# Software Engineering: Lamp

Jason Adler, Shefali Emmanuel, & Dwayne Ferguson

HFOSS

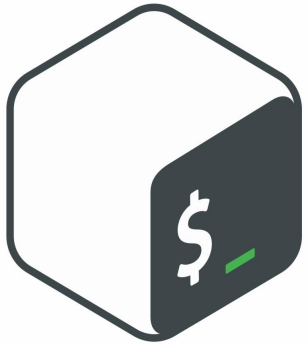
+

Language

Glucosio

# Tools

# Languages



# Requirements

- JDK 11.0.5
- Gradle 4.4.1
- Java 1.8.0_222
- Android SDK 27
- VirtualBox 6.0.10
- Android Studio 3.5
- Linux Ubuntu (64-bit) 18.4

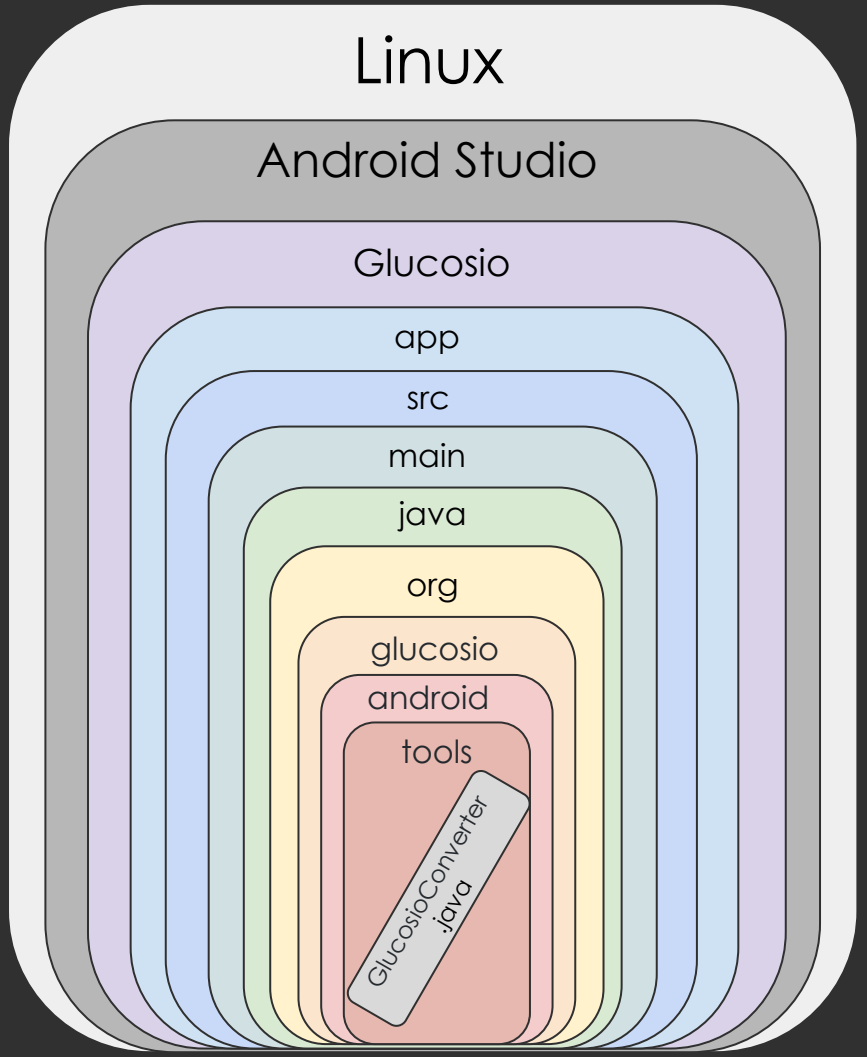
# Testing Schedule

- Sprint 1
  - Build project without modifications
- Sprint 2
  - Produce Test Plan + 12 methods to test
- Sprint 3
  - Design & Build Driver
  - Architecture & Formal Documentation
- Sprint 4
  - Create 25 test cases
- Sprint 5
  - Design & Inject 5 faults into the code
- Sprint 6
  - Finalize Driver, Presentation, Project, Paper

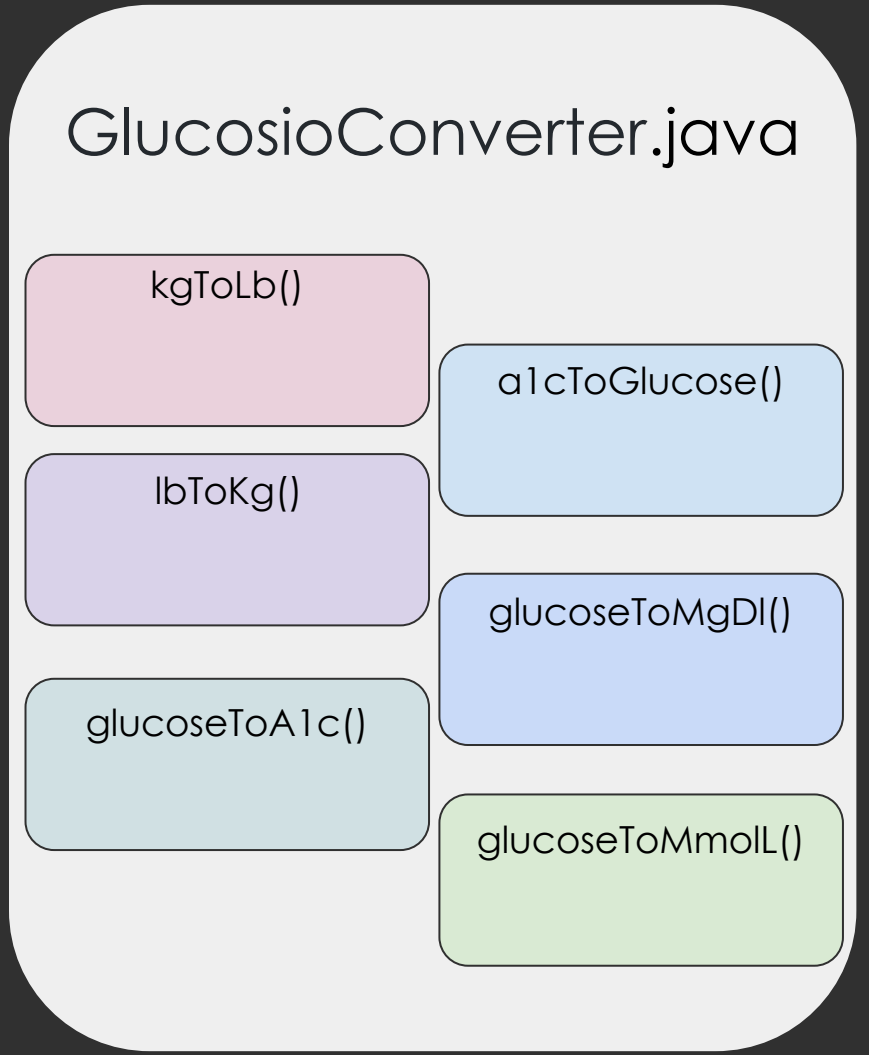
# Build Project

- Command Line
  - Run Script
- Android Studio
  - *Helped identify initial build issues.

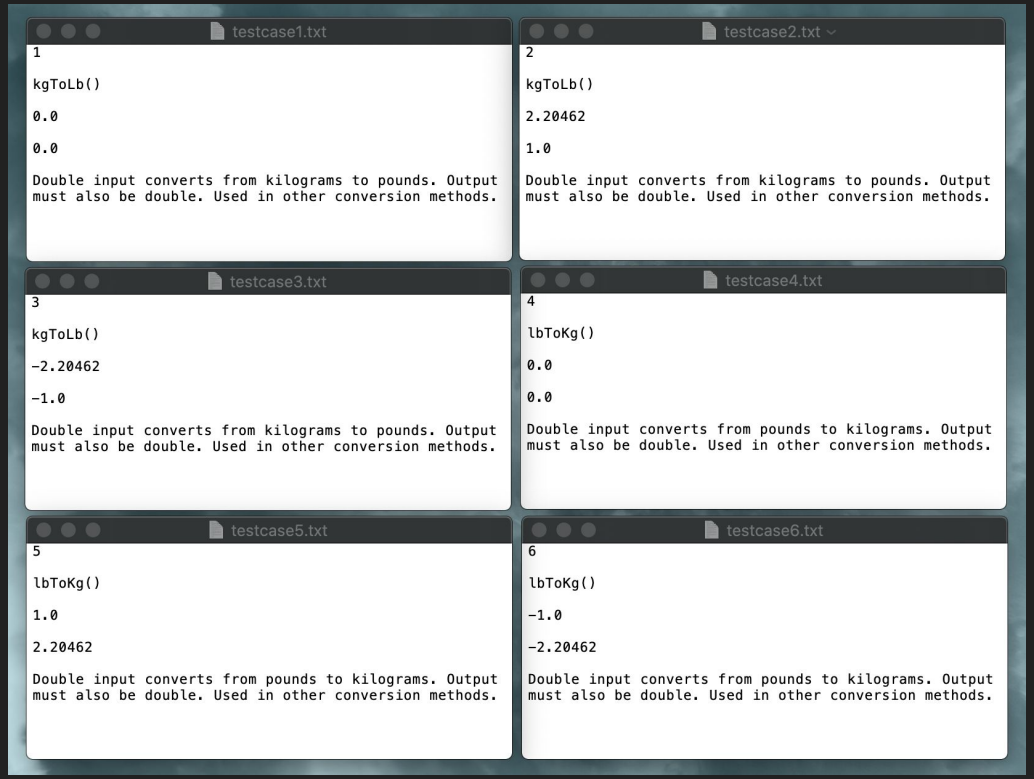
# Java File Tested



# Methods Tested



# Test Cases



testcase1.txt

```
1

kgToLb()

0.0

0.0

Double input converts from kilograms to pounds. Output
must also be double. Used in other conversion methods.
```

testcase2.txt

```
2

kgToLb()

2.20462

1.0

Double input converts from kilograms to pounds. Output
must also be double. Used in other conversion methods.
```

testcase3.txt

```
3

kgToLb()

-2.20462

-1.0

Double input converts from kilograms to pounds. Output
must also be double. Used in other conversion methods.
```

testcase4.txt

```
4

lbToKg()

0.0

0.0

Double input converts from pounds to kilograms. Output
must also be double. Used in other conversion methods.
```

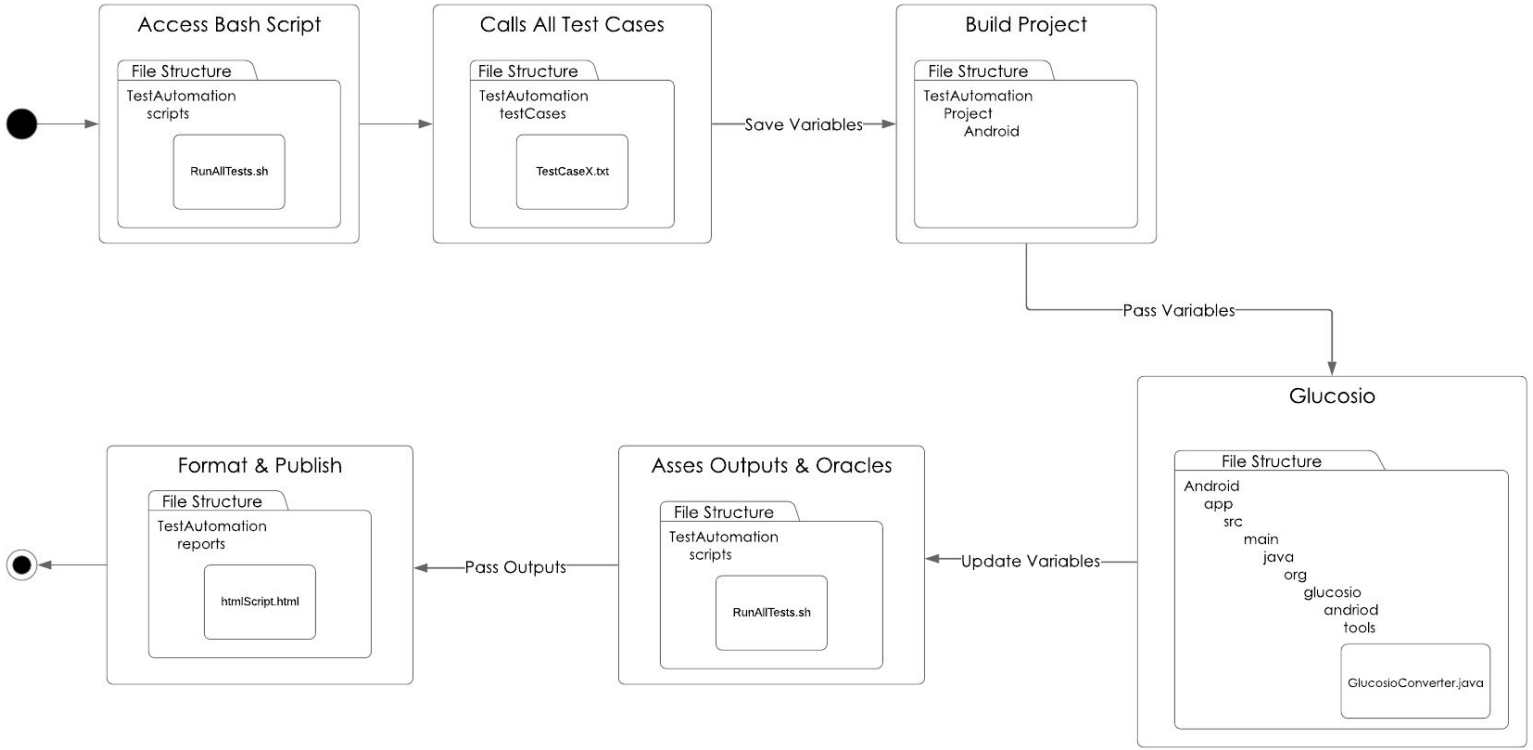
testcase5.txt

```
5

lbToKg()

1.0

2.20462

Double input converts from pounds to kilograms. Output
must also be double. Used in other conversion methods.
```

testcase6.txt

```
6

lbToKg()

-1.0

-2.20462

Double input converts from pounds to kilograms. Output
must also be double. Used in other conversion methods.
```

# Architecture



# Demo

# Faults Injected

- 5 Changes to 4 Methods
- Mutations
- Pre-Change Results
- Post-Change Results

- Example

Original

- return round((a1c * 28.7) - 46.7, 2);

Mutated

- return round((a1c * 28.7) **+** 46.7, 2);

# Fault Results

| Test Number | Method Name | Requirement | Input Value | Output Value | Oracle | Result |
|---|---|---|---|---|---|---|
| 1 | kgToLb() | Double input converts from kilograms to pounds. Output must also be double. Used in other conversion methods. | 0.0 | 0.0 | 0.0 | PASSED |
| 2 | kgToLb() | Double input converts from kilograms to pounds. Output must also be double. Used in other conversion methods. | 1.0 | 2.2 | 2.20462 | FAILED |
| 3 | kgToLb() | Double input converts from kilograms to pounds. Output must also be double. Used in other conversion methods. | -1.0 | -2.2 | -2.20462 | FAILED |
| 4 | lbToKg() | Double input converts from pounds to kilograms. Output must also be double. Used in other conversion methods. | 0.0 | 0.0 | 0.0 | PASSED |
| 5 | lbToKg() | Double input converts from pounds to kilograms. Output must also be double. Used in other conversion methods. | 2.20462 | 1.0020999999999998 | 1.0 | FAILED |
| 6 | lbToKg() | Double input converts from pounds to kilograms. Output must also be double. Used in other conversion methods. | -2.20462 | -1.0020999999999998 | -1.0 | FAILED |
| 7 | lbToKg() | Double input converts from pounds to kilograms. Output must also be double. Used in other conversion methods. | 2.20462 | 1.0020999999999998 | 1.0 | FAILED |
| 8 | lbToKg() | Double input converts from pounds to kilograms. Output must also be double. Used in other conversion methods. | -2.20462 | -1.0020999999999998 | -1.0 | FAILED |
| 9 | a1cToGlucose() | Converts from A1C to Glucose. A1C Input, Glucose Output (Double): A1C X 28.7 - 46.7. | 0.0 | 46.7 | -46.7 | FAILED |
| 10 | a1cToGlucose() | Converts from A1C to Glucose. A1C Input, Glucose Output (Double): A1C X 28.7 - 46.7. | 10000000.0 | 287000046.7 | 286999953.3 | FAILED |
| 11 | a1cToGlucose() | Converts from A1C to Glucose. A1C Input, Glucose Output (Double): A1C X 28.7 - 46.7. | 1.0 | 75.4 | -18.0 | FAILED |
| 12 | a1cToGlucose() | Converts from A1C to Glucose. A1C Input, Glucose Output (Double): A1C X 28.7 - 46.7. | -1.0 | 18.0 | -75.4 | FAILED |
| 13 | a1cToGlucose() | Converts from A1C to Glucose. A1C Input, Glucose Output (Double): A1C X 28.7 - 46.7. | 10.0 | 333.7 | 240.3 | FAILED |
| 14 | glucoseToA1C() | Converts from Glucose to A1C. Glucose Input, A1C Output (Double): AVG glucose + 46.7 / 28.7. | 0.0 | 1.5574912891986064 | 1.63 | FAILED |
| 15 | glucoseToA1C() | Converts from Glucose to A1C. Glucose Input, A1C Output (Double): AVG glucose + 46.7 / 28.7. | 1.0 | 1.5923344947735194 | 1.66 | FAILED |
| 16 | glucoseToA1C() | Converts from Glucose to A1C. Glucose Input, A1C Output (Double): AVG glucose + 46.7 / 28.7. | -1.0 | 1.5226480836236935 | 1.59 | FAILED |
| 17 | glucoseToA1C() | Converts from Glucose to A1C. Glucose Input, A1C Output (Double): AVG glucose + 46.7 / 28.7. | 10.0 | 1.9059233449477353 | 1.98 | FAILED |
| 18 | glucoseToMmolL() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Millimoles per liter. | 0.0 | 0.0 | 0.0 | PASSED |
| 19 | glucoseToMmolL() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Millimoles per liter. | 100.0 | 6.0 | 5.56 | FAILED |
| 20 | glucoseToMmolL() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Millimoles per liter. | -100.0 | -6.0 | -5.56 | FAILED |
| 21 | glucoseToMgDl() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Milligrams per 100 millilitres. | 10.0 | 180.0 | 180.0 | PASSED |
| 22 | glucoseToMgDl() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Milligrams per 100 millilitres. | 0.0 | 0.0 | 0.0 | PASSED |
| 23 | glucoseToMgDl() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Milligrams per 100 millilitres. | 100.0 | 1800.0 | 1800.0 | PASSED |
| 24 | glucoseToMgDl() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Milligrams per 100 millilitres. | -100.0 | -1800.0 | -1800.0 | PASSED |
| 25 | glucoseToMgDl() | Calculate concentration of glucose. Input(Double): glucose, Output(Double): Milligrams per 100 millilitres. | 1000.0 | 18000.0 | 18000.0 | PASSED |

# Problems Encountered

- Building Project - Version Requirements
  - Android Studio
  - Gradle
  - Java
  - VirtualBox
- Unable to create an Android Virtual Machine on VirtualBox/Ubuntu.
- the learning curve with android development
- Hard disk allocated space on VirtualBox

# Hindsight

- No Previous Exposure - Pros and Cons
  - Gradle
  - Android Studio
  - GitHub
  - Automated testing framework
  - Linux OS
  - Command line
- Format of the project

# Lessons Learned

- Communication
  - Slack
  - GitHub
  - Consistent Meetings and Conversation
- Roles
  - Balance
- Resources
  - Google
  - School vs. Industry
- Time Management
  - Agile development

# An Automated Testing Framework: *HFOSS Glucosio*

**Jason Adler, Shefali Emmanuel, & Dwayne Ferguson**
College of Charleston, Department of Computer Science

## Abstract

*Glucosio* is a free and open-source software intended to aid in both type 1 and type 2 diabetes management. It consists of a Research API, Android app and iOS app. Our project's aim is to design and build an automated testing framework to evaluate the functionality of *Glucosio*.

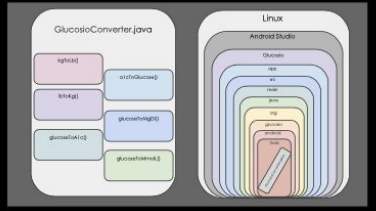
## Background

Diabetes management has evolved greatly in modern history. Changes in glucose management, types of insulin, and the administration of the insulin has offered significant improvement to the lives of those living with diabetes. *Glucosio* is an example of those improvements in the field of diabetes management. It has both Android and iOS functionality and is designed to track blood glucose test results.

## Testing Schedule

1. Evaluate and Build current project without modifications
2. Produce Test Plan including at least 5 test cases that will be developed
3. Design & Build an automated testing framework
4. Test Project by utilizing a driver and 25 test cases
5. Design and inject 5 faults into the code that will cause at least 5 tests to fail, but not all tests to fail. Exercise framework and analyze the results.
6. Present Final Project with 30 test cases

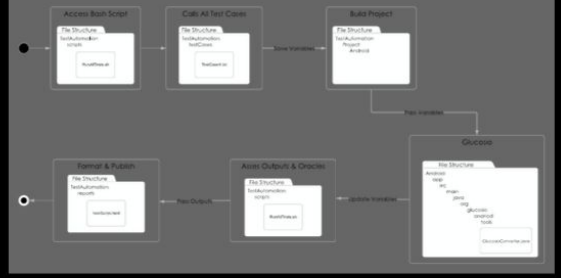
## Java File and Methods Tested




## The Framework

The automated framework involves a bash script and java drivers which correspond to specific test cases. The team has developed 25 test cases from 6 methods. The test cases are saved as .txt files which only contain the necessary information detailed in the "Test Case Template." Depending on the method being tested, a corresponding driver will need to be used to instantiate an object of the class and pass the proper values to the tested method.

User runs 'runAllTests.sh' in their command line once the project has been cloned to user's computer. The script will begin a loop to access each of the '.txt' files. It will save the test case information as variables, run the appropriate driver with the test case information, and then output "Test # Complete" in the command line. The output values are compared to the expected output from the .txt file and assigned as 'Passed' or 'Failed'. Upon completion of the loop, and all test case .txt files being processed, the script will open the users browser and display the .html file that has been generated with the results. A table has been formatted to display all pertinent information, including the date and time the report was generated, as well as test number, method name, requirement, input value, output value, oracle, and result.




## Test Cases Examples

## Test Case Template

- The team has written a script to execute the methods with test values. We chose five methods to test five different times and the results are outputted to an .html file opened in the user's web browser.
- The test cases contain:

1. Test number or ID
2. Method being tested
3. Expected outcome(s)
4. Test input(s) including command line argument(s)
5. Requirements

## Problems Encountered

- Building Project - Version Requirements
  - Android Studio
  - Gradle
  - Java
  - VirtualBox
- Unable to create an Android Virtual Machine on VirtualBox/Ubuntu.
- The learning curve with android development
- Hard disk allocated space on VirtualBox

## Run Test Cases

1. Clone project to computer
2. Open terminal
3. Navigate to Lamp-master/TestAutomation/
4. Use command ./scripts/runAllTests.sh
5. Testing should complete and output to html file in browser

## Tools

Android Studio · Gradle · GitLab · GitHub

# References


- http://logo-load.com/4445-virtualbox-logo.html
- https://commons.wikimedia.org/wiki/File:Breezeicons-apps-48-android-studio.svg
- https://pngio.com/images/png-614970.html
- https://www.pngtube.com/viewm/hmwRxJi_bash-logo-shell-script-logo/
- https://www.stickpng.com/img/icons-logos-emojis/tech-companies/java-logo
- https://about.gitlab.com/press/press-kit/
- https://www.stickpng.com/img/icons-logos-emojis/tech-companies/gradle-logo