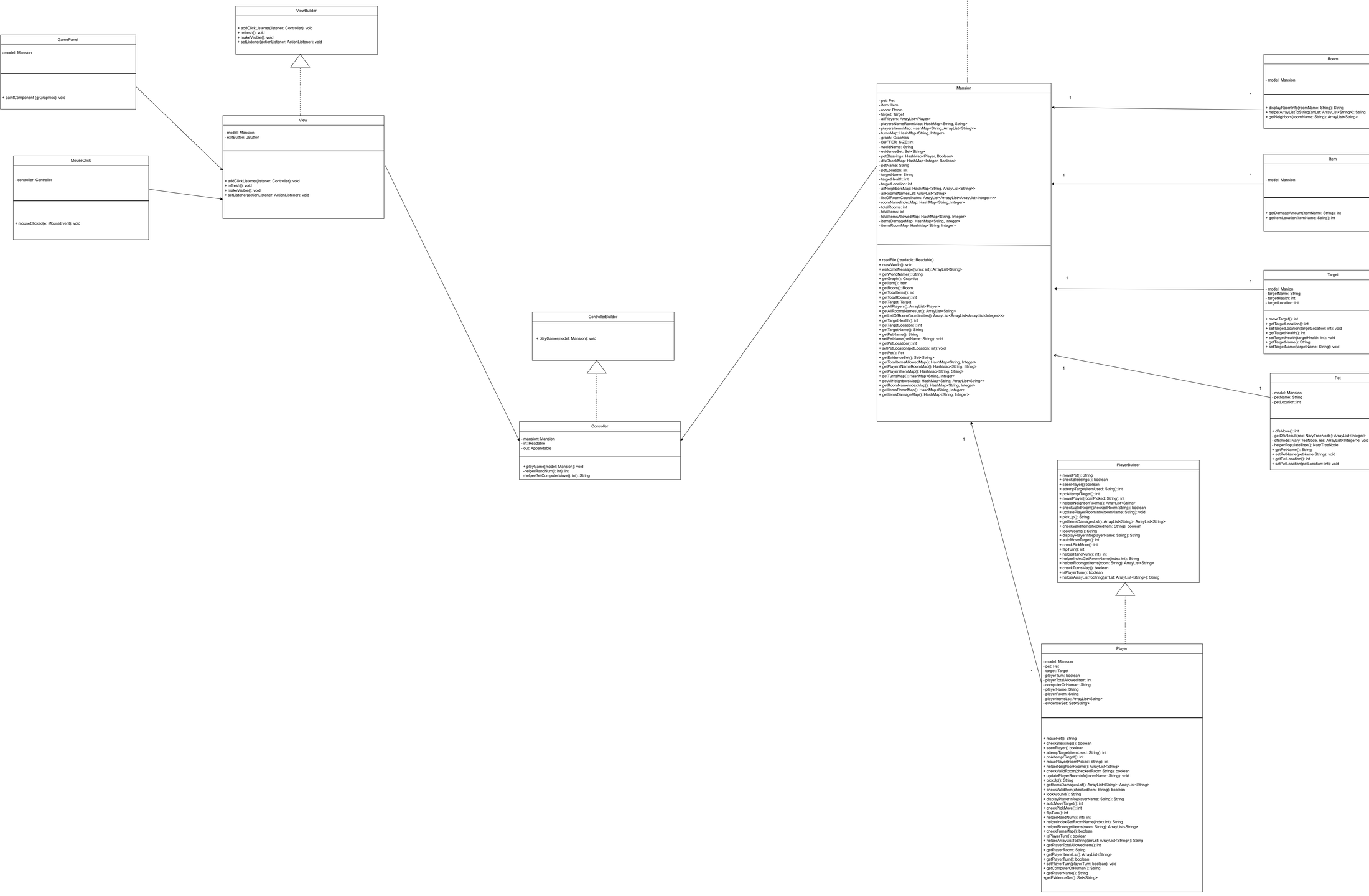

Testing Plan (Milestone 4 - View)

Since this part of the milestone will be about implementing the View. Thus, thetesting of will be also focusing on the correct implementation of the Viewclass:
1. viewBuilder.java
2. View.java
3. GamePanel.java
4. MouseClick.java
/
Because this part of the implementing will be on the GUI side, so it will be adone a bit differently for the testing than how we done it in the previousmilestones.

1. viewBuilder.java is just an interface for the View. so there won't be anynecessary testing here but make sure the interfaces that covers all the majormethods for View.java.

2. View.java will be tested through a mock testing.
1). 1st to create a mock class named: ViewMock.java. This will be the sameway I implemented the View.java class but adding the appendable in and toappend information to it. so I could use such information for the testing (verysimilar to the mock testing I've done for the model with the controller inmilestone 2.

3. GamePanel.java since I haven't really started the real implementation, but Ithink I'll mostly use the paintComponent() method to draw the Game play.Thus, for the testing it will be very similar to the 2. (1) testing; it will be also amock testing since it's not possible / very hard to test the GUI directly withcode. As long as I make sure the appendable information matches with myexpected string information. Then the testing will be considered as passed.

4. MouseClick.java
1). test if the controller correctly handles the handleCellClick with validarguments. This class will also use the idea of mock testing. Where I couldappend certain values into appendable everytime the use click the mouse itwill trigger the appendable condition inside a MouseMock.java class.

2). test handling click error: one test that tests that the controller correctlyhandles the handleCellClick with arguments that are out of bounds. Similar tothe 4. (1) testing, this testing will be done with the appendable as well. If theuser clicked some places out of the bounds, I'll just append those 'X' and 'Y"values that the user clicked on the board with the mouseClicked(MouseEvente) method that defined in the MouseAdapter class (note: this MouseAdapterwill be extended by this MouseClick.java class specifically to override thismouseClicked() method.

Hence, the above testings will cover all the testings for the View part of thedesign. Besides that, I barely can see how else I'll be testing the view; all theactual methods are more in the "model" part of the design; the "View" part ofthe testing will be very similar to the "Controller" part of the testing whereutilizing the "Mock testing" will be essential for this part of the testing as well.

Addtional concerns: besides the view part of the testing, since I kind ofomitted the testings for the newly added methods in the "Model" during theimplementation of the Milestone 3 (because I kind of though milestone ismore about connecting to the controller and testing the controller; also I wasalso running a bit out of time to finish all the testings before the deadline). Iwill try to update my milestone 3 part of the testings as well to make sure thatmy testings will also be a very holistic type of testing for this "MVC" milestone.