Credit Name: File Structures and Exception Handling
Assignment Name: FindAndReplace

How has your program changed from planning to coding to now? Please explain?

```
//Declaration area
File dataFile;
FileReader in;
BufferedReader readFile;
FileWriter out;
BufferedWriter writeFile;
String fileName, searchPhrase, replacePhrase, fileContent;
```

I started out by creating the variables I needed for FindAndReplace

```
try
{


}
catch (FileNotFoundException e)
{
    System.out.println("File does not exist or could not be found.");
    System.err.println("FileNotFoundException: " + e.getMessage());


}
catch (IOException e)
{
    System.out.println("Problem with input/output");
    System.err.println("IOException: " + e.getMessage());
}
```

Next I created a try catch statement that would find the phrase and replace it.

```
Scanner input = new Scanner(System.in);

System.out.println("Enter file name to find and replace a word/phrase: ");
fileName = input.nextLine();

System.out.println("What word/phrase would you like to replace?: ");
searchPhrase = input.nextLine();


System.out.println("What would you like to replace it with?: ");
replacePhrase = input.nextLine();
```

I also added code to take in the user's input for the filename, phrase/words to replace, and what they want to replace it with.

```
try
{
    //Set up BufferedReader
    dataFile = new File(fileName);
    in = new FileReader(dataFile);
    readFile = new BufferedReader(in);


    while((fileContent += readFile.readLine()) != null)
    {
        System.out.println(fileContent);
    }
    readFile.close();
    in.close();


    System.out.println(fileContent);



}
```

I did ctrl f in the online textbook to look for ways to replace text and came across a method in the String class called replaceAll(). I was going to take all the text from the text file and store it in a string which I could use replaceAll() on, to find and replace the text. Later I could erase the text in the file and replace it with the new text.

```
try
{
    //Set up BufferedReader
    dataFile = new File(fileName);
    in = new FileReader(dataFile);
    readFile = new BufferedReader(in);


    while((curLine = readFile.readLine()) != null)
    {
        fileContent += curLine;
        System.out.println(fileContent);
    }
    readFile.close();
    in.close();


    System.out.println(fileContent);



}
```

I didn't account for fileContent adding the null to its value instead of replacing it. So I had to create a new variable that would add itself to fileContent outside of the while conditions.

```
try
{
    //Set up BufferedReader
    dataFile = new File(fileName);
    in = new FileReader(dataFile);
    readFile = new BufferedReader(in);


    //Get all the text from the file and add it to a String
    while((curLine = readFile.readLine()) != null)
    {
        fileContent += curLine;
    }
    readFile.close();
    in.close();


    System.out.println(fileContent);


    fileContent = fileContent.replaceAll(searchPhrase, replacePhrase);


    System.out.println(fileContent);



}
```

After working that out I just added 2 lines of code to find and replace the text and print it to see if it worked which it did.

```
try
{
    //Set up BufferedWriter
    dataFile = new File(fileName);
    out = new FileWriter(dataFile);
    writeFile = new BufferedWriter(out);


    //Add text back to file
    writeFile.write(fileContent);


    writeFile.close();
    out.close();
}
```

I added another try statement to write the modified text back to the file. There was an issue with the program not keeping track of new lines that needed to be added to the String though.

```
try
{
    //Set up BufferedReader
    dataFile = new File(fileName);
    in = new FileReader(dataFile);
    readFile = new BufferedReader(in);


    //Get all the text from the file and add it to a String
    while((curLine = readFile.readLine()) != null)
    {
        fileContent += curLine + "\n";
    }
    readFile.close();
    in.close();


    //Find and replace user selected phrase/word
    fileContent = fileContent.replaceAll(searchPhrase, replacePhrase);
}
```

I went back to the place where I added the lines and added a new line indicator every time a new line was added. It worked aside from how it added a new line even to the end of the text which meant that it would continuously grow larger and larger every time you used the program.

```
//Get all the text from the file and add it to a String
while((curLine = readFile.readLine()) != null)
{
    if(firstLine)
    {
        fileContent += curLine;
        firstLine = false;
    }
    else
    {
        fileContent += "\n" + curLine;
    }

}
readFile.close();
```

I decided that the easiest way to fix this issue was to move the new line to the front. To stop the program from adding a new line to the very start of the file I also added code to check if it was reading the first line or not. There wasn't a way I could think of to check if it was currently reading the last line (so that it wouldn't put a new line at the end) so I decided to go with this option.