Control Flow Testing

#### 

#### 

Team 16:

Raghav Gupta

Chelsea Jaculina

Bala Nyan Kwaw

Mike Lee

Yihua Li

**February 25, 2018**

**Professor Badari**

**CMPE 187**

# Table of Contents

[**Table of Contents**](#_fvgztqx5fz61) **1**

[**Overview**](#_8pvx7xuczgkg) **2**

[Introduction](#_rfoa3jdhntvi)

[Purpose](#_gxjj4t1fsork)

[**Test Oracle**](#_mcynqj6harbr) **3**

[Environment](#_hxp4hhr1f9w7)

[S](#_cpfwcrahhykn)trategy

[**Test Suite**](#_ofs5nz8bmdo6) **6**

Inputs

Outputs

Test Cases

Statement, Path, & Branch Coverages

[**Control Flow**](#_qwje6aoaym93) **Diagrams 8**

Progress Bar

Tower Of Hanoi

[**Lessons Learned**](#_996e3tixdkie) **11**

**Source Code 11**

## 

# Overview

## Introduction

The homework was tasked in two small projects:

**Progress Bar:** Shows its progress as a sector of a circle. The screen is visualized as a square on the X, Y plane with its bottom-left corner at (0, 0), and its upper-right corner at (100, 100). Every point on the screen is either red or blue. Initially, the progress is 0%, and all points on the screen are blue. When the progress percentage, P, is greater than 0%, a sector of angle (P% \* 360) degrees is colored red, anchored by the line segment from the center of the square to the center of the top side, and proceeding clockwise.

**Tower of Hanoi:** A game where there are three slots and N number of disks placed one over the other in decreasing size. The objective of this game is to move the disks one by one from the first slot to the last slot. However, there is only one condition; we can not place a bigger disk on top of a smaller disk.

## Purpose

The purpose of this assignment is to use the concept of control flow testing to test the flow of data in the programs: the Progress Bar and the Tower of Hanoi. Our goal of using control flow testing is to check if the source code detects any unexpected flow of data in the programs. That is, when our source code is executed, the program should follow the conditional flow as expected from the algorithm and end at the expected return output.

# Test Oracle

## Environment

The hardware used for programming and testing is a Macbook Pro 13.3 from 2015, with a 2.7GHz Intel Core I5 processor, 8GB 1867MHz DDR4 of memory.

The software used for programming and testing is OS X High Sierra version 10.13.3. The program is written and tested in Java code using the Eclipse version Oxygen, with Java version 8 build 1.8.0\_144. The tests are run in Eclipse IDE.

The program used for creating the Control flow diagrams was the lucidchart.com (a plugin by Google) and Code2flow ( <https://code2flow.com/app> ) for verification of the diagram through a third party software.

## Strategy

The strategies that we used to both test the Progress Bar and Tower of Hanoi program is first to write a test oracle to determine that our program is executing the way that we predicted it to. Then, we need to design control flow diagrams so that we can develop code. Next, we need to check that are code are covering each statement, path, and branch coverage. Once we are sure that we covered everything, we will test the program to validate that all of our test cases work.

# Test Suite

A test suite is a set of tests that tend be in done in units. Each test suite helps validate if the program is working the way that it is expected to. In **Table 1**, shows a table consisting of our test cases that our team has written for our Progress Bar program.

## 3.1 Inputs

### Progress Bar

* The first input value that the user will enter is for the progress bar percentage. This integer value will be between 0 and 100 inclusive.
* The second input value that the user will enter is for the X-coordinate on the axis. This integer value will be tested and has to be between 0 and 100 inclusive.
* The third input value that the user will enter is for the Y-coordinate on the axis. This integer value will be tested and has to be between 0 and 100 inclusive.

### Tower of Hanoi

* The input value that the user will enter is the number of disks we want to move.

## 3.2 Outputs

### Progress Bar

* The output will depend on the progress bar percentage, X-coordinate, and Y-coordinate. It will given a message saying it “RED” or “BLUE.”

### Tower of Hanoi

* The output will consist of the algorithm that the disk should be moved in. It will give String that has the disk’s origin to the disk’s destination.

## 3.3 Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Purpose | Input | Expected Output | Actual Output | Pass/Fail | Execution Path ***(Check with Figure 1 in 4.2)*** |
| 1 | Check for invalid value of progress | -20, 20, 20 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(F)-4(T) |
| 2 | Check for invalid value of x-axis coordinate | 19, 222, 23 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(F)-4(F)-5-  6(T) |
| 3 | Check for invalid value of y-axis | 50, 12, -122 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(F)-8-9(F)-  10(T) |
| 4 | Check for the case that the point is inside of the progress bar | 100, 50, 50 | RED | RED | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(F)-8-9(F)-  10(F)-11-12-13(F)-14-15-16-17-  18(T)-20 |
| 5 | Check for the case that the point is outside of the progress bar | 1,2,3 | BLUE | BLUE | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(F)-8-9(F)-  10(F)-11-12-13(F)-14-15-16-17-  18(F)-19 |

**Table 1:** Progress Bar Test Cases for Complete Statement and Branch Coverage

### **Coverages**

A statement coverage means that every statement is executed.

* **Statement coverage** = = 100%, the first five test cases did cover all the statements of the program.

A branch coverage means that all outcomes of each condition is executed at least once.

* **Branch coverage** = = = 68.75%

A path coverage means that every path is executed at least once.

* **Path coverage** = = = 2%

## 3.4 Additional Test Cases

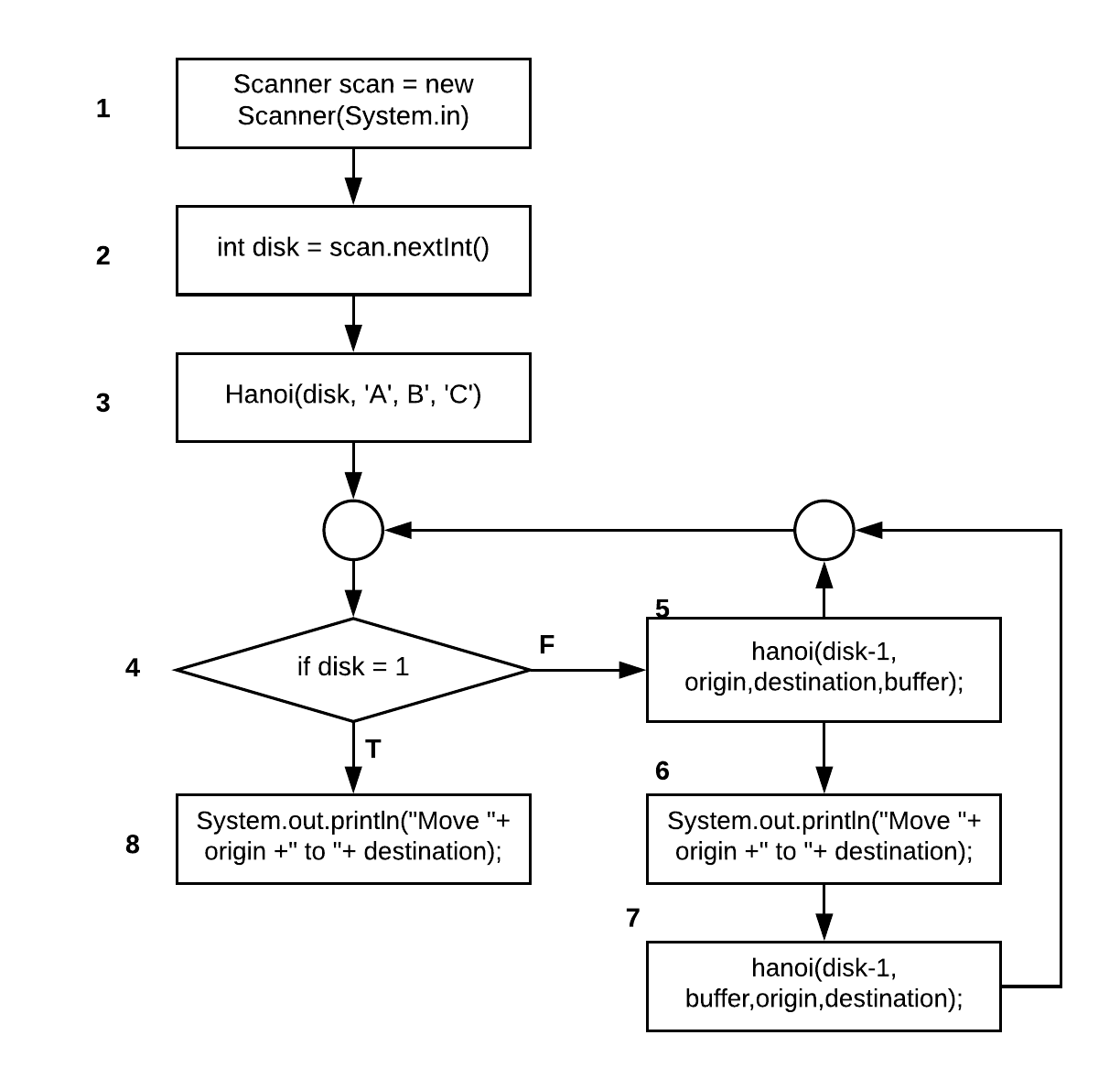
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Purpose | Input | Expected Output | Actual Output | Pass/Fail | Execution Path |
| 6 | Check if value of progress is greater than 100 | 191, 20, 20 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(T) |
| 7 | Check if value of x-axis is less than 0 | 20, -21, 33 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(T) |
| 8 | Check if value of y-axis is greater than 100 | 11, 22, 333 | Invalid value, please re-enter | Invalid value, please re-enter | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(F)-8-9(T) |
| 9 | Check the case when the point is outside of the circle | 50, 100, 2 | BLUE | BLUE | Pass | 1-2-3(F)-4(F)-5-  6(F)-7(F)-8-9(F)-10(F)-11-12-13(T)-19 |

**Table 2:** Additional Test cases for the Progress Bar

**Branch Coverage** after adding additional test case = = 100%, combining the original test cases and the additional test cases, the branch coverage is increased to 100%. We checked for validity of the three inputs. We also checked all cases that the point lies on: in the progress bar, outside of the progress bar but inside the circle, or outside of the circle but inside the screen.

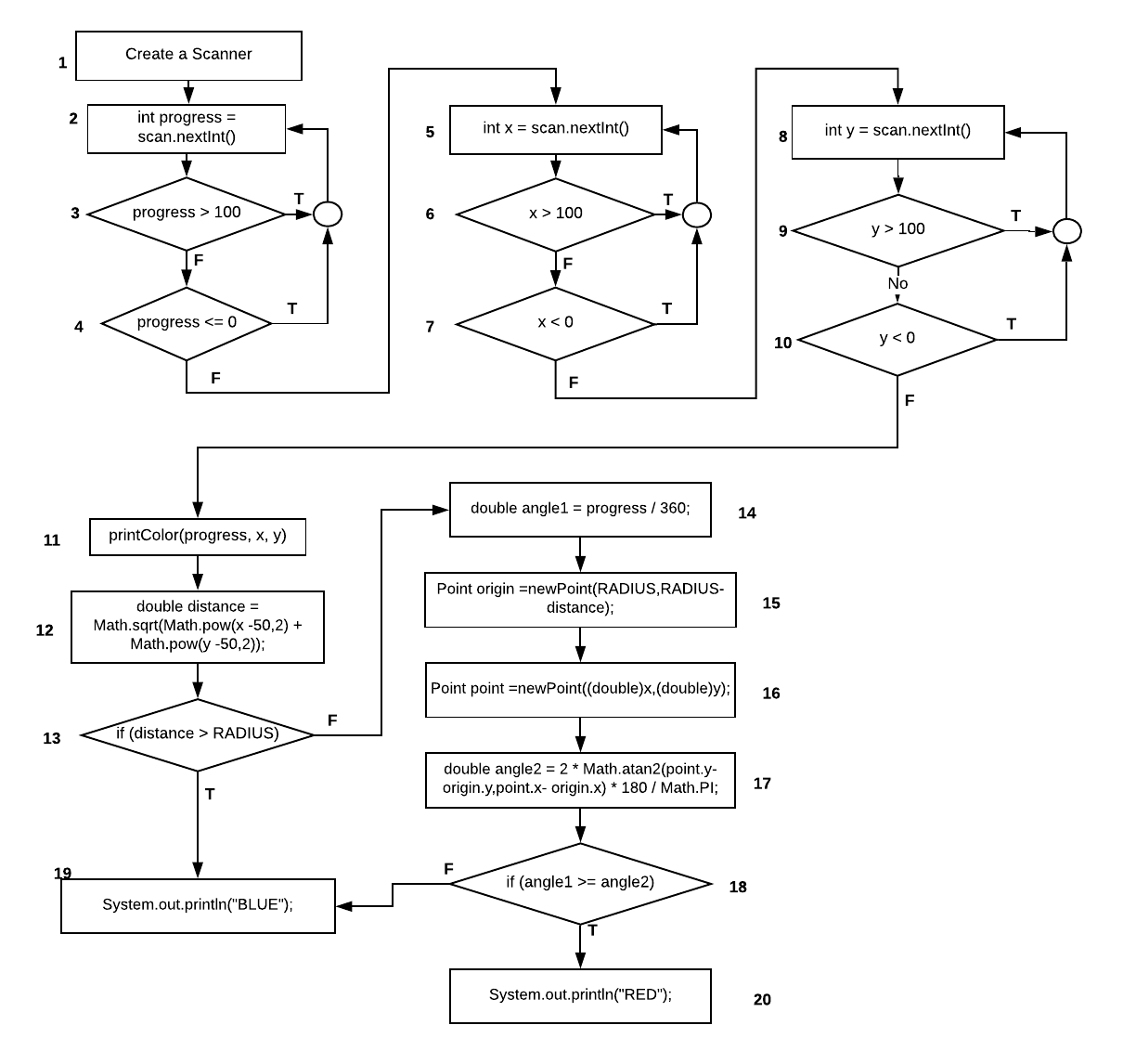
# 4. Control Flow Diagrams

## 4.1 Tower of Hanoi



**Figure 1:** Tower of Hanoi Control Flow Diagram

## 4.2 Progress Bar



**Figure 2:** Progress Bar Control Flow Diagram

# 6. Lessons Learned

The lessons that our team learned in this project is how to write a code for a Progress Bar, Tower of Hanoi, and write an oracle for our test cases. However, the main thing that we learned is the concept of Control Flow Testing (CFD). We learned how to draw control flow diagrams using LucidChart and Code2flow software. The diagrams were based on the test cases that we developed for the Progress Bar program. Also, we also learned how to determine the percentage for each statement, path, and branch coverage that was achieved.

# 

# 

# 

# 

# 7. Source Code

