cse15l-lab-reportswi24

Debugging, File Exploration, and Text Analysis

2/13 Lab Report 3

Part 1: Bugs

Last week, I stepped through the process of indentifying and isolating progam bugs.

For the `ArrayExamples` class, I used `JUnit` tests to check for buggy methods. Let's look at one example.

> **Buggy Method:** `reverseInPlace(int[] arr)` **Failure-Inducing Input**
>
> ```
> int[] fails = {3,2,1};
> assertArrayEquals(new int[]{1,2,3}, ArrayExamples.reverseInPlace(fails));
> ```
>
> **Asymptomatic Input**
>
> ```
> int[] passes = {3,3,3};
> assertArrayEquals(new int[]{3,3,3}, input1);
> ```
>
> **Symptom**
>
> ```
> carmenhe@Carmens-MacBook-Pro cse15l-lab3 % bash test.sh ArrayTests
> JUnit version 4.13.2
> .E
> Time: 0.006
> There was 1 failure:
> 1) testReverseInPlace(ArrayTests)
> arrays first differed at element [2]; expected:<3> but was:<1>
>         at org.junit.internal.ComparisonCriteria.arrayEquals(ComparisonCriteria.java:78)
>         at org.junit.internal.ComparisonCriteria.arrayEquals(ComparisonCriteria.java:28)
>         at org.junit.Assert.internalArrayEquals(Assert.java:534)
>         at org.junit.Assert.assertArrayEquals(Assert.java:418)
>         at org.junit.Assert.assertArrayEquals(Assert.java:429)
>         at ArrayTests.testReverseInPlace(ArrayTests.java:14)
>         ... 30 trimmed
> Caused by: java.lang.AssertionError: expected:<3> but was:<1>
>         at org.junit.Assert.fail(Assert.java:89)
>         at org.junit.Assert.failNotEquals(Assert.java:835)
>         at org.junit.Assert.assertEquals(Assert.java:120)
>         at org.junit.Assert.assertEquals(Assert.java:146)
>         at org.junit.internal.ExactComparisonCriteria.assertElementsEqual(ExactComparisonCriteria.java:8)
>         at org.junit.internal.ComparisonCriteria.arrayEquals(ComparisonCriteria.java:76)
>         ... 36 more
>
> FAILURES!!!
> Tests run: 1,  Failures: 1
> ```
>
> From the output after running my tests, it appears that the *failure-inducing input* is not reversing as expected.
>
> **The Bug**

- Before fix:

```
// Changes the input array to be in reversed order
static void reverseInPlace(int[] arr) {
    for(int i = 0; i < arr.length; i += 1) {
        arr[i] = arr[arr.length - i - 1];
    }
}
```

The current issue is that the original value at each position is updated without being retained, so when we reach the later half of the list, the values aren't being changed

- After fix:

```
// Changes the input array to be in reversed order
static void reverseInPlace(int[] arr) {
    int replaced;
    for(int i = 0; i < arr.length/2; i += 1) {
        replaced = arr[i];
        arr[i] = arr[arr.length - i - 1];
        arr[arr.length - i - 1] = replaced;
    }
}
```

To fix the bug, I created a new variable called `replaced` to store the value being replaced, so it's not being overwritten. Then I only need to iterate through half of the list so the elements in the latter half of the list are updated with the value stored in the `replaced`.

Part 2: Researching Commands

The `grep` command** Recall that at it's simplest form, the `grep` command takes a String pattern and file path and prints out all the lines in the file that contain the pattern.

`grep [options] "pattern" /file path/`

Options

1. `-r` Recursively searches through the specified directory the given String pattern. Syntax: `grep -r "pattern" /directory path/`

   For each file that has the pattern, lines that contain the matches are printed out.

```
$ grep -r "base pair" technical/plos
technical/plos/journal.pbio.0020223.txt:        Watson–Crick base pairing, the proximity of t
technical/plos/journal.pbio.0020190.txt:        sequence, which is a specific series of eight
technical/plos/journal.pbio.0020190.txt:        chromosomes, on the order of one or two thous
```

I can combine this with other commands, such as `wc`, to count the total number of matches, and specify the file type in the directory pattern. This example shows that 226 lines within the text files in technical/biomed contain a match.

```
$ grep -r "base pair" technical/biomed/*.txt | wc
226    2326   22534
```

Source: Chat_GPT - Prompt: How does grep -r work and how can it be used to explore files? - Output: Provided the user manual definition of `-r` in context of `grep`, and 6 conceptual applications of `grep -r` with syntax. I included the definitions above, translated into my own words, and created my own examples with a newfound understanding of how to apply `grep -r`.

2. `-l` : Limits the output to a list of the files in the specified file path that contain a match of the pattern. Syntax: `grep - l "pattern" /file path/`

   When I use the expansion file path, all relevant files names containing the pattern are printed.

   ```
   $ grep -l "base pair" technical/plos/*.txt
   technical/plos/journal.pbio.0020190.txt
   technical/plos/journal.pbio.0020223.txt
   ```

   When a specific file path is specified, if it contains a match, it is the only file printed.

   ```
   $ grep -l "base pair" technical/plos/journal.pbio.0020190.txt
   technical/plos/journal.pbio.0020190.txt
   ```

   Source: Chat_GPT - Prompt: How does grep -l work and how can it be used to explore files? - Output: Provided the user manual definition of `-l` in context of `grep`, and 4 conceptual applications of `grep -l` with syntax. I included the definitions above, translated into my own words, and created my own examples with a newfound understanding of how to apply `grep -l`.

3. `-c` :

4. `-f file` :