~~Odd~~/Even/~~Compact~~\* Semester (2020)

**BINUS UNIVERSITY BINUS INTERNATIONAL**

**Assignment Cover Letter (Individual/~~Group~~\* Work)**

**Student Information**: ***Surname Given Names Student ID Number***

1. Setiawan Nathanael 2440042866

**Course Code : COMP6699 Course Name: Object Oriented Programming**

**Class : L2AC-LEC Name of Lecturer(s) :** 1. Jude Joseph Lamug Martinez

**Major : International Computer Science**

**Title of Assignment** : Dungeons and Draggers

(if any)

**Type of Assignment : Final Project**

**Submission Pattern**

**Due Date : 22/6/2021 Submission Date : 21/6/2021**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I/~~we~~\* understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I/~~we~~\* declare that the work contained in this assignment is my/~~our~~\* own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

|  |  |  |
| --- | --- | --- |
| Signature of Student: | (Name of Student) |  |
| 1. |  |  |
| *etc.* |  | \*) Delete the inappropriate option |

Table of Contents

**Cover page1**

**Table of contents2**

**Project Specifications3**

**Solution Design3**

**Explanation of solution4**

**Evidence9**

**Sources12**

***Project Specification:***

Dungeons and Dragons (5th edition) is a popular table top roleplaying game that was created by Wizards of the Coast. The game involves a number of player’s roleplaying as their own characters in a “campaign”. A campaign is the world the game takes place in, and everything from terrain, buildings, and non-player characters are determined by someone called a Dungeon Master (DM). A DM’s role is extremely important, as they determine how interactions play out, what players see through the character’s eyes, and how combat encounters work. This is why a DM will regularly use multiple tools to more easily remember and keep track of everything that goes on in the campaign. While there are already a multitude of different software tools available to use for creating and keeping track of worlds, maps, characters, etc. The tools that are available for combat, which is a massive aspect of the Dungeons and Dragons experience, are sadly limited. Sophisticated software tools such as Foundry Virtual Tabletop cost 50$ USD (722,572Rp) and free alternatives makes it difficult to keep track of the character’s data, which is detrimental to the general flow of the game.

Because of the inaccessible nature of combat tools for DMs, I have decided to create a tool which will help in this situation. The biggest need of this solution is a grid and pieces system. Since Dungeons and Dragons utilize a grid combat system, creating a grid to act as the ground is important. The pieces are also important, as they represent players/enemies. This grid system is going to be what the program will be based on. Additional features which can be included are a way to create new pieces, discern the pieces, holding data, and showing data.

***Solution Design:***

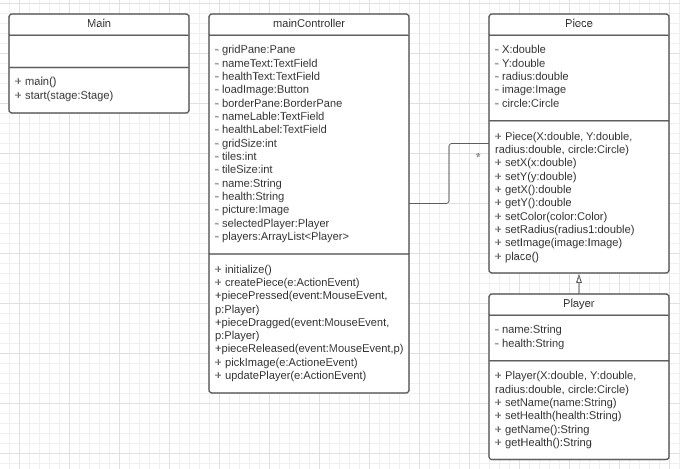
The solution will be a combat DM screen in the form of an application. The application will try to help organize and help the DMs and will utilize the JavaFX library, using exclusively JavaFX functions and variables alongside basic Java functions and variables. The solution should contain 3 aspects, the aspects consists of the stage window where everything will be placed inside, the grid and pieces which the main body of the program will consist of, and the side bar where additional features can be implemented.

The stage window will be a simple window, with a custom logo that has been created from the Dungeons and Dragons logo PNG, and a mouse pointer PNG, both of these image credits can be found in the ***Sources*** section of this document. This logo will be shown on the taskbar and on the top left of the window itself. This window should contain the entirety of the project, being filled with mostly the grid.

The grid system will be built around the code from the 2 tutorials by Mark Goadrich, with full credits in the ***Sources*** section of this document. The grid will be comprised of many rectangle objects, positioned such that it creates a grid. This grid will be static, meaning there will be no event functions assigned to them. The pieces on the other hand will be a circle which will have the diameter equal to the width of the grid tiles. These pieces will be assigned functions so it can be dragged, dropped, and snap onto the grid. The pieces are also filled with multiple properties which can be get and set using getters and setters. The pieces should still function with null values, as to allow customization for the users.

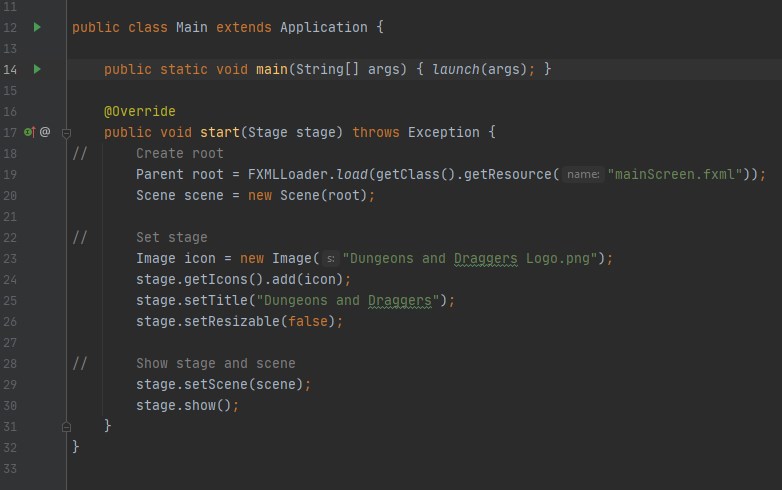
Finally, the side bar will consist of different buttons and input fields. This side bar will be used to store useful controls such as a button to create pieces, an input field to assign values into, and many more potential uses. This side bar could also be used to show the values of the pieces, and other potential more uses. Another function which has been implemented is an image loader, where the code is based on another tutorial’s code, by Cool IT Help, with full credits in the ***Sources*** section of this document.

The UML diagram:



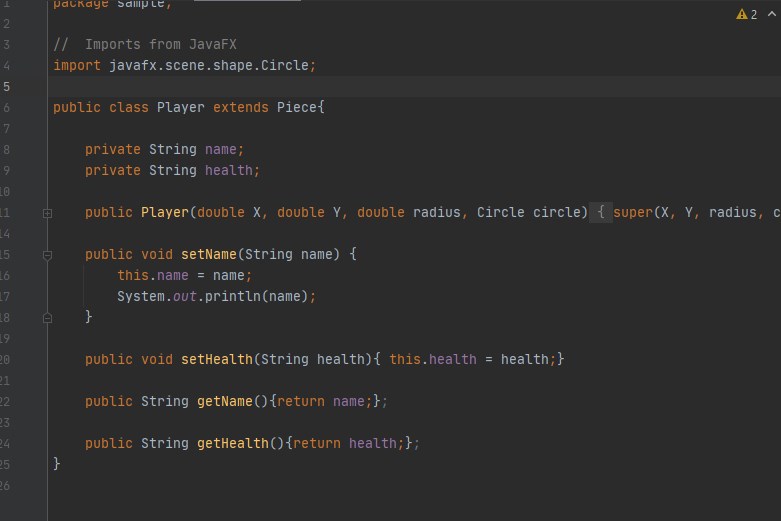
***Explanation of solution:***

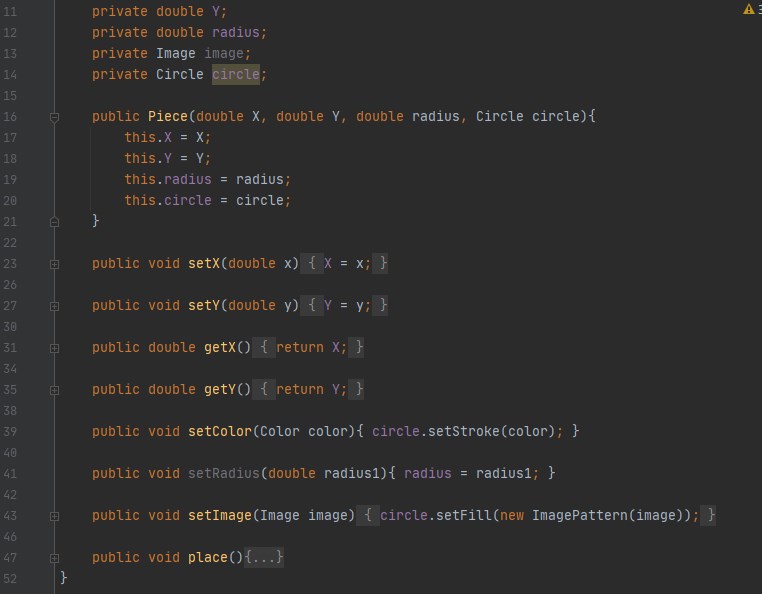
**Stage window:**

****

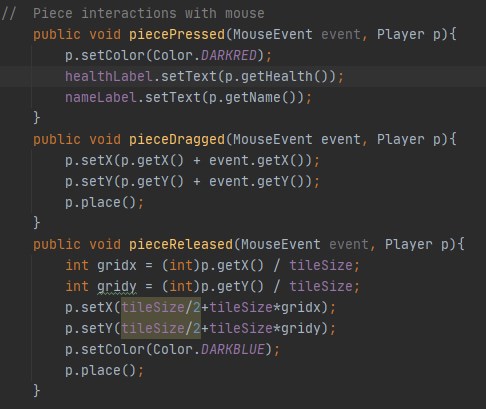
The code for the stage window is a simple main() driver code which creates the roots, sets the stage, and shows the stage and scenes.

**Grid and pieces:**

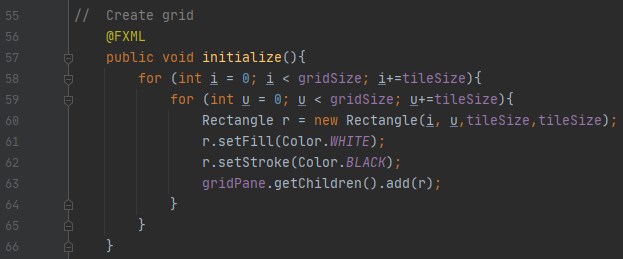




The pieces are basically objects which can be interacted with on the grid. The pieces and players class are created so that future types of pieces can extend from the original piece class. These pieces are mostly controlled inside of the main controller code, alongside initializing the grid and creating the pieces button.

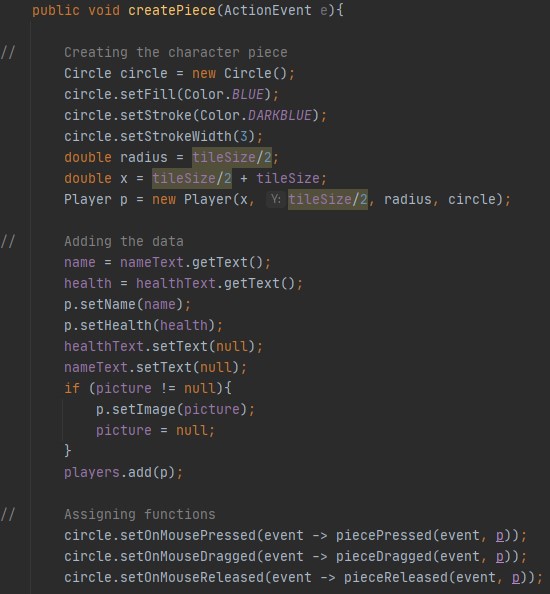


This code determines the behavior of the pieces when they are clicked, dragged, and released. The click function indicates when a piece is clicked/held, where the piece will change the color of its border, and load the values stored within that piece object and displays it on the sidebar. The drag function simply allows the user to drag the piece with visual feedback. And finally the released function snaps the piece to a grid place which it is closest to. The create piece code will be discussed in the sidebar section

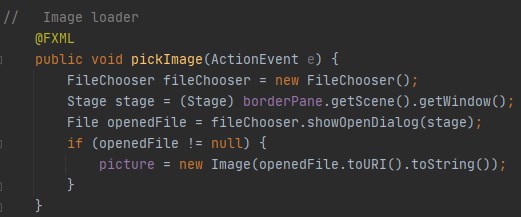


This is the code to initialize the grid. The grid is placed inside a pane, and the grid is made up of rectangle objects which are initialized in a loop. The code for grid initializing and pieces properties are a modification of Mark Goadrich’s code from the tutorial.

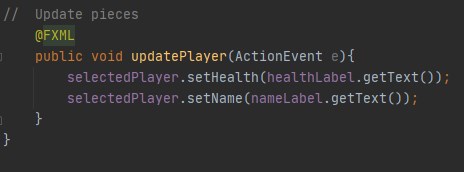
**Sidebar:**



This is the code to create the pieces object. This is assigned to an action event to the create button. It will take the values of the text areas and it will assign them to the properties of the pieces.



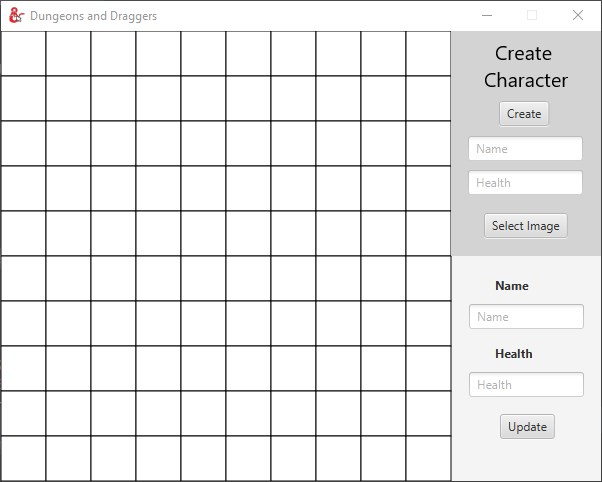
This code is used to select image files and convert them into an Image variable. The image will be used as the fill of the piece. This code is built of off the code in Cool IT Help’s tutorial.



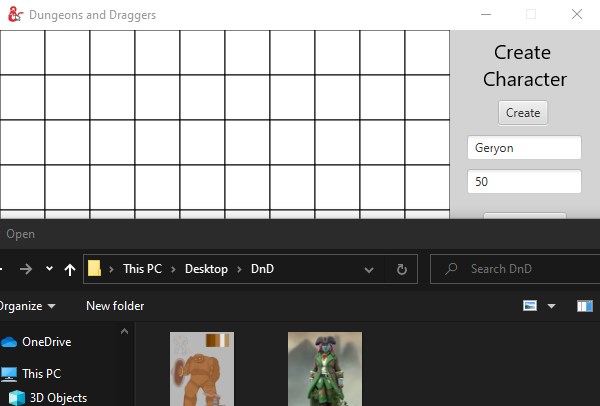
This code is used to update the selected player’s data when the button is pressed in the sidebar and the fields have been changed.

***Evidence:***

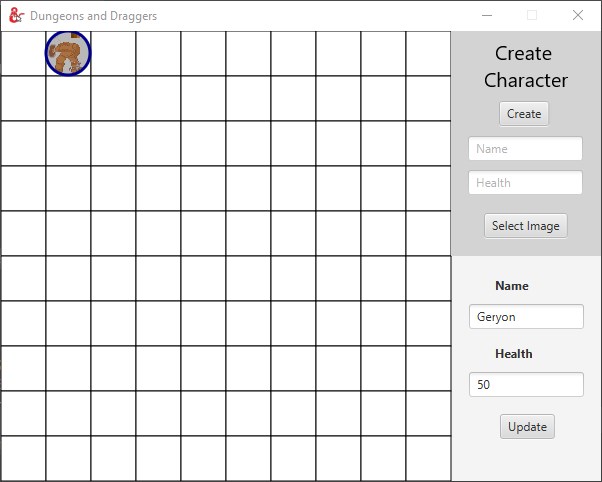
Screen after starting the program:

**

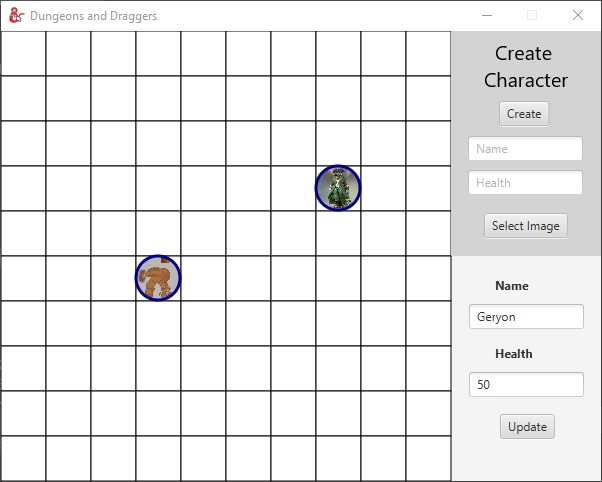
Inserting image:

**

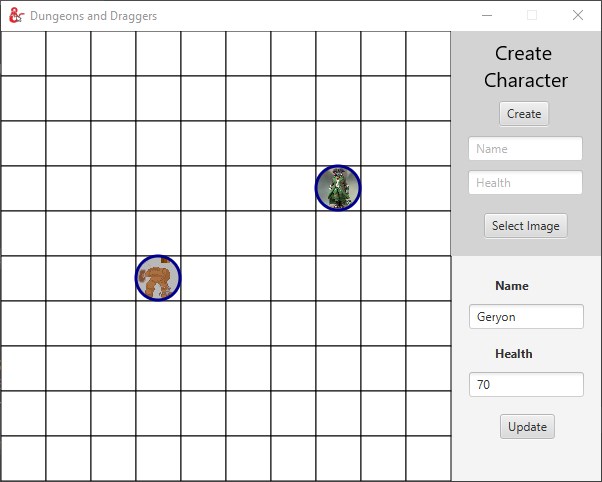
After piece inserted:

**

Piece moved around with 2 pieces:

**

Updated piece information:

**

***Sources:***

***Tutorials:***

**JavaFX:**

Bro Code, (22, 3, 2021), JavaFX GUI Course ☕ (𝙁𝙧𝙚𝙚), <https://youtu.be/9XJicRt_FaI>

**Grids and Pieces system:**

Mark Goadrich, (25, 4, 2020), Snap to Grid Demo in JavaFX: 1 - Grid and Pieces, <https://youtu.be/9YWYxj6LAUE>

Mark Goadrich, (25, 4, 2020), Snap to Grid Demo in JavaFX: 2 - Movement and Snapping, <https://youtu.be/ME6WfnR6zys>

**Image input system:**

Cool IT Help, (30, 12, 2017), Single FileChooser & Multiple FileChooser Tutorial | JavaFX FileChooser, <https://youtu.be/sC5Er7jQZN4>

***Image sources:***

**Dungeons and Dragons Logo PNG:**

Pngfind, (n.d.), <https://www.pngfind.com/mpng/ihwowoT_d-d-logo-png-dungeons-dragons-transparent-png/>

**Mouse PNG:**

Nicepng, (n.d.), Mouse Pointer Comments - Mouse Icon Png White, <https://www.nicepng.com/ourpic/u2e6w7u2i1w7q8o0_mouse-pointer-comments-mouse-icon-png-white/>