All Informations about this Refactoring

Information about the Original Project: The Original Project consists of a really simple game where the goal is to survive while small walls try to crush it against the ceiling. **To run it** just download the following repository and run the program in eclipse: https://github.com/Glank/Java-Games/tree/master/FallDown . Similarly, **to run the refactored project** just follow the same step, however it is in this repository: https://github.com/DavidAlexandreTS/FallDownRefactor .

Information about applied of the Design Patterns

1. **Singleton**
   The purpose of this standard is to ensure that a given class has only one instance, in the project in question it was applied to the Ball class, thus avoiding duplication or excessive calls and ensuring security against bugs or duplication errors. The getInstance method is always called in place of the constructor that previously existed there (as public), which increases the performance of the Program and leaves the code much more elegant.

2. **Memento**
   Similar to a "Payroll" (Undo / Redo) project task of this discipline, the purpose of this pattern is to protect the state of an object that can be retrieved at a desired time. The application of this standard is not complete, as it should happen, the changes would alter the operation of the program and the teacher asked to maintain the operation. In any case the application of Memento here is very cool, imagine being defeated by the game, and have your points recovered. However, as the teacher asked not to make any big changes, the button to do Undo was not added, the whole code is there and can be seen through the back () and goin () methods, the classes were also created Memento and Save, which are the application of this Design Pattern.

3. **Builder**
   Builder as a standard ensures that your objects are built in a simpler and more elegant way. It was applied to this project directly in the Brick Class (which later was called Scenario). The constructor of this class has been changed to private as the default requires and access is given through the Builder class which is a component within the Scenario class itself.

Relevant Changes

1. The Brick class became Scenario for that the accumulated Builder I decided to rename as something more generic.
2. No Program Functionality has changed.
3. The application of Memento is cool, however I have left it out of practice and only implemented all its basic structure (because its addition needs a button and some relevant changes in the features of the program, which was requested not to occur).
4. The Color of some components in this game has changed.