Littlekawayi233 / cse15l-lab-reports Public

Code Issues Pull requests Actions Projects Wiki Security

main

cse15l-lab-reports / lab-report-4-week-7.md

Littlekawayi233 s ✓ History

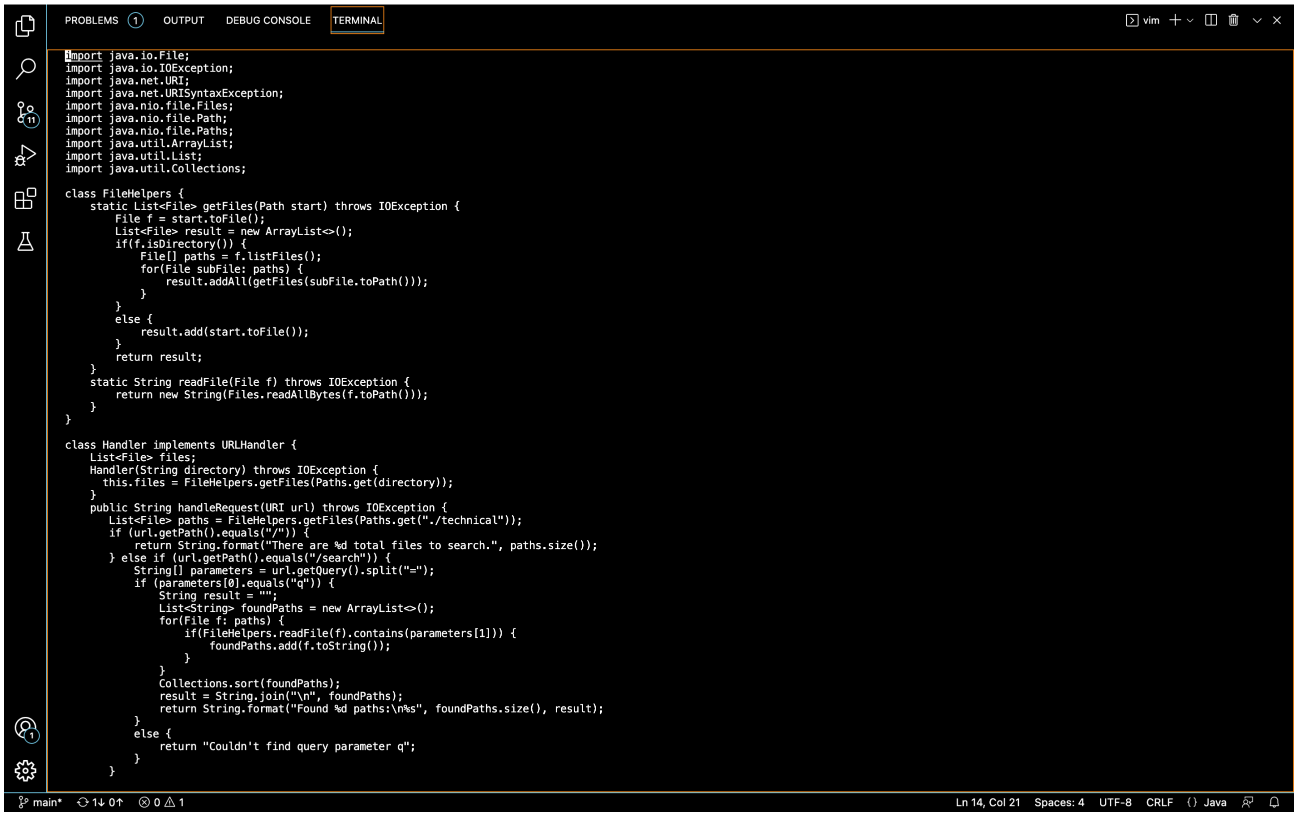
1 contributor

Part 1

Changing the name of the start parameter and its uses to base:

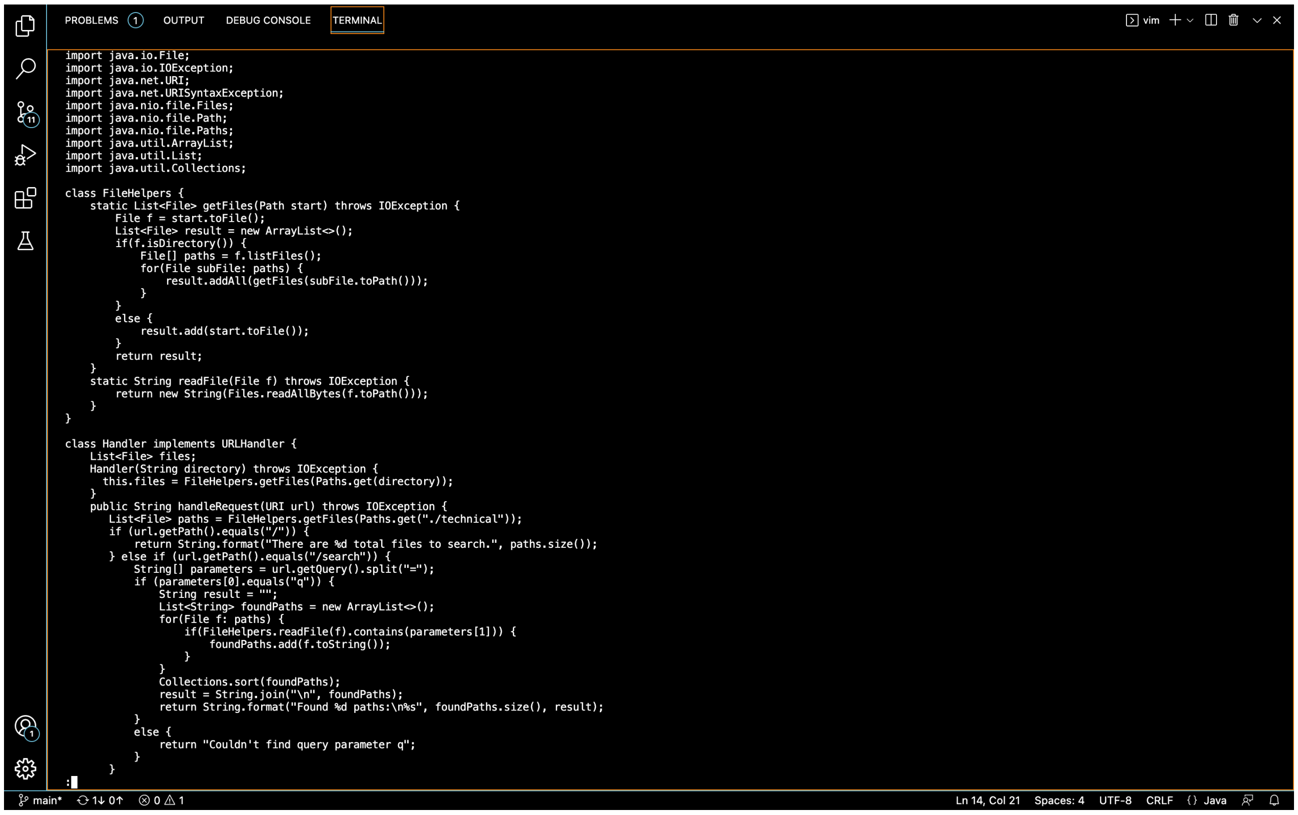
```
:%s/start/base/g<Enter>:w<Enter>
```

This is the original file.



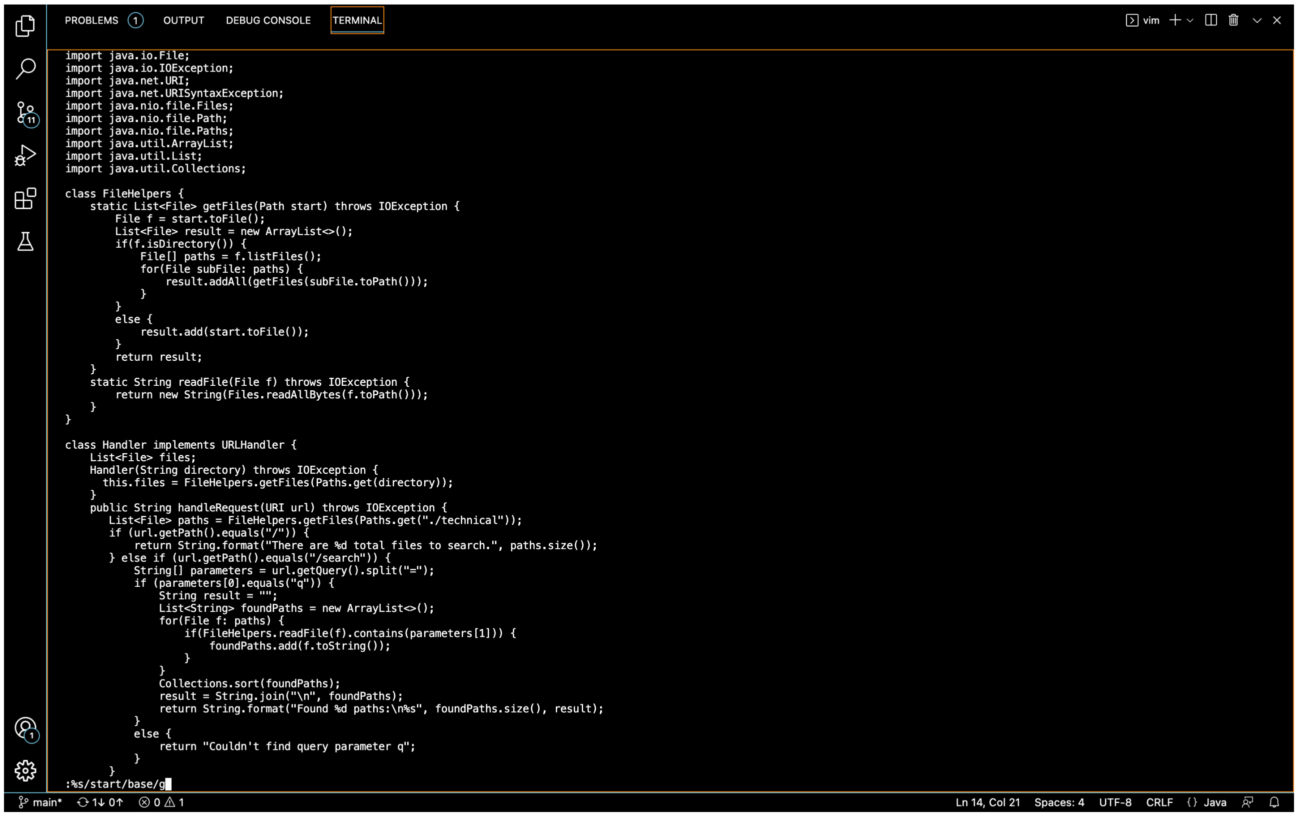
```
import java.io.File;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;

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));
    }
    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")) {
               String result = "";
               List<String> foundPaths = new ArrayList<>();
               for(File f: paths) {
                   if(FileHelpers.readFile(f).contains(parameters[1])) {
                       foundPaths.add(f.toString());
                   }
               }
               Collections.sort(foundPaths);
               result = String.join("\n", foundPaths);
               return String.format("Found %d paths:\n%s", foundPaths.size(), result);
           }
           else {
               return "Couldn't find query parameter q";
           }
       }
```

That's my first key pressed **:**



```
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));
    }
    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")) {
               String result = "";
               List<String> foundPaths = new ArrayList<>();
               for(File f: paths) {
                   if(FileHelpers.readFile(f).contains(parameters[1])) {
                       foundPaths.add(f.toString());
                   }
               }
               Collections.sort(foundPaths);
               result = String.join("\n", foundPaths);
               return String.format("Found %d paths:\n%s", foundPaths.size(), result);
           }
           else {
               return "Couldn't find query parameter q";
           }
       }
:
```

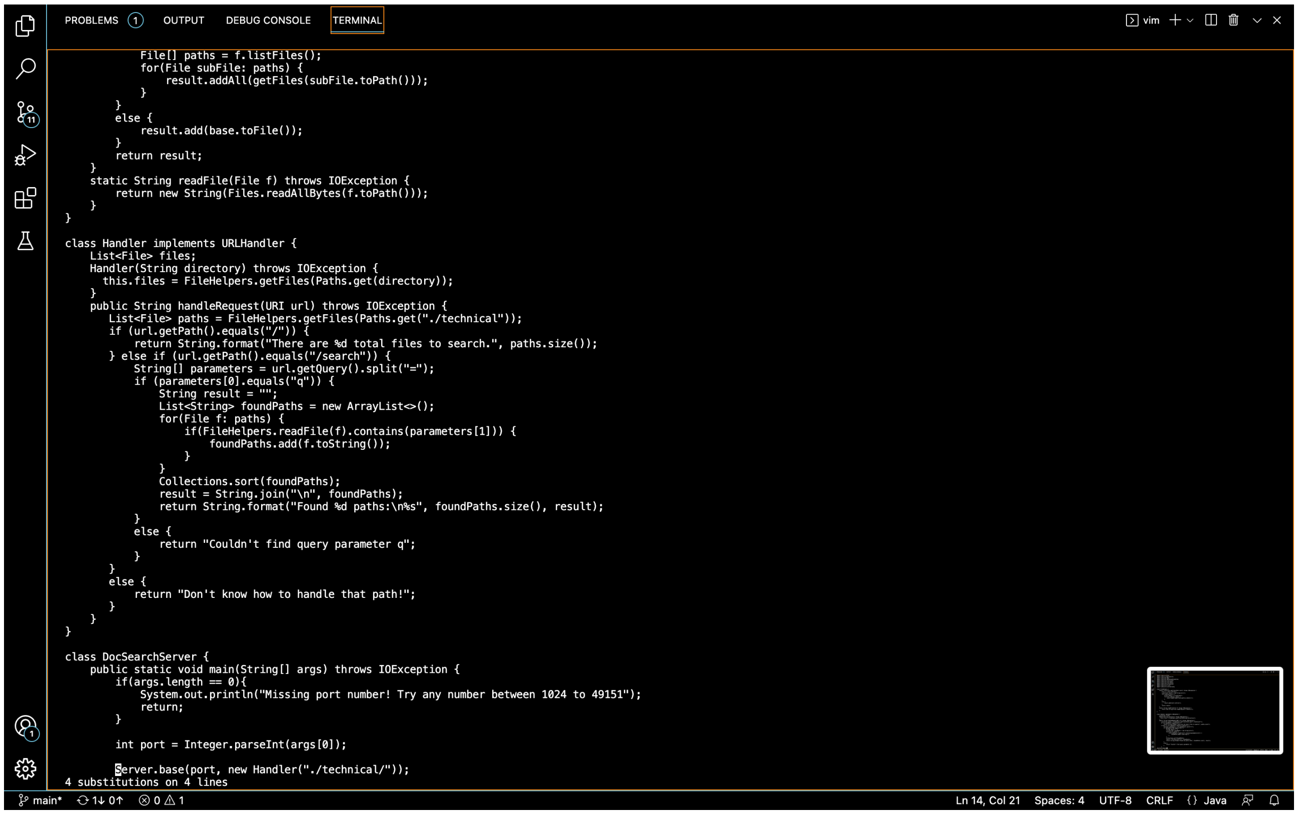
After typing `%s/start/base/g`



```
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));
    }
    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")) {
               String result = "";
               List<String> foundPaths = new ArrayList<>();
               for(File f: paths) {
                   if(FileHelpers.readFile(f).contains(parameters[1])) {
                       foundPaths.add(f.toString());
                   }
               }
               Collections.sort(foundPaths);
               result = String.join("\n", foundPaths);
               return String.format("Found %d paths:\n%s", foundPaths.size(), result);
           }
           else {
               return "Couldn't find query parameter q";
           }
       }
:%s/start/base/g
```

```
main*  1↓ 0↑  ⊗ 0 ⚠ 1                    Ln 14, Col 21  Spaces: 4  UTF-8  CRLF  {} Java
```

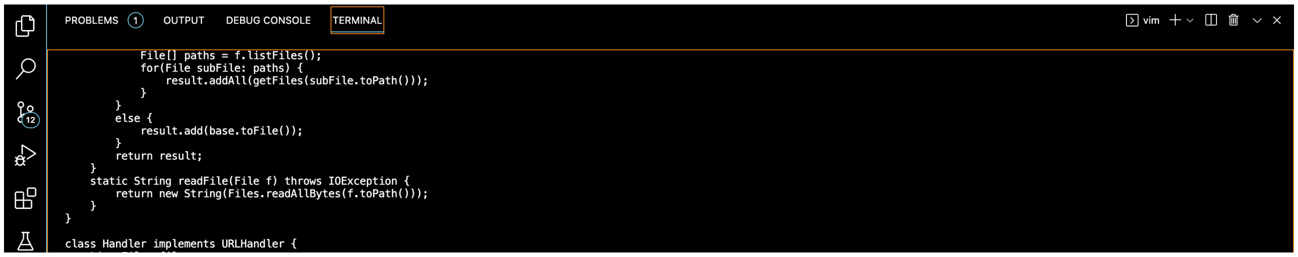
After pressing `<Enter>`



```
            File[] paths = f.listFiles();
            for(File subFile: paths) {
                result.addAll(getFiles(subFile.toPath()));
            }
        }
        else {
            result.add(base.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 = 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")) {
               String result = "";
               List<String> foundPaths = new ArrayList<>();
               for(File f: paths) {
                   if(FileHelpers.readFile(f).contains(parameters[1])) {
                       foundPaths.add(f.toString());
                   }
               }
               Collections.sort(foundPaths);
               result = String.join("\n", foundPaths);
               return String.format("Found %d paths:\n%s", foundPaths.size(), result);
           }
           else {
               return "Couldn't find query parameter q";
           }
       }
       else {
           return "Don't know how to handle that path!";
       }
    }
}

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.base(port, new Handler("./technical/"));
4 substitutions on 4 lines
```

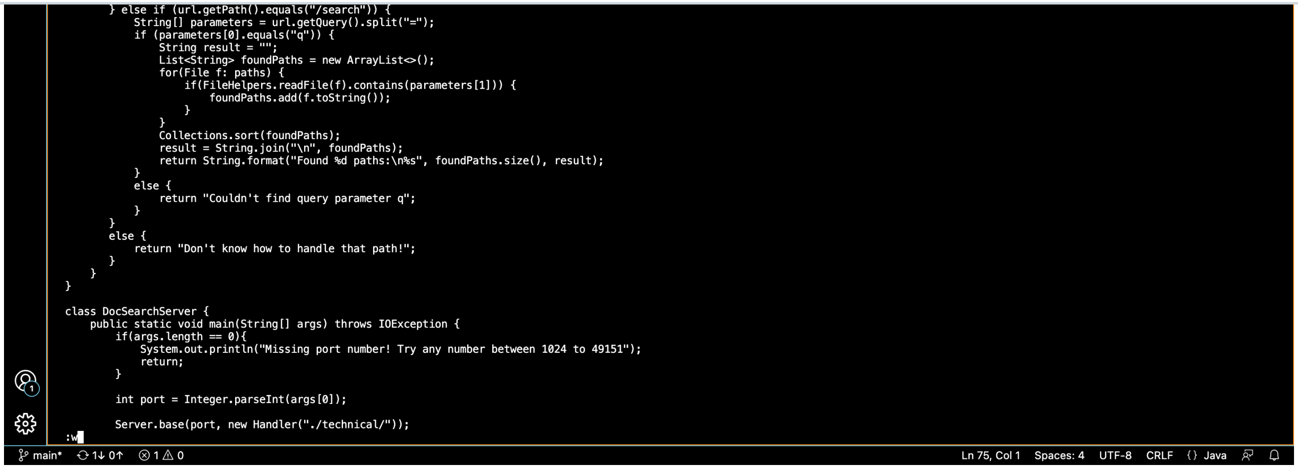
```
main*  1↓ 0↑  ⊗ 0 ⚠ 1                    Ln 14, Col 21  Spaces: 4  UTF-8  CRLF  {} Java
```

Then I typed `:w` to save the changes



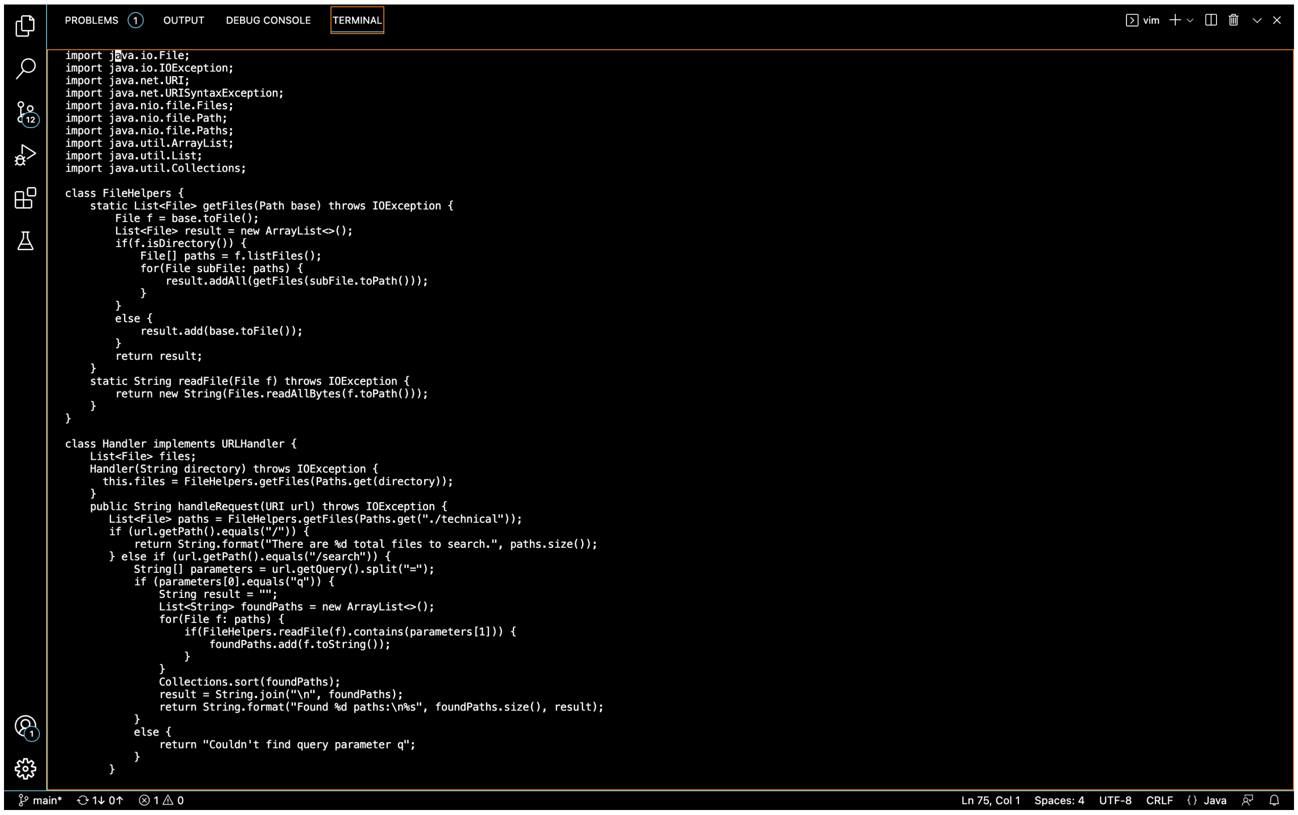
```
            File[] paths = f.listFiles();
            for(File subFile: paths) {
                result.addAll(getFiles(subFile.toPath()));
            }
        }
        else {
            result.add(base.toFile());
        }
        return result;
    }
    static String readFile(File f) throws IOException {
        return new String(Files.readAllBytes(f.toPath()));
    }
}

class Handler implements URLHandler {
```

39 lines (24 sloc) | 1.43 KB ...



```
        } else if (url.getPath().equals("/search")) {
            String[] parameters = url.getQuery().split("=");
            if (parameters[0].equals("q")) {
                String result = "";
                List<String> foundPaths = new ArrayList<>();
                for(File f: paths) {
                    if(FileHelpers.readFile(f).contains(parameters[1])) {
                        foundPaths.add(f.toString());
                    }
                }
                Collections.sort(foundPaths);
                result = String.join("\n", foundPaths);
                return String.format("Found %d paths:\n%s", foundPaths.size(), result);
            }
            else {
                return "Couldn't find query parameter q";
            }
        }
        else {
            return "Don't know how to handle that path!";
        }
    }
}

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.base(port, new Handler("./technical/"));
:w
```

Finally, I pressed `<Enter>` to save the file.



```
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 base) throws IOException {
        File f = base.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(base.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 = 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")) {
               String result = "";
               List<String> foundPaths = new ArrayList<>();
               for(File f: paths) {
                   if(FileHelpers.readFile(f).contains(parameters[1])) {
                       foundPaths.add(f.toString());
                   }
               }
               Collections.sort(foundPaths);
               result = String.join("\n", foundPaths);
               return String.format("Found %d paths:\n%s", foundPaths.size(), result);
           }
           else {
               return "Couldn't find query parameter q";
           }
       }
```

Part 2

For the first one, I spent 76 seconds fixing the test and confirming it works. The difficulty I encountered with was typing the wrong character which caused me failed at the first time. The second one only took me 45 seconds and I did it well.

Which of these two styles would you prefer using if you had to work on a program that you were running remotely, and why?

I would prefer using the second style as it costs me less time. When we use the first style for small programs, we often need more typing to solve the same problem.

What about the project or task might factor into your decision one way or another? (If nothing would affect your decision, say so and why!)

I would use the first style when I am working on a project or task which typically requires a lot of typing and editing. Working on Visual Studio Code helps me find syntax errors and helps me run the code and debug it. Vim is just a text editor and can't run the code, which means I can't get feedback until I complete writing the file.