Team Project TP-2.3:

Submission Details for TP-2 Artifacts

John Luukkonen, Rhyan Foo Kune

University of St. Thomas

Software Analysis and Design

SEIS 635

This document describes our submission for TP-2 and offers a synopsis of the required deliverables. Each artifact chosen for our project is outlined and summarized in this document.

GitHub Repository: https://github.com/rfookune/TP2-skunk-app

SHA-1 identifier: 470366648f8144877c4ca63bc63eaa469c268625

PART 1: Project Description

During our TP-1 development cycle, we did not take a test-first approach. For our TP-2 we decided to reimplement the complete Skunk Game application using Test-Driven Development and agile process. Our goal was to compare the difference in quality and robustness of our software between a TDD approach and the standard approach taken during our TP-1 development.

PART 2: Artifacts

This section describes the artifacts required for our TP-2 submission as well as any links and instruction needed to access those artifacts.

2.1 Basic Deliverables

This section includes all basic requirements for our TP-2 submission.

2.1.a Form a team of 2-4 students and select a project

Our team consist of John Luukkonen and Rhyan Foo Kune.

2.1.b An application with working Java code

Our final Java application submission for this project can be found at the following repository:

GitHub Repository: https://github.com/rfookune/TP2-skunk-app
SHA-1 identifier: 470366648f8144877c4ca63bc63eaa469c268625

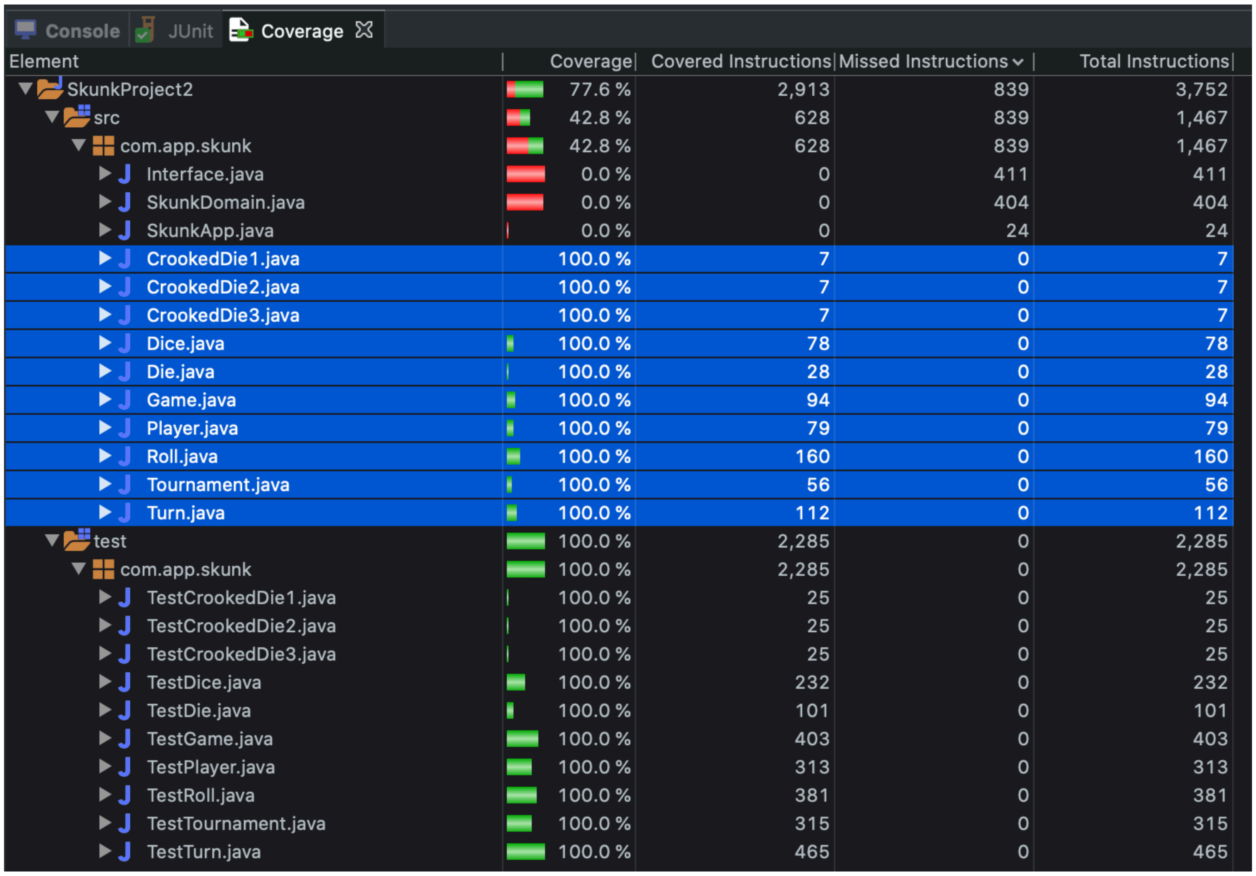
2.1.c Code management using Git and GitHub

Throughout this project we used git for version control and GitHub as our git repository hosting service platform. *Please refer to **2.1.b** for the link to our GitHub repository for this project*.

Master Branch Commits: https://github.com/rfookune/TP2-skunk-app/commits/master

2.1.d Comprehensive JUnit tests for all the non-UI code in your application

To test the code coverage of our application we used a free Java code coverage tool for Eclipse called ECLEmma. The results as shown below reflects 100% code coverage for all our business logic classes. We did not write any test units for our UI Layer or Controller (`Interfce`, `SkunkApp` and `SkunkDomain` classes).



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
SkunkProject2	77.6 %	2,913	839	3,752
src	42.8 %	628	839	1,467
com.app.skunk	42.8 %	628	839	1,467
Interface.java	0.0 %	0	411	411
SkunkDomain.java	0.0 %	0	404	404
SkunkApp.java	0.0 %	0	24	24
CrookedDie1.java	100.0 %	7	0	7
CrookedDie2.java	100.0 %	7	0	7
CrookedDie3.java	100.0 %	7	0	7
Dice.java	100.0 %	78	0	78
Die.java	100.0 %	28	0	28
Game.java	100.0 %	94	0	94
Player.java	100.0 %	79	0	79
Roll.java	100.0 %	160	0	160
Tournament.java	100.0 %	56	0	56
Turn.java	100.0 %	112	0	112
test	100.0 %	2,285	0	2,285
com.app.skunk	100.0 %	2,285	0	2,285
TestCrookedDie1.java	100.0 %	25	0	25
TestCrookedDie2.java	100.0 %	25	0	25
TestCrookedDie3.java	100.0 %	25	0	25
TestDice.java	100.0 %	232	0	232
TestDie.java	100.0 %	101	0	101
TestGame.java	100.0 %	403	0	403
TestPlayer.java	100.0 %	313	0	313
TestRoll.java	100.0 %	381	0	381
TestTournament.java	100.0 %	315	0	315
TestTurn.java	100.0 %	465	0	465

2.1.e A final team presentation of your project to the class

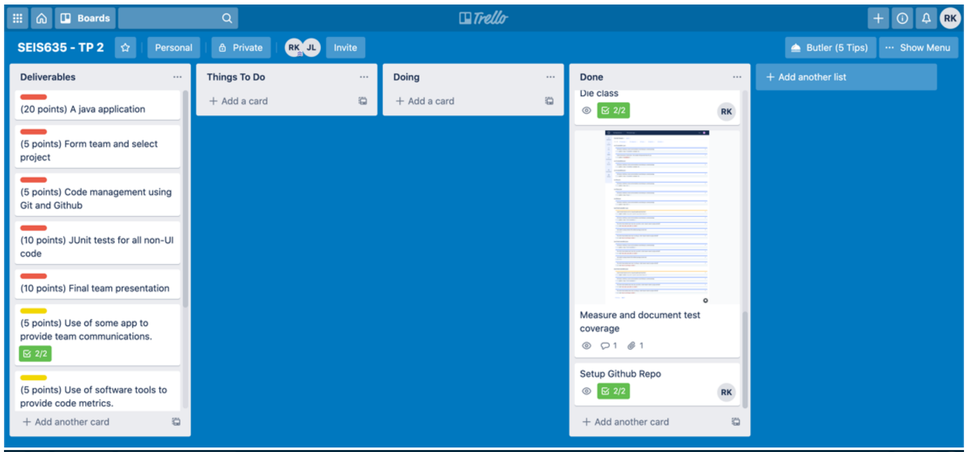
Please refer to the 635-TP2-presentation_Luukk_Fooku.pdf file for a copy our team presentation.

2.2 Optional Deliverables

Beyond the required deliverables outlined above, we also choose the following additional optional deliverables as part of our final project submission.

2.2.a Use of some app to provide team communications

We used Microsoft Teams for file sharing as well as daily team communication. We also used a Trello board for task management and to keep track of our deliverables.

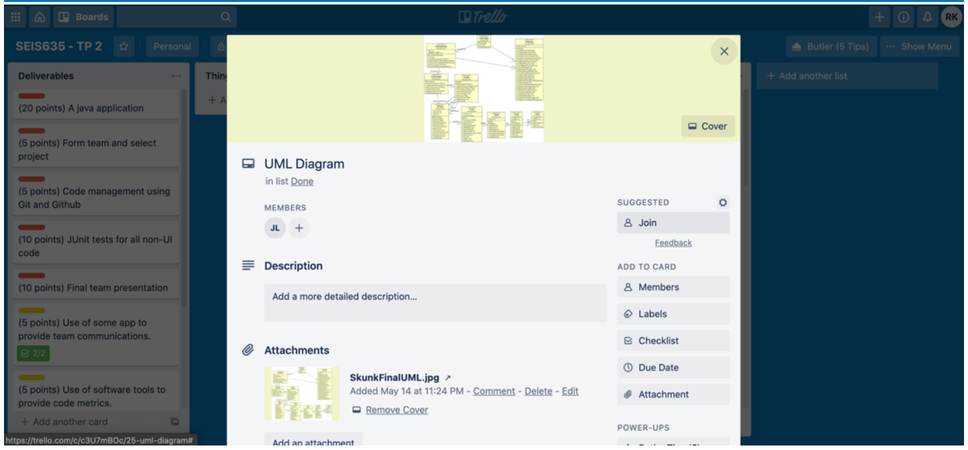


Deliverables

- (20 points) A java application
- (5 points) Form team and select project
- (5 points) Code management using Git and Github
- (10 points) JUnit tests for all non-UI code
- (10 points) Final team presentation
- (5 points) Use of some app to provide team communications. 2/2
- (5 points) Use of software tools to provide code metrics.

Done

- Die class 2/2 RK
- Measure and document test coverage 1 1
- Setup Github Repo 2/2 RK



UML Diagram

in list Done

Description

Add a more detailed description...

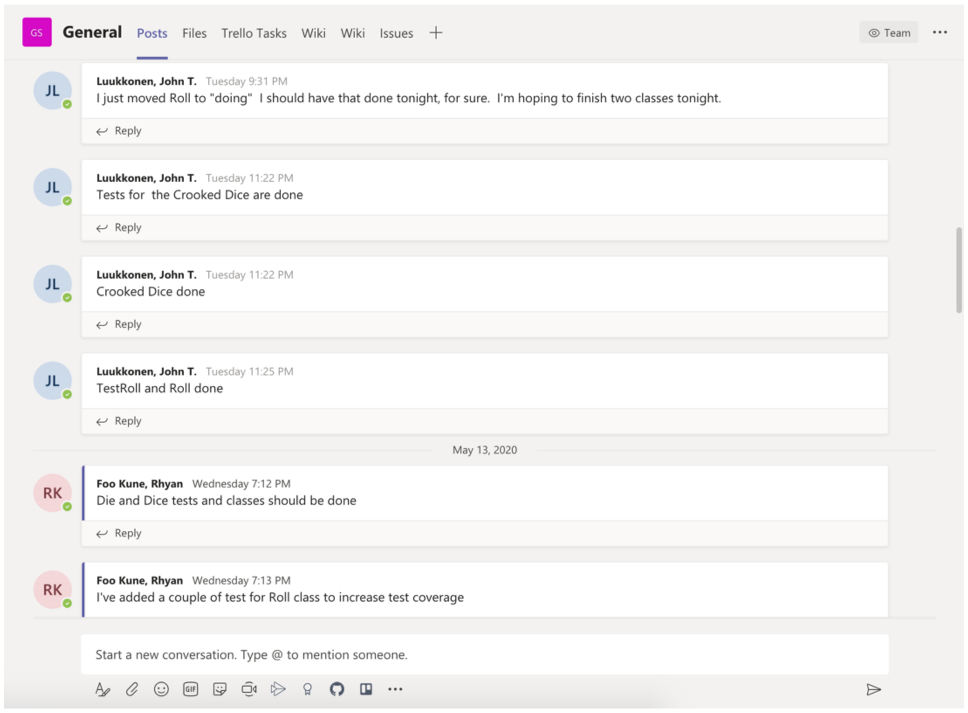
Attachments

SkunkFinalUML.jpg

Added May 14 at 11:24 PM - Comment - Delete - Edit

Remove Cover

https://trello.com/c/c3U7mBOc/25-uml-diagram#



Luukkonen, John T. Tuesday 9:31 PM
I just moved Roll to "doing" I should have that done tonight, for sure. I'm hoping to finish two classes tonight.

Luukkonen, John T. Tuesday 11:22 PM
Tests for the Crooked Dice are done

Luukkonen, John T. Tuesday 11:22 PM
Crooked Dice done

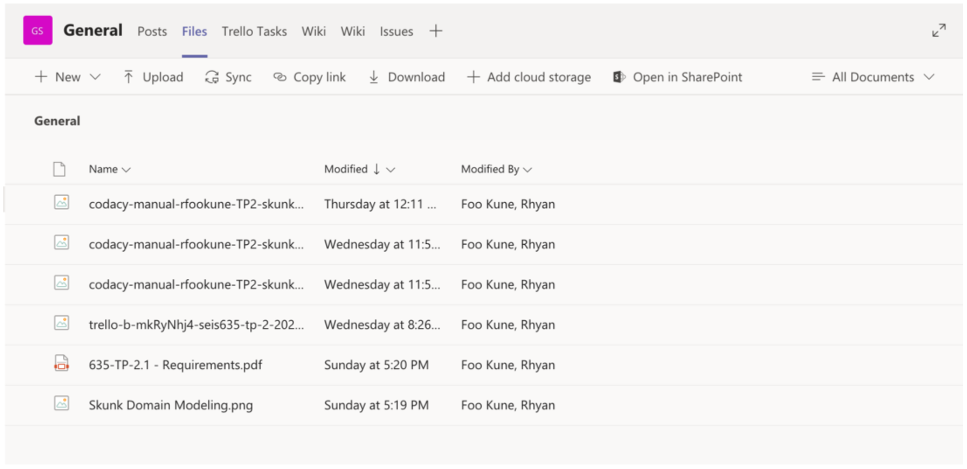
Luukkonen, John T. Tuesday 11:25 PM
TestRoll and Roll done

May 13, 2020

Foo Kune, Rhyan Wednesday 7:12 PM
Die and Dice tests and classes should be done

Foo Kune, Rhyan Wednesday 7:13 PM
I've added a couple of test for Roll class to increase test coverage

Start a new conversation. Type @ to mention someone.

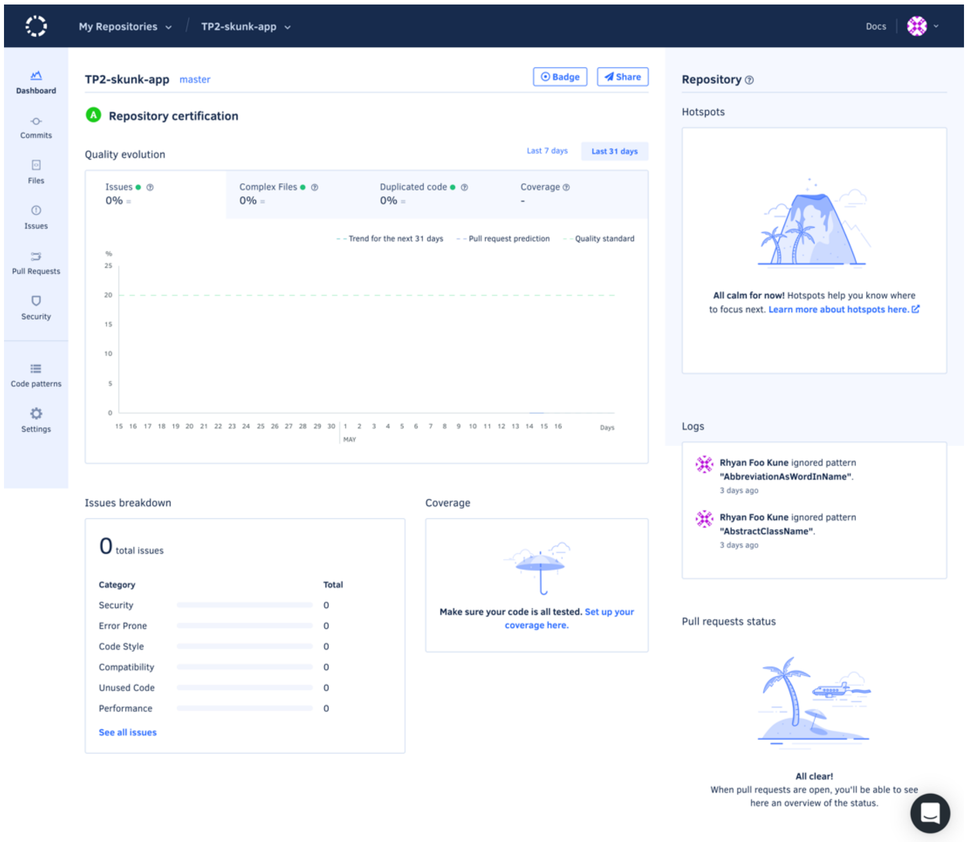


General

Name	Modified	Modified By
codacy-manual-rfookune-TP2-skunk...	Thursday at 12:11 ...	Foo Kune, Rhyan
codacy-manual-rfookune-TP2-skunk...	Wednesday at 11:5...	Foo Kune, Rhyan
codacy-manual-rfookune-TP2-skunk...	Wednesday at 11:5...	Foo Kune, Rhyan
trello-b-mkRyNhj4-seis635-tp-2-202...	Wednesday at 8:26...	Foo Kune, Rhyan
635-TP-2.1 - Requirements.pdf	Sunday at 5:20 PM	Foo Kune, Rhyan
Skunk Domain Modeling.png	Sunday at 5:19 PM	Foo Kune, Rhyan

2.2.b Use of software tools to provide code metrics

In order to run code metrics on our applications we used a web-based software called codacy. Codacy aim to automate the process of static code review. It connects directly to your code repository and run a detailed analysis of your code on every commit. Codacy provides a seamless integration in your workflow with integration with GitHub, Slack, Jira, GitLab and BitBucket.



TP2-skunk-app master

Repository certification

Quality evolution

Issues	Complex Files	Duplicated code	Coverage
0%	0%	0%	-

Issues breakdown

0 total issues

Category	Total
Security	0
Error Prone	0
Code Style	0
Compatibility	0
Unused Code	0
Performance	0

See all issues

Coverage

Make sure your code is all tested. Set up your coverage here.

Repository

Hotspots

All calm for now! Hotspots help you know where to focus next. Learn more about hotspots here.

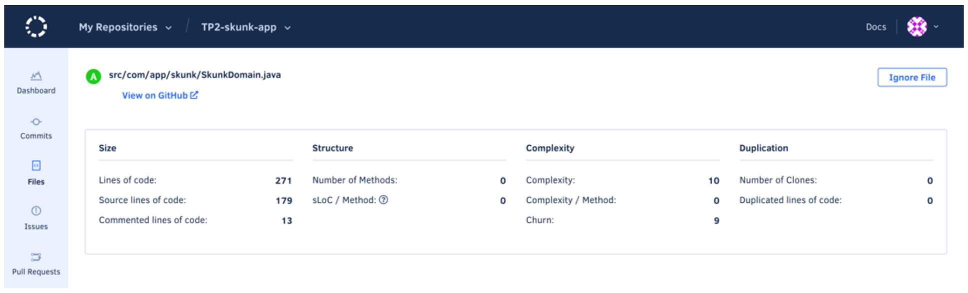
Logs

Rhyan Foo Kune ignored pattern **"AbbreviationAsWordInName"**.
3 days ago

Rhyan Foo Kune ignored pattern **"AbstractClassName"**.
3 days ago

Pull requests status

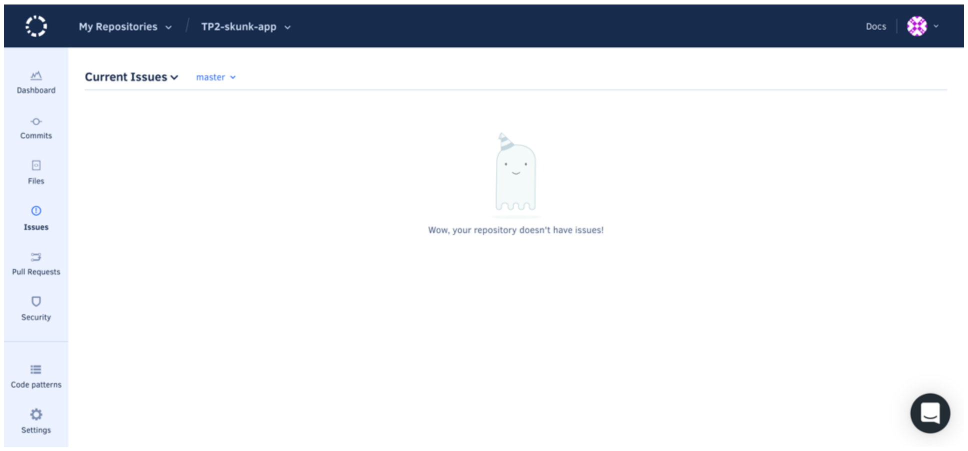
All clear!
When pull requests are open, you'll be able to see here an overview of the status.



src/com/app/skunk/SkunkDomain.java

View on GitHub

Size		Structure		Complexity		Duplication	
Lines of code:	271	Number of Methods:	0	Complexity:	10	Number of Clones:	0
Source lines of code:	179	sLoC / Method:	0	Complexity / Method:	0	Duplicated lines of code:	0
Commented lines of code:	13			Churn:	9		



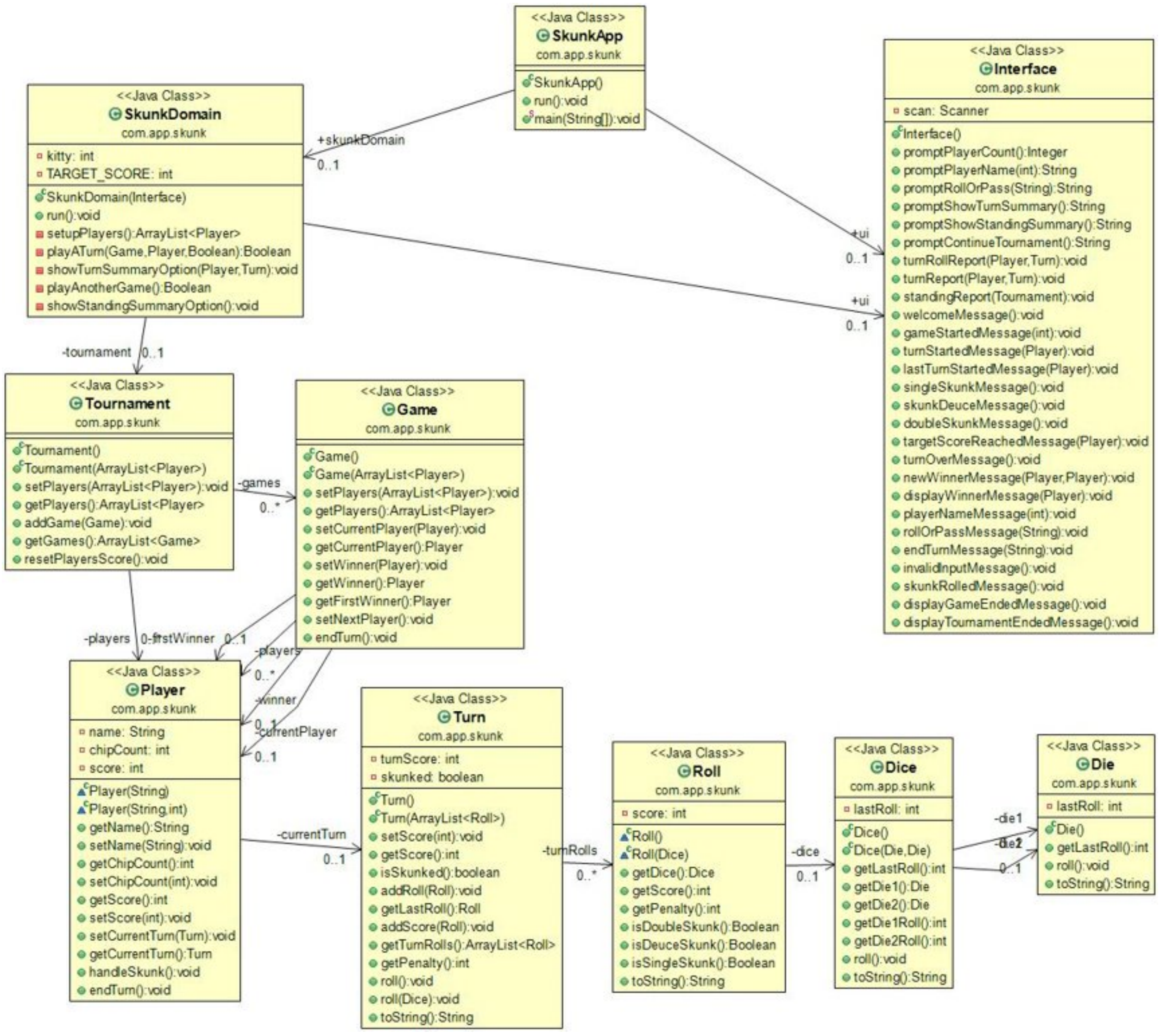
Current Issues master

Wow, your repository doesn't have issues!

2.2.c Superior separation of Presentation Logic (UI) and Business Logic (Domain Layer)

To achieve superior separation of Presentation Logic (UI) and Business Logic (Domain Layer) we followed the Model-View-Controller (MVC) software design pattern. This ensures that all our UI code (View) only interacts with our domain layer (Model) through a controller gateway. There is no direct communication between the presentation logic classes and business logic classes.

This separation can be seen in our UML class diagram shown below. In this diagram, the `Tournament`, `Game`, `Player`, `Turn`, `Roll`, `Dice` and `Die` classes form our models, the `SkunkDomain` class represent our Controller and the `Interface` our View.



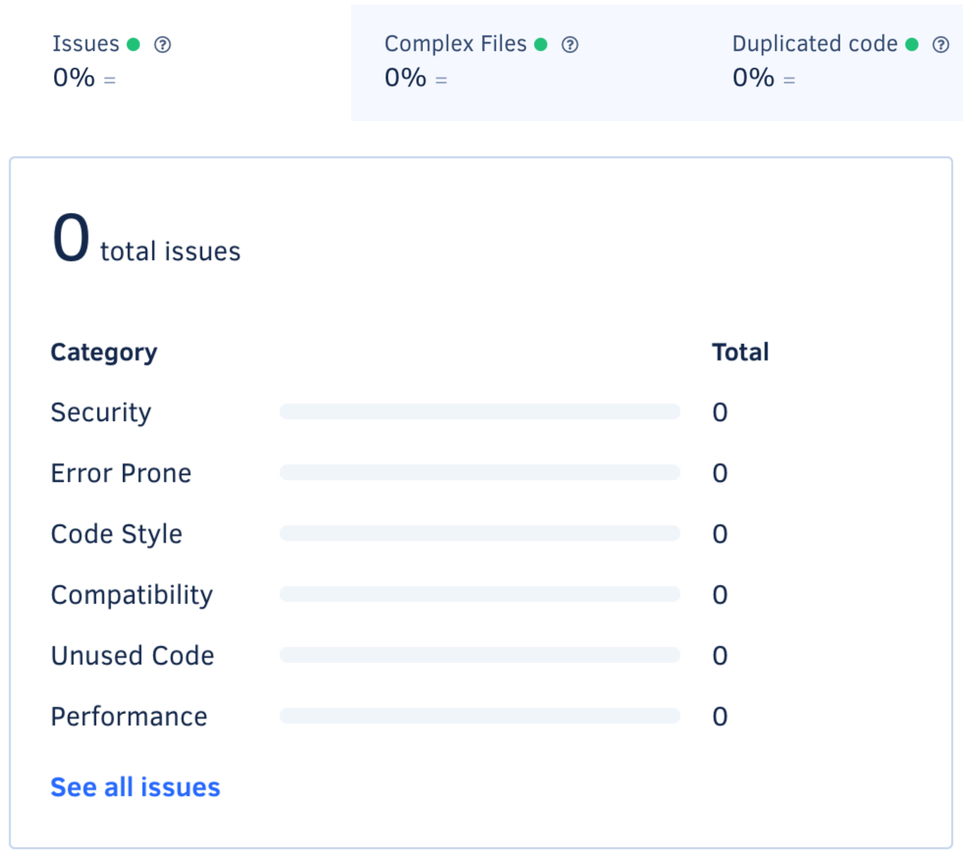
2.2.d Re-implement the complete app using Test-Driven Development (TDD)

For our TP-2 we decided to reimplement the complete Skunk Game application using Test-Driven Development and agile process. Our commits reflect the use of TDD throughout the development of our application, with test being written before the implementation of the class.

Using code metric provided by our code metric software tool, Codacy, we can compare the implementation of the Skunk Game application using our TDD (TP-2) vs. Standard (TP-1) approach.

Besides out code style issues, we can see that we were able to reduce the error prone areas of our applications using a TDD approach.

Skunk Game – TDD Approach (TP-2)



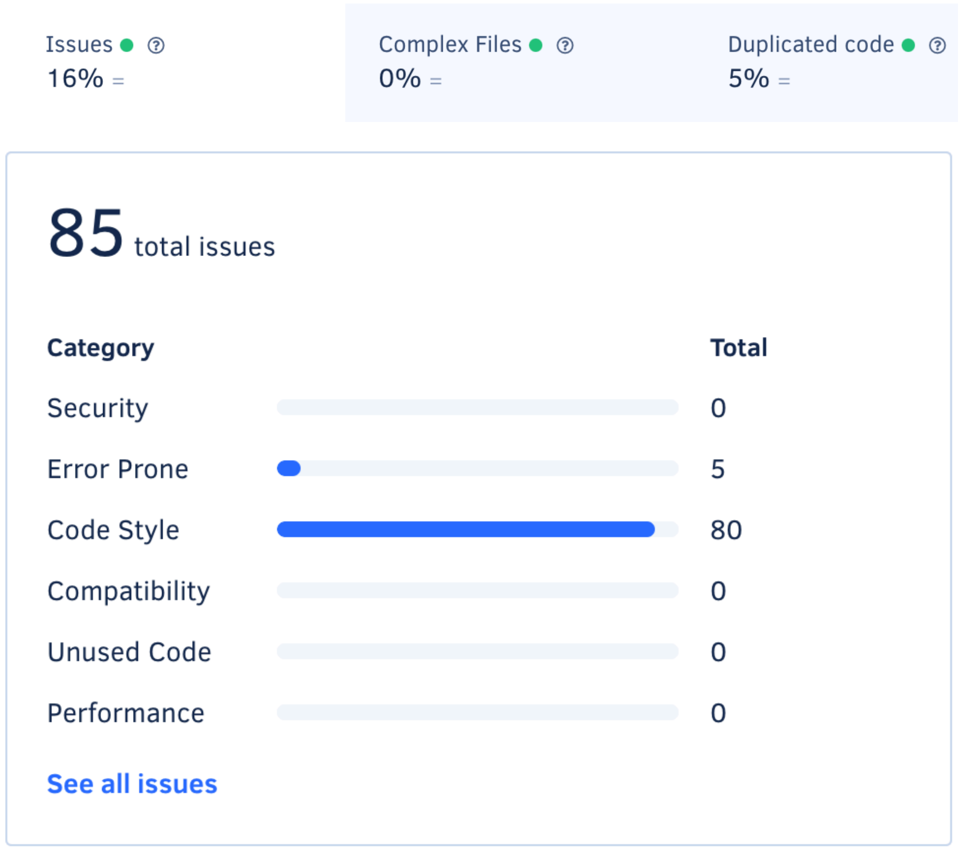
Issues ●	Complex Files ●	Duplicated code ●
0% =	0% =	0% =

0 total issues

Category	Total
Security	0
Error Prone	0
Code Style	0
Compatibility	0
Unused Code	0
Performance	0

See all issues

<u>Skunk Game – Standard Approach (TP-2)</u>



Issues ● 16% =

Complex Files ● 0% =

Duplicated code ● 5% =

85 total issues

Category	Total
Security	0
Error Prone	5
Code Style	80
Compatibility	0
Unused Code	0
Performance	0

See all issues