

Some reflections made during the project:

- I messed up the RED / GREEN / REFACTOR every now and then, sorry for that!
- If I found a bug or something that was strange I fixed it in the REFACTOR regardless

The task started as a console based Yatzy game, but due to requests from my dedicated junior beta testers I changed it to a GUI instead. I hate making GUIs and it was probably 7 years since I played around with java.swing the last time. This caused some bad design issues.

- I should have deleted the old console beginning I made.
- When doing a GUI I think that it would have been better just to plot all the objects on the window first and then later add the "backend" functions. I did not do it this way. Instead I made it more Agile and added things bit by bit and that resulted in revisiting test cases and modification of these.
- I might not have been fully following TDD as I added some of the GUI parts without modification on the test cases. I think some of them might have been modified more if I would have been fully true to TDD.
- A pure AAA is not followed in the test cases (Assign, Act, Assert) and only one Assert per test case is not followed. However there are millions of versions on how to do it.
- Mocking, well we normally avoid mocking in the sense that everything is fake using Mockito (<http://www.vogella.com/tutorials/Mockito/article.html>) as in our case it just doesn't add much. Hence we rather extend the class and this however opens up the class as methods etc are setup as protected instead of private. This is a trade off with this approach.

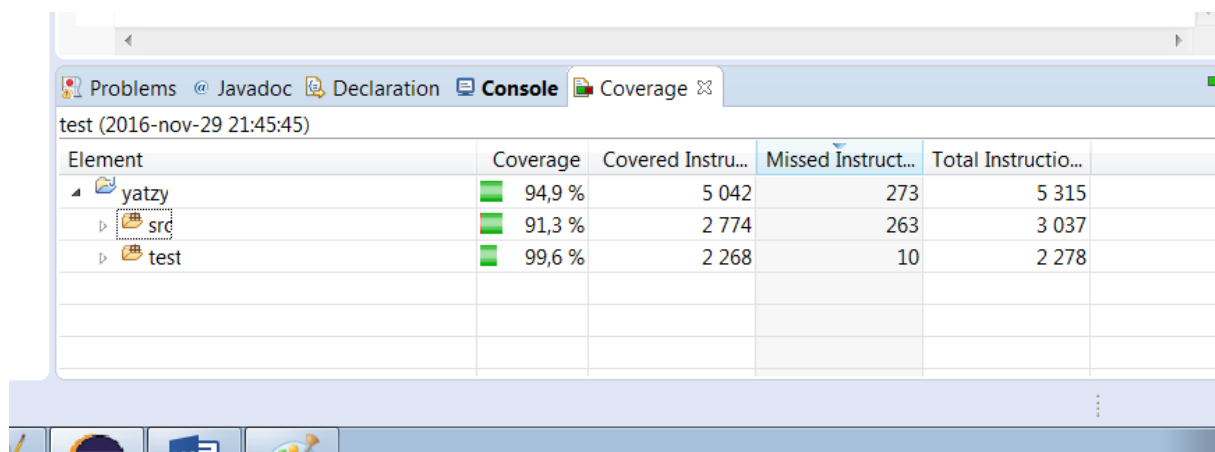
Summary:

If I would do this all over again I would do it this way:

1. First test case would just cover all Swing objects on the board (assert them as notNull)
2. Implement all Swing Objects
3. Make the test case Object per Object
4. Implementation Object per Object

Using this approach I think the end result would be a bit cleaner and some of the logical division between model and view would have been clearer. There are some parts especially in the state handling when playing the game that should rather be in model but is now in view due to a lot of updating of GUI elements that should have been done differently.

Code coverage is 94% in total and 91% of the application is covered in the test cases.



The screenshot shows an IDE window titled 'test (2016-nov-29 21:45:45)' with a 'Coverage' tab selected. The window displays a table of code coverage data for the 'yatzy' project.

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
└─ yatzy	94,9 %	5 042	273	5 315
└─ src	91,3 %	2 774	263	3 037
└─ test	99,6 %	2 268	10	2 278