Assignment 1.1

由 Triphol "Pao" Nilkuha (admin)创建, 最终由 Kim, Yongjin修改于 九月 21, 2018

Assignment 1.1 - Extending Your Chess Library

d documentation, and also extending your library by adding two custom
ing good documentation now - and throughout the assignment 1.X - will
the week, you should have a clean, easy-to-understand, and extensible

oth are free and have powerful refactoring tools available.

here at UIUC. A few staff prefer IntelliJ. If you've never tried it out,

typically **reuse** your code from the previous week for each
ainability.

hing is still unclear.

r moderator, the TAs, or your peers from discussion section suggest
our tests *before* implementing your functionality last week, but if for
g your test suite *before* you begin refactoring. Doing so will help you
gs.

nma for Eclipse, or the integrated code coverage features in IntelliJ

e naming, decompose larger methods into smaller separate methods,

e-packaged binaries here, or run the following command on the EWS
s is run from the root directory of your project (i.e. from

xf doxygen-1.8.2.linux.bin.tar.gz && cp doxygen-1.8.2/bin/doxygen ./

ly run the following:

tain autogenerated HTML & latex found under *html* and *latex*. Take a

Summary

Table of Contents

Reading

- The Joel Test: 12 Steps to Better Code
- Optional: Code Complete chapter 24: Refactoring

Questions that appear in lecture quizzes may come from the assigned reading, so it is in your best interest to complete it.

Submission

This assignment is due at the **beginning of your discussion section the week of September 24th, 2018**. Please be sure to submit in Gitlab, and ask your moderator or TA before the deadline if you have any questions.

Objectives

- Clean up any problems in your code for Assignment 1.0, expand your test suite if necessary, and fix any algorithmic shortcomings
- Auto-generate documentation for your library
- Two custom chess pieces and static user interface using Observer Pattern

Resources

- Design Patterns
  - Design patterns
  - Model-View-Controller architecture
  - Model-View-Controller (MVC) Explained Through Ordering Drinks At The Bar
  - Observer Pattern
- Refactoring
  - Refactoring
  - Refactoring in Eclipse
  - Refactoring in InteliJ

Grading

We will bias clarity over cleverness in evaluating your code. You should adopt a *"teaching"* mindset, always asking yourself, *"How would I change what I have written to make it as easy as possible for someone else to understand my code without my direct assistance?"*

Refer to the standard Grading rubric if any portion of this rubric is unclear. _ Note that the standard rubric assumes a scale of 0-2 for each category_.

Category	Weight	Scoring	Notes
Basic Preparation	2	0-1	Ready to go at the start of section

pand on the documentation of your public methods and classes.
L generated by Doxygen will be this programmer's first contact with your
us using Doxygen which areas could use further explanation. If you do
rate your documentation.

the _latex, *html*, or *doxygen* files). This uses up a tremendous amount of
n on the rubric. **However, please ensure that *Doxyfile*, or whatever
itlab. Without it, we cannot give you credit for the Doxygen-related**

te two custom chess pieces. All the usual chess rules apply to your
nplicity, your custom pieces do not have to implement special moves,
ia useful, but you are free to create your own movement.

Bar

GUI should have **ZERO USER INTERACTION**. The only thing required
our time implementing chess moves, you may even lose points if
'C architecture, by strictly separating the Model, View, and Control
view. Simply display a chess board with the normal set of chess pieces
display a static image in your JFrame.

y careful.

t will not meet the requirements for this assignment without modification.
ng of the autogenerated code will be required. In the staff's opinion, it is
om scratch.

Category	Weight	Scoring	Notes
Cleverness	2	0-2	The hardest points on the rubric
Code Submission	4	0-2	Submitted correct content on time and to the correct location in the repository
Decomposition	4	0-2	Project is adequately decomposed into different classes and methods
Documentation	4	0-2	Comments for each class and each function are clear and are following style guides
Effort	2	0-2	Perform considerable amount of work
Naming	2	0-2	Variable names and method names are readable and are following Java conventions
Overall Design	4	0-2	Have nice approaches and structures in overall
Participation	5	0-2.5	Interact with the group 2 times (ask a question, make a comment, help answer a question, etc.)
Presentation	4	0-2	Present the code clearly
Requirements - Custom Chess Pieces	5	0-2.5	• 2 points - Custom Chess Pieces are a simple combination of two standard chess pieces. Any piece which is in this link would constitute 2 points. • 2.5 points - Custom Chess Pieces are unique (not in this link) and the algorithm used to describe its movement is complex and different from the movements of standard chess pieces.

```
nLookAndFeelClassName());

xample");

CLOSE);

);
```

Category	Weight	Scoring	Notes
Requirements - Static User Interface	5	0-2.5	• 2 points - The UI consists of a neat layout which accurately represents a chess board. The UI consists of pictures for each of the chess pieces. • 2.5 points - The UI is extremely appealing and shows that the student has put in considerable effort in the design of the layout. The UI consists of clickable buttons for each tile and basic interactions are enabled in the UI.
Requirements - Doxygen Generation	4	0-2	• 2 points - Generate and submit a Doxygen file
Requirements - Refactoring	5	0-2.5	• 2 points - There is not much duplicate/redundant code and the student has put effort to refactor code from last week. There is minor but not significant scope for improvement. • 2.5 points - Every piece of code is functional and overall design of the project is so well designed that it cannot be refactored/improved any further.
Testing	5	0-2.5	Unit tests written for all new code & expanded last week's tests if necessary • 2 points - 80% of the new code is tested and the tests cover edge cases for each function. • 2.5 points - 95% of the new code has been tested and every possible edge cases have been tested for.
Testing (Script)	4	0-2	• 2 points - Make a complete manual test plan for GUI
Total	**61**		