]))
    }
}

```

74,49

Bot

Like above, `/FileH` should find the first instance of `FileH`, but since the cursor is near the bottom of the file, no further instances are found, so it gets looped back to the top.



nnc\$ will find the 3rd instance from top and then delete anything between the cursor and the end of the line before entering insert mode.

```
class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.GetFiles(Paths.get(directory));
    }
    public String handleRequest(URI url) throws IOException {
        List<File> paths = FileHelpers.GetFiles(Paths.get("./technical"));
        if (url.getPath().equals("/")) {
            return String.format("There are %d total files to search.", paths.size());
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {
                c
                38,27
                19%
            }
        }
    }
}

class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.GetFiles(Paths.get(directory));
    }
    public String handleRequest(URI url) throws IOException {
        List<File> paths = 
        if (url.getPath().equals("/")) {
            return String.format("There are %d total files to search.", paths.size());
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {
                -- INSERT --
                38,27
                19%
            }
        }
    }
}
```

Fill in the replaced line with `this.files`; this way the file paths taken in as a command-line argument will be properly stored.

```
class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.GetFiles(Paths.get(directory));
    }
    public String handleRequest(URI url) throws IOException {
        List<File> paths = this.files;
        if (url.getPath().equals("/")) {
            return String.format("There are %d total files to search.", paths.size());
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {
                38,37
                19%
            }
        }
    }
}
```

Do not forget to then leave insert mode as well as save and exit with `:wq`.

```

import java.util.List;
import java.util.Collections;

class FileHelpers {
    static List<File> getFiles(Path start) throws IOException {
        File f = start.toFile();
        List<File> result = new ArrayList<>();
        if(f.isDirectory()) {
            File[] paths = f.listFiles();
            for(File subFile: paths) {
                result.addAll(getFiles(subFile.toPath()));
            }
        }
        else {
            result.add(start.toFile());
        }
        return result;
    }
    static String readFile(File f) throws IOException {
        return new String(Files.readAllBytes(f.toPath()));
    }
}

class Handler implements URLHandler {
    List<File> files;
    Handler(String directory) throws IOException {
        this.files = FileHelpers.getFiles(Paths.get(directory));
    }
    public String handleRequest(URI url) throws IOException {
        List<File> paths = this.files;
        if (url.getPath().equals("/")) {
            return String.format("There are %d total files to search.", paths.size());
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {

```

Part 2

Visual Studio Code: 60.15 seconds

Vim (Remote Desktop): 62.23 seconds

No major difficulties arose during either time trials.

1. At the moment, due to my unfamiliarity with Vim, I would prefer making all of my edits on the local desktop and then scp the changed files into the remote desktop. Possibly if I do become more fluent with using Vim and knowing all its feature shortcuts, I may find using Vim on the remote desktop to be the more efficient method.
2. If I am editing only a few files, then I might prefer to do all my work on the local desktop. However, if the task involves many files across many folders and workspaces, doing the edits on remote can be an advantage, since scp takes time and having to use cp to organize files on the remote desktop can be tedious.

[Back to index](#)

[Back to github](#)