Automatic Testing Framework for Contrast Finder

CSCI 362
Team: GoFish
Members: Lexus Hartung, Levi Hagen, Steven Higgins

Introduction

Computer science may sound like a straightforward discipline, but it is truly a job of many hats. While we implicitly say we are taking part in a science through the title, we leave at the wayside all the extremely important skills that a true computer scientist must have in order to function in our field. Some of these essential abilities include, Teamwork, Time Management, and Creativity. These softer skills are often overlooked, and it is the goal of this paper to bring them to the forefront by showing examples of them in action. Throughout a semester long team project, every member of our team displayed these fine attributes consistently and it is our hope that the fruits of our labor show the importance of these skills.

Chapter 1

Project Selection

OpenMRS

At first our project is OpenMRS, a database that is meant to connect doctors all over the world. This network of professionals can hopefully help to build the infrastructure of newly developing healthcare markets and combat the serious diseases threatening these areas such as AIDS, tuberculosis, and malaria. This software runs on a combination of mySQL, Mavin, and optionally tomcat.

Experiences for OpenMRS

We started from a bit of a disadvantage as one of the members computers experiences technical difficulties and had to be sent off to be fixed. With two members trying to run the project and one forced to restart on getting Ubuntu on their computer we got behind. MySQL was being an utter nightmare. It did not want to cooperate with any of the coding elements or the web application.

Results for OpenMRS

Attempting to work with a database that is being manipulated by Mavin code and a web server that's trying to update based on the database proved to be too complex for us. MySQL really gave us the greatest troubles, taking too much space on one persons machine and just not getting along with the web server on another's. Because of our struggles with the database we will be changing projects, going instead with Contrast Finder.

Contrast Finder

After having issues getting OpenMRS we switched to Contrast Finder for our new project and it is a web service that is meant to identify the difference between different colors. This project is meant to help make websites more accessible to those with colorblindness and to help web designers help make their websites truly distinctive so that they are easier to interpret for everyone. Contrast Finder runs on a combination of Mavin and Tomcat.

Experiences

One members computer still isn't back from repair and the new replacement doesn't seem capable of running virtual machine. The computer should be back soon but we are still a bit crippled in our capacities. While one of use was able to get it running it wasn't easy going. We started out trying to run an older version, and this version was less stable. We were also having issues with the dependencies the project requires. Tomcat was being very uncooperative.

Results

Luckily we got the project to finally load into tomcat and it ran 15 automatic tests when it got itself oriented. All of the tests ran perfectly and spat out very nice little reports of their work. We will be looking more into these reports to see exactly how these tests are running and should be able to use them as examples to make our own test cases.

Chapter 2

The Testing Process

Phase one: Test the system's ability to find colors based on their names
Phase two: Test the system's ability to compare contrast between two colors
Phase three: Test whether or not the system returns the correct hue when given a color Phase four: Tests the system's ability to properly convert Hexadecimal to RGB
Phase five: Tests to see if the system accepts non-capital Hexadecimal inputs

Testing Schedule

Oct 1- Oct 8: Set up the design for the automated testing framework, including documentation to how it will work and the like
Oct 9 – Oct 16: Build the testing framework with previous design as a guide. Really just working to get it functioning
Oct 17 – 24: Test the framework to just make sure it's working as intended
Oct 25 – 28: Update the previous documentation to reflect the current implementation of the system
Oct 29: Present our test framework and explain its functionality.
Nov 4: Team meeting to make sure 10 test cases are completed and running properly
Nov 11: Team meeting to make sure all 25 test cases are complete and running properly
Nov 12: Present the running of our test cases from the test framework and explain any questions regarding the tests function.
Nov 13 – 14: Conceptualize the design of the faults for the system and document this process
Nov 15 – 17: Build the faults and inject them into the code.
Nov 18: Test and make sure the faults are working as they should. Update the necessary documentation.
Nov 19: Present the results of our fault design
Nov 21/ Nov 26: Present the overall process that got us to this point and demonstrate the fruits of our labor

Test Recording Procedures

All tests will have the following categories filled out for each of their tests:
< Test Case ID > - The ID of the specific test case
< Requirement Tested > - Specifying which of the project requirements we will be addressing
< Component tested > - The greater component being tested
< Method tested > - The specific method being tested
< Test Input Data > - The test data that will be used for the test
< Expected Result > - What you expect the test to produce
< Actual Result > - What the test actually produces

Hardware and Software Requirements

The software required for this process include a Disc image File for Ubuntu 18.04, Oracle VM VirtualBox or a comparable virtual machine, Docker, Git, Java, and the source code from the Contrast Finder Repository.
The hardware required is at least 5277 MB of memory to allocate towards the Virtual Machine.

Contraints

Constraints that may come up in the foreseeable future could include computer malfunctions for members, members getting ill, ext. These all come down to losing a member for an unspecified amount of time, which will simply lead to us redistributing the workload between two members until the third is able to rejoin the group.

Test Cases

Test Case 1
Requirement: The method takes a string that is a color name and checks to see if the color is a valid entry.
Component Being Tested: ColorNameLookup
Method: getColorNameFromStr(String colorStr) Test Inputs: Yellow
Expected Outcome: Yellow

Test Case 2

Requirement: The method takes a string that is a color and checks to see if the color is a valid entry.
Component Being Tested: ColorNameLookup
Method: getColorNameFromStr(String colorStr)
Test Inputs: YELLOW Expected Outcome: Yellow

Test Case 3

Requirement: The method takes a string that is a color and checks to see if the color is a valid entry.
Component Being Tested: ColorNameLookup
Method: getColorNameFromStr(String colorStr)
Test Inputs: yellow Expected Outcome: Yellow

Test Case 4

Requirement: The method takes a string that is the Hexadecimal value of a color and returns the RGB of the same color
Component Being Tested: ColorConverter
Method: hex2Rgb(String colorStr)
Test Inputs: 00AA00
Expected Outcome: rgb(0,170,0)

Test Case 5

Requirement: The method takes a string that is the Hexadecimal value of a color and returns the RGB of the same color
Component Being Tested: ColorConverter
Method: hex2Rgb(String colorStr)
Test Inputs: 0000aa
Expected Outcome: rgb(0,0,170)

Chapter 3

Experiences

During this portion of the project the main focus was to get the driver working for each of our test cases. This part was straightforward and just java programing and being able to analyze other people code and use it. After we got a driver working for our test case we worked on a script that would automate the testing process. This was difficult because no one in our group knew much of bash. Our script reads all the test case files and sends the data into our driver and then takes the data returned by the driver and processes it to see if the output is what is suppose to be.

How-To Documentation

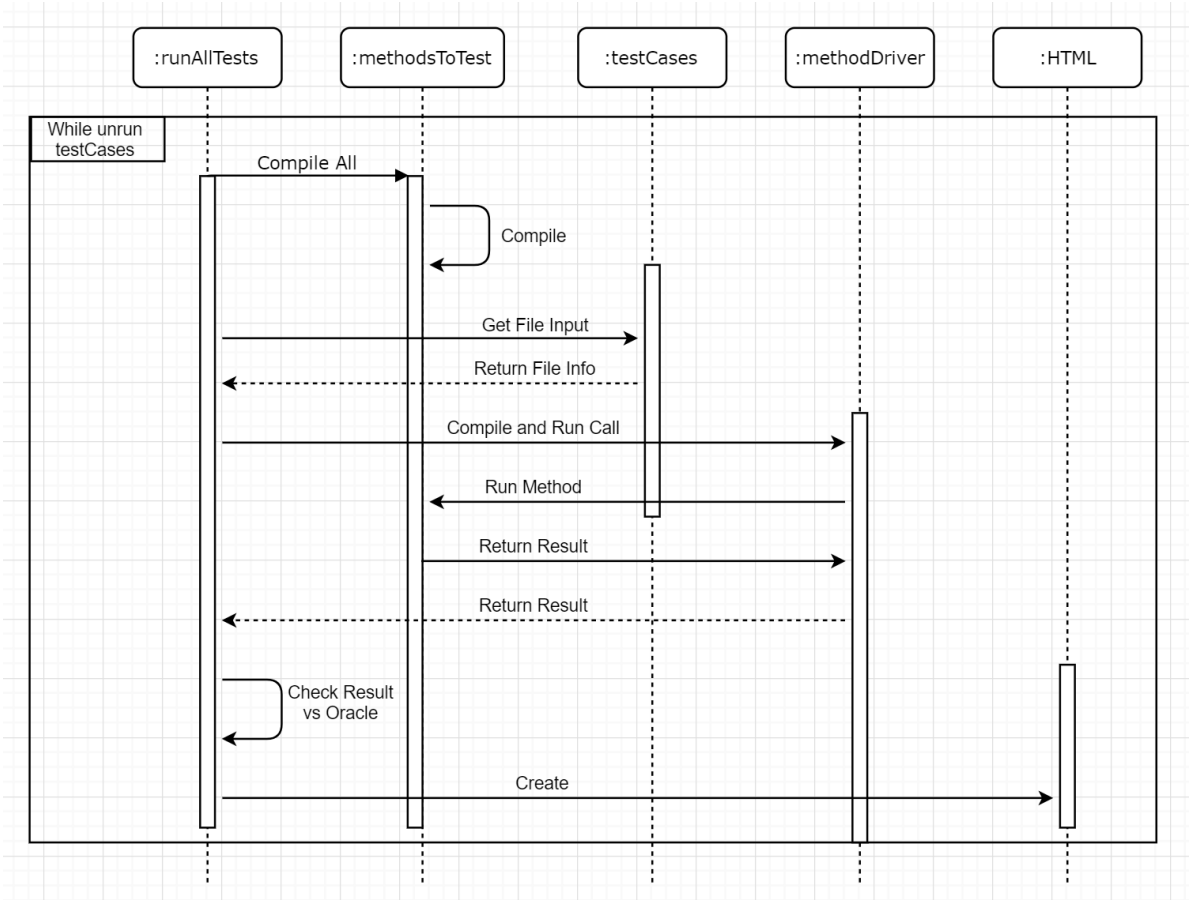
To Run Script install JDK 12
open terminal
Change Directory to ..//GoFish/TestAutomation/scripts/
run script with ./runAllTests.sh

Methods Tested

getColorNameFromStr(String colorStr) - Takes a color name and standardizes the name to lower case with the first letter capitalized.

hex2rgb(String colorStr) - Converts a Hex String into Rgb form then converts it into a Color obj.

The Architectural Description:



Chapter 4

Experiences

For this task the process was fairly straightforward we made our script more general and this made it where we only had to create three more drivers for the next 3 methods we are testing, and once the drivers were properly created the script was able to compile them and run them just like the first 2 methods.

Each of these methods has its own driver.

Some of the test cases we intentionally give it an invalid input and see how it handles it for now all we do is catch the error it throws and print “ERROR” as the actual result but this will be changed to give the specified error that is returned by the method that is being tested.

Methods Tested

getColorNameFromStr(String colorStr) - Takes a color name and standardizes the name to lower case with the first letter capitalized.

hex2rgb(String colorStr) - Converts a Hex String into Rgb form then converts it into a Color obj.

distanceColor(Color fgColor, Color bgColor) - Takes two colors and returns the distance between their Rgb values

colorFromRgbStr(String colorStr) - Takes a String containing a Rgb and creates a java color obj.

rgb2Hsl(Color color) - Takes a Java color obj and converts to HSL(Hue, Saturation, Lighting)

Framework

In our testing framework we use a shell script that will compile all the java files that have the methods that we are testing and also the driver files that we created to run each of the methods that we are testing. Our framework test 5 different methods and totals 25 test cases.

Chapter 5

Experiences

For this task we injected 5 faults into our methods that we are testing. We put three faults in ColorCoverter.java and one fault in to ColorNameLookup.java and ContrastChecker.java. The faults below do not make all test fail. They cause six test cases to fail, Test Cases 01, 11, 14, 15, 21, and 25. Injecting these faults into the code showed us that just little discrepancies and errors in the logic of the code can cause issues with the functionality of the program.

Faults Added

ColorConverter.java Faults

```
1. //FAULT ADDED - Changed max angle to 180 private static final int
MAX_ANGLE = 180;
2. //FAULT ADDED- Changed the max letter to E
   private static final String HEXADECIMAL_DICTIONNARY = "[0-9A-Ea-
   f]+"; // FFF, FFFFFF
3. //FAULT ADDED- Removed toLower Case command from string str
   public static Color colorFromRgbStr(String colorStr) {
   Color color = null;
   String str = colorStr.replaceAll("\\s", ""); // replace ' ', \t,
   \n, ...
   if (str.matches(RGB_DICTIONNARY) |
   str.matches(SHORT_RGB_DICTIONNARY)){ // ex: rgb(255,255,255) or
255,255,255
   str = str.replaceAll("rgb\\(", ""); str = str.replaceAll("\\)",
""); String[] strList = str.split(",");
   int r = int g = int b = if(
   &&
   Integer.parseInt(strList[0]); Integer.parseInt(strList[1]);
Integer.parseInt(strList[2]);
   r <= RGB_MAX && g <= RGB_MAX && b <= RGB_MAX
   r >= RGB_MIN && g >= RGB_MIN && b >= RGB_MIN){
   color = new Color(r, g, b); }
}
```

```
   return color;
    }
```

ColorNameLookUp.java Faults

```
1. //FAULT ADDED- Yellow changed to yellow
private ArrayList<ColorName> initColorList() { ... colors.add(new
ColorName("yellow", 0xFF, 0xFF, 0x00)); ...}
```

ContrastChecker.java Faults

```
1. //FAULT ADDED- the green value is being cubed rather than squared
public static double distanceColor(final Color fgColor, final Color
bgColor) {
int redFg = fgColor.getRed();
int redBg = bgColor.getRed();
int greenBg = bgColor.getGreen();
int greenFg = fgColor.getGreen();
int blueFg = fgColor.getBlue();
int blueBg = bgColor.getBlue();
return (Math.sqrt(Math.pow(redFg - redBg, 2) +
Math.pow(greenFg - greenBg, 3) + Math.pow(blueFg - blueBg, 2))); }
```

Chapter 6

Throughout the semester we worked on a framework for testing methods in Contrast Finder. In this project we learn many different things, one of which was that if you need help ask before your problem gets out of hand so that you can stay on track. This project taught us how to plan and work as a group to accomplish the task. We had to learn quickly how to assign tasks to group member in a way that tasks were completed on time and correctly.

Self Evaluation

We all feel like we all were able to work well together and delegate different task to each group member and get each milestone of the project done in an efficient way. At the beginning we all had disagreements on how to do certain coding tasks for the project but were able to come together and compromise on what we believed would be the best way to achieve the task at hand. All and all this experience was valuable to experience and during it we learned a lot about project management and teamwork.

Suggestions

The main suggestion that we had for this project was that the deliverables have more details about what we should have in them.