Project Overview

From Scott Prigmore

Project repo: <https://github.com/tprigmore/p_2_skunk>

This project is a Java implementation of the dice game Skunk. The game supports multiple players and plays the full game.

This implementation is a complete redo of first implementation of P1. In this second implementation of the SkunkApp, the main goals were to do a better job of test-driven design (TDD), test coverage and layer separation.

In the first Skunk implementation, I used TDD. However, I struggled to do this correctly. I took too many short cuts. I was able to get good test coverage (in the 90 % range). However, the tests and design were not that clean.

In this implementation, I used TDD properly. I created a first failing test, then created the test. I used Eclipse to create classes, data, and methods as required by the tests. So, the tests came first. I would then create a test and add logic to the class to get a simple passing test. Then I would add more functionality until the code satisfied the tests.

I used TDD all the way up to the controller. This was the highest level of software below the PL. The code was cleaner, and I had 100 % coverage. There was no unused code at this point.

There were a couple things I learned in the second pass at TDD. Tests became more difficult and longer as I got to the higher levels. The lower lever classes were straightforward.

Also, I found it hard to resist the capabilities of Eclipse. For example, I would need a setter. I used Eclipse to create it, but it also would create a getter. This violated TDD a bit. Also, I may never need the getter. But it was automatic in Eclipse. There are lots of other code generating features built into Eclipse. Good time savers but they are not in the TDD process.

But overall, I learned that TDD has many benefits. It takes a lot of discipline to stick to the TDD process. I need to improve my skills with it because I found times where I just had to violate the process.

My first pass at layer separation required me to recode the higher levels and add a couple classes. These high-level changes required that I make a lot of changes to the tests. I lost the TDD methodology at this point. The controller test coverage suffered a little bit because of this.

Also, the first implementation has poor separation of the presentation layer (PL) and the domain layer (DL). The PL has too much of the game control logic. I struggled to separate the two layers. With Professor Level’s guidance, I was able to come up with an implementation with good separation.

To play the game, clone the design from the above project git repo. The repo contains the Eclipse project. In Eclipse, right-click on the “SkunkApp.java” and select “Run As” > “java implementation”. If you don’t know how to play the game, the application will ask you if you want to see the rules.

This game is fun to play. Well at least for the first 50 times or so. 😊