Refactoring

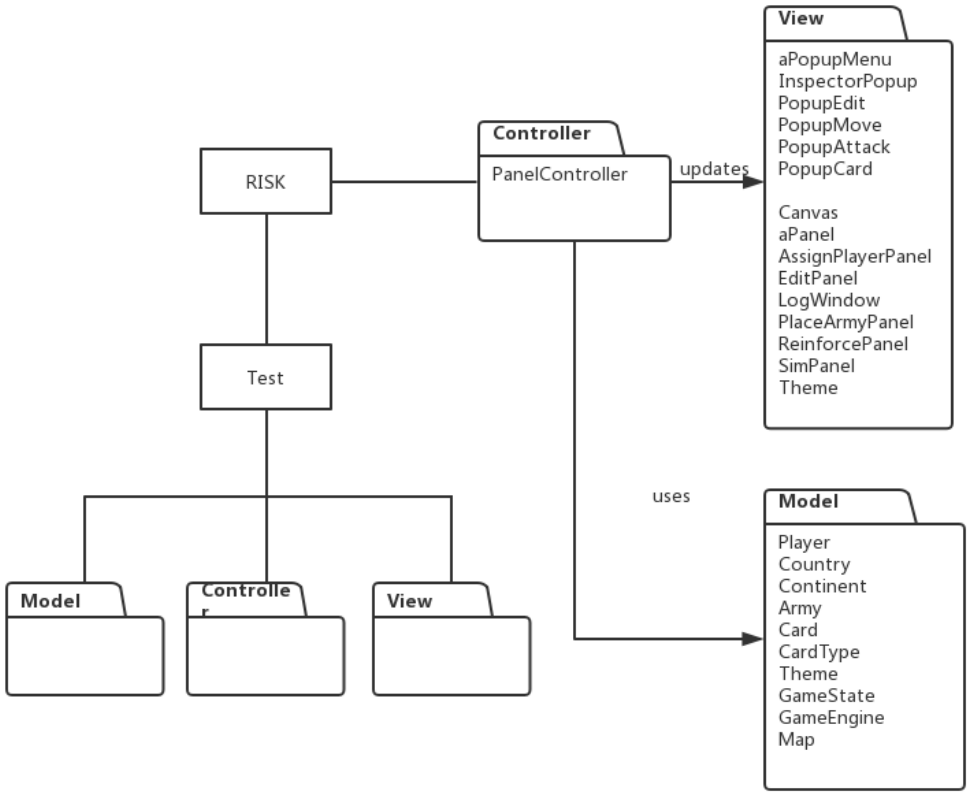
1. architecture refactoring

The game uses the MVC software architecture to divide the software system into three basic parts: Model, View, and Controller.

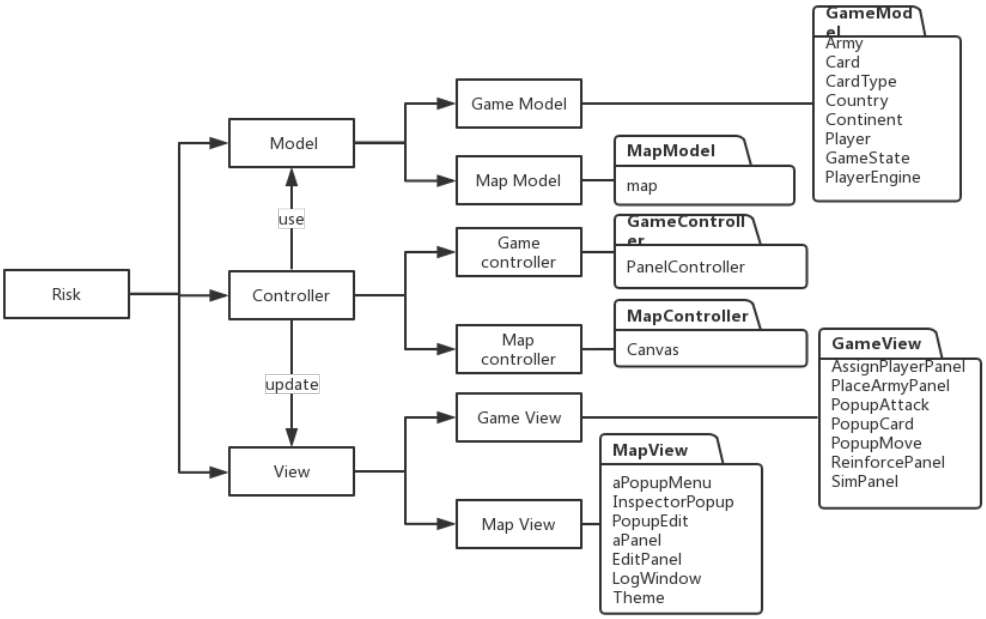
In our first version, we divided all classes into Model, View, and Controller. Since the function increasing, we decided to divide the Model package into Map Model and Game Model; Controller package into Map controller and Game Controller; View package into Map view and Game view.

In this way, make our test and debug easier.

Architecture design in version1 (before refactoring):



Architecture design in version2 (After refactoring):



2. Class refactoring

1. Implementation of the reinforcement, attack and fortification as methods of the Play View class.
2. Implementation of "player world domination view" using Observer pattern.
3. In our version2, we refractory map class and canvas class. Put all function related to front-end operations into canvas and map is mainly the class deal with user's actions.
4. We also divided reinforcement state into two part: choose card and reinforcement comparing put these two function together in version1. This is because both choose card and reinforcement need to calculate bonus army number. If we put them together, it is hard to debug which part get the wrong army number.

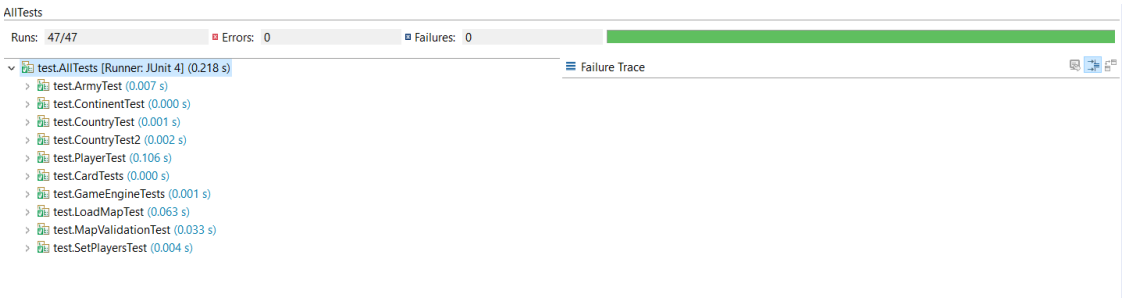
3. method refactoring

1. Combining similar methods into one method.
2. Rename Method.
3. Consolidate conditional Expression.

4.Test checking

we apply the same unit test cases in both two versions, all of them pass all test cases. We can assume we change the external architecture without change internal functions.

Test cases in version1:



Test cases in version2: