Assignment HW2

Cover Page

Prepared for:
Dr. Mehra Borazjany
TA is TBD

Prepared by:
Alex Lundin
Ali Haider
SE-4367.0U1-Testing

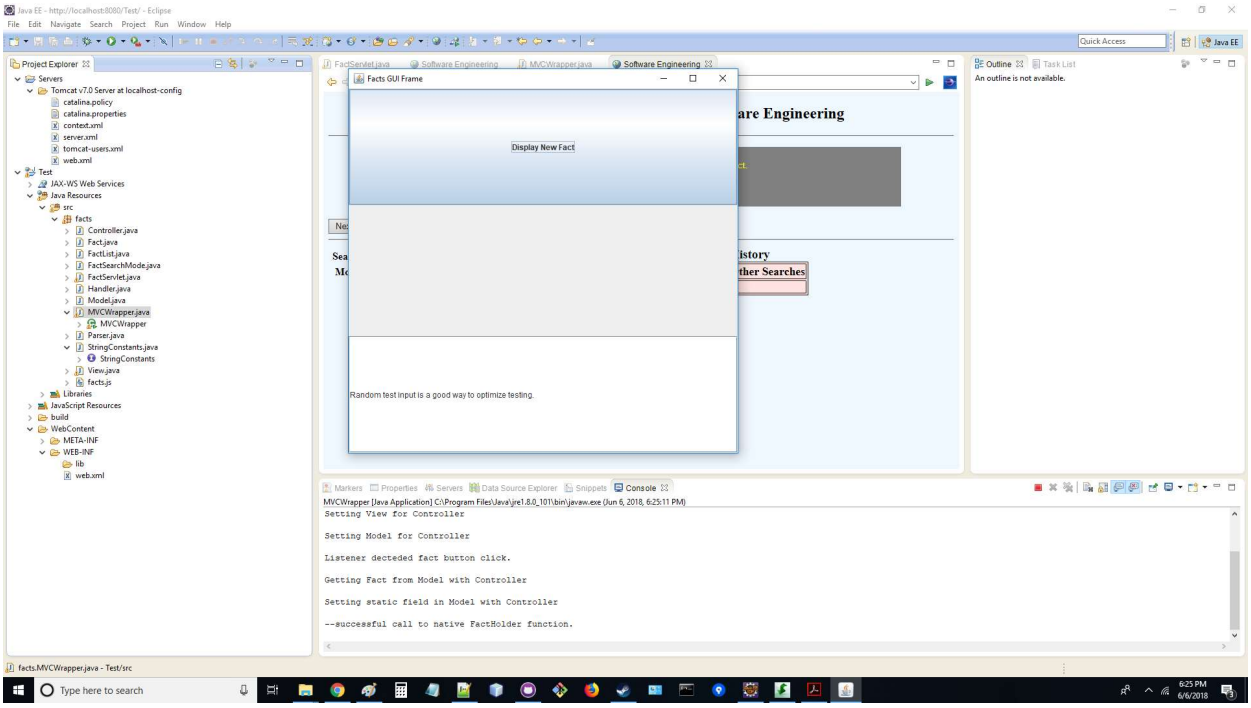
Assignment Choice:
Simple desktop computer GUI

06 June 6, 2018

Proof Of Working Software

Github Link

https://github.com/AlexLundinEducational/SE-4367-Testing



Phase 1 Development

https://knowledge.autodesk.com/community/screencast/9ab30326-5031-45e0-b856-5ff206242024

Phase 2 Development

https://knowledge.autodesk.com/community/screencast/18e387d4-7c61-416a-96d9-26ed922c8539

Phase 3 Development

https://knowledge.autodesk.com/community/screencast/090da05a-5702-41cd-a5ea-1dca3a2862bc

Successful Use Of an active GUI

https://autode.sk/2Je6wml

Proof of Facts Web App Use

https://knowledge.autodesk.com/community/screencast/8d902e29-efae-4a32-a043-e8892cd269b2

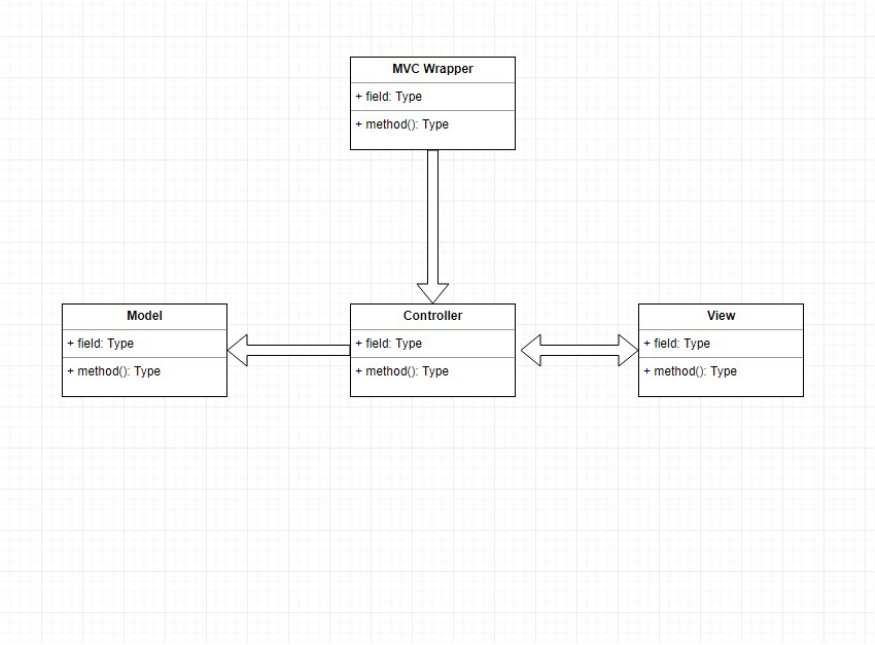
Proof of Maintainability Assessment and Fully completed GUI

https://autode.sk/2HoonRK

Documentation Log

1. Reductions/removals/modifications
    a. JavaServlet.java
        - Refactored the hard-coded string paths into their own file
        - this makes the files easier to maintain

2. New Files
    a. StringConstants.java
        - this is a refactoring of the strings used in the JavaServlet.java file
    b. MVCWrapper
        - Holds the main routine
        - Creates the Jframe GUI
        - This class DOES NOT instantiate as an object
        - This allows MVCWrapper to implement (abstractly extend) the ActionListener interface
        - Keeping this class static is a concept is the key to the whole design pattern
        - MVCWrapper's main purpose is to listen to the user clicks
    c. Controller
        - Communication between the model and view
    d. Model
        - Holds data
    e. View
        - Displays data in the Jframe GUI
        - This class requests data from the Controller
        - The Controller physically controls the data flow between components



Maintainability Assessment

IEEE defines software maintenance as "The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment." Our group decided to implement a GUI interface on our server. We obtained the original servlet source code from the facts repository and modified the FactServlet.java source file.

What did the original programmers do that made it hard to change the software?

What made the code difficult to modify was the lack of comments in the original source files. If someone reading the source code is not familiar with web-based java applications, understanding each method can be difficult. As the programmers, we would include a detailed description showing input and output values of each of the functions in the source file.

The overall lack of documentation made the reading demand much higher. We had to read each file to learn how to use the methods that we already in place. Any baseline documentation would have helped. A full UML diagram would have been best. But even a small sequence diagram would speed maintenance up.

What did the original programmers do that made it easy to change the software?

The code was easy to change given the organization of the original source files. Each function followed a universal naming convention.

The string paths were actually very simple to find, they were at the top of the file and they were collected together. So that made it easy to UNDERSTAND HOW to change the string paths if needed. It did not make the paths themselves easy to change because this required immersing ourselves into the project to actually find them.

What would you do differently if you did it again?

We would keep all string constants outside of the designs and in their own file, which we did. This makes updating build paths a breeze because when a programmer wants to tailor to a new location, they don’t even have to open any of the design files to make the project run. This is a design refactoring guideline we learned in Software Project Management. It’s extremely important on large teams and big projects. Building Software must not consume developer or testers time in those large environments.

Ali Haider
Alexander Lundin

Homework2 **SE 4367: Software Testing, Verification, Validation and QA**

PLEASE PRINT ALL PARTICIPANT NAMES ON TOP OF THE PAGE

Class, Section and Group# SE-4367 Testing, Section 0U1 Group 1

Total Points (Out of 100 points) _____

Instruction:

1. Answer to the problem on a **PDF** file **(PDF file only)** and save it as HW#_YOUR-TEAM#.PDF
2. Submit the **PDF** file to eLearning before the due date

Last week you downloaded the small web application. Your job this week is to modify the given server to have a different interface. You can change it to run in one of two ways:

1. As a stand-alone command line interface on your computer
2. As a simple GUI on your computer (not a web application)

As you make the changes, **keep a simple documentation log** of what you do. Note which components and methods you change and which ones you no longer use. Summarize the changes in a few words. You do not need to go to the level of which variables you create or delete or how you change the control flow. I just want the highlights. This should be about one page.

Next, write a short assessment of the maintainability of the software. What did the original programmers do that made it **hard to change** the software? What did the original programmers do that made it **easy to change** the software? What would you do differently if you did it again?

Submission

Submit four items in a PDF file:

1. A very simple cover page with all partner names and contributions, what kind of interface you created (CL, GUI, or mobile), and link to where you placed your source files.
2. A screen shot showing your working software
3. The simple documentation log
4. Your maintainability assessment

Grading

We will grade on several factors.

- (20 pts) Whether each item is included
- (15 pts) Whether the modified software works
- (20 pts) Clarity and thoughtfulness of the documentation log
- (20 pts) Clarity and thoughtfulness of the maintainability assessment
- (15 pts) Quality and maintainability of the changes you made to the code (this is why we need access to source files)
- (10 pts) Other factors to be determined while grading such as team collaborations,..