Assignment HW2

**Cover Page**

Prepared for:

Dr. Mehra Borazjany

TA is TBD

Prepared by:

Alex Lundin

Ali Haider

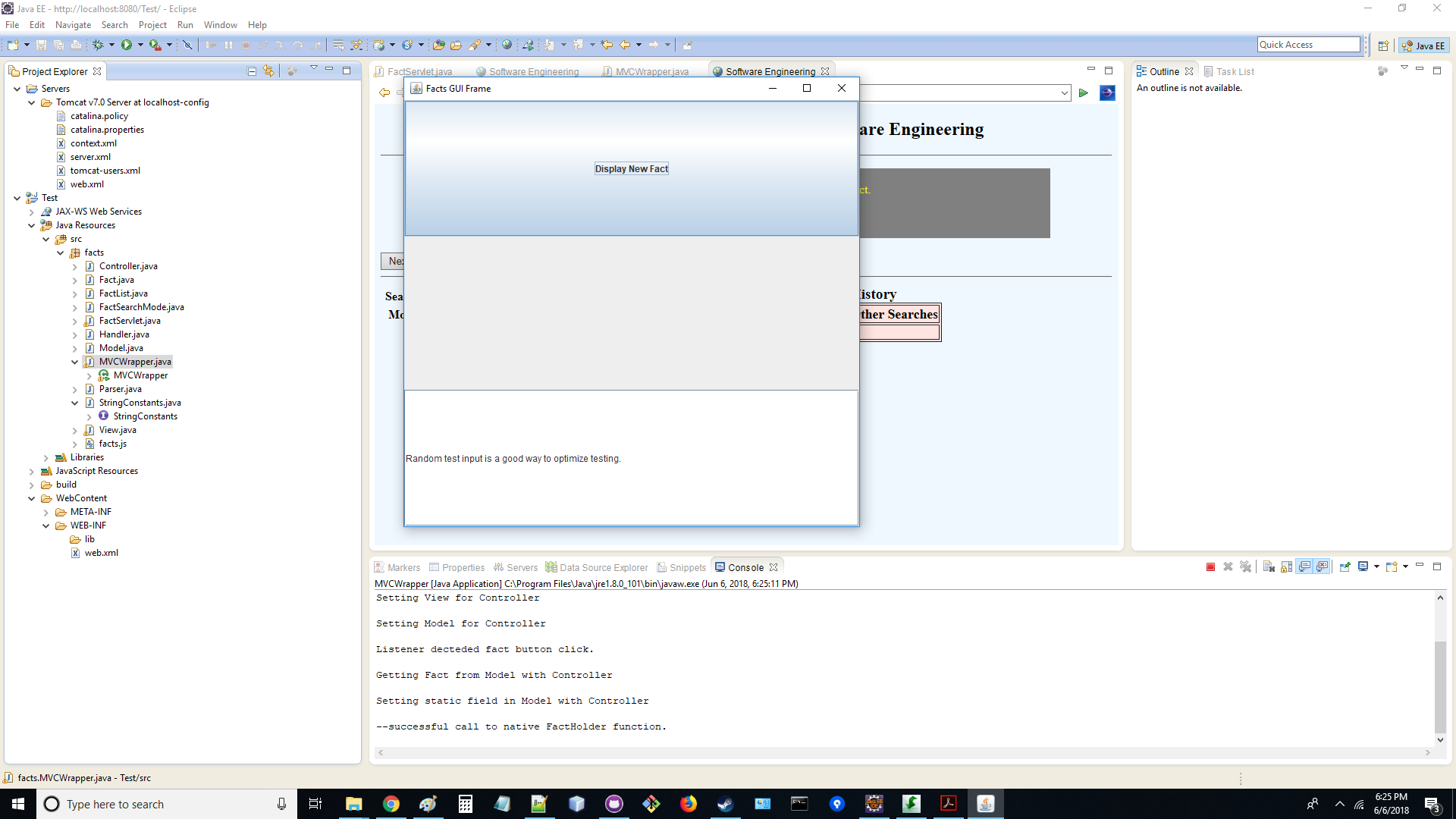
SE-4367.0U1-Testing

**Assignment Choice:**

Simple desktop computer GUI

06 June 6, 2018

**Proof Of Working Software**



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

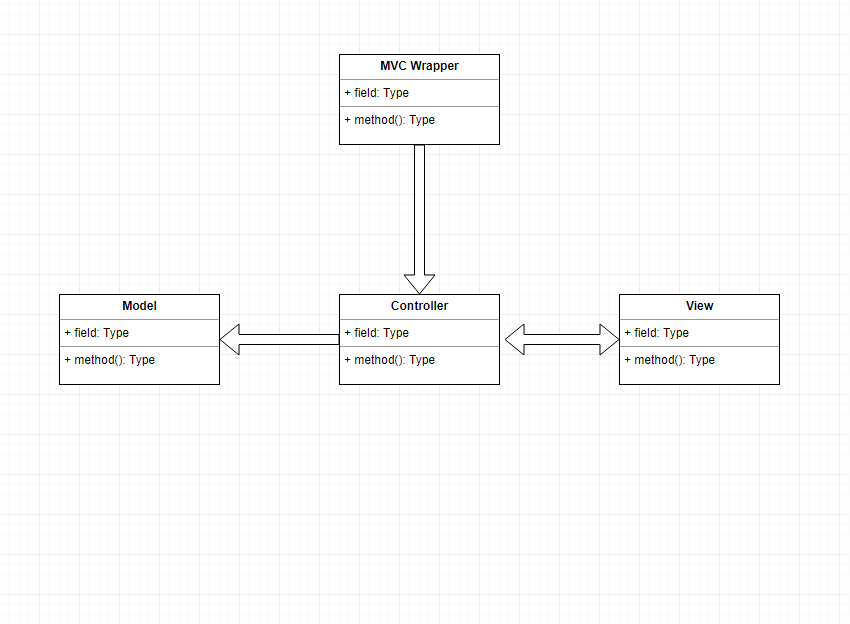
**Documentation Log**

1. Reductions/removals/modifications
   1. JavaServlet.java

* Refactored the hard-coded string paths into their own file
* this makes the files easier to maintain

1. New Files
   1. StringConstants.java

* this is a refactoring of the strings used in the JavaServlet.java file
  1. 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
  2. Controller
* Communication between the model and view
  1. Model
* Holds data
  1. 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**

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

The overall lack of documentation make the reading demand much higher. We had to reach 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.

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

The string constants 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.

**What would you do differently if you did it again?**

I 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 I 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.