

# Sprint 0: Design Patterns

**Team: Java Doctors**

**Team Members:**

**Alexander Thomas Hills Shea**

**Zoya Hassan**

**Jonathan Choi**

**Safa Al-Siaudi**

Mentor: Manav Bhojak  
CSC207H5: Software Design

# Table of Contents

<b>Observer</b>	<b>3</b>
<b>Factory</b>	<b>4</b>
<b>Composite</b>	<b>5</b>
<b>Singleton</b>	<b>6</b>
<b>Strategy</b>	<b>7</b>
<b>Memento</b>	<b>8</b>

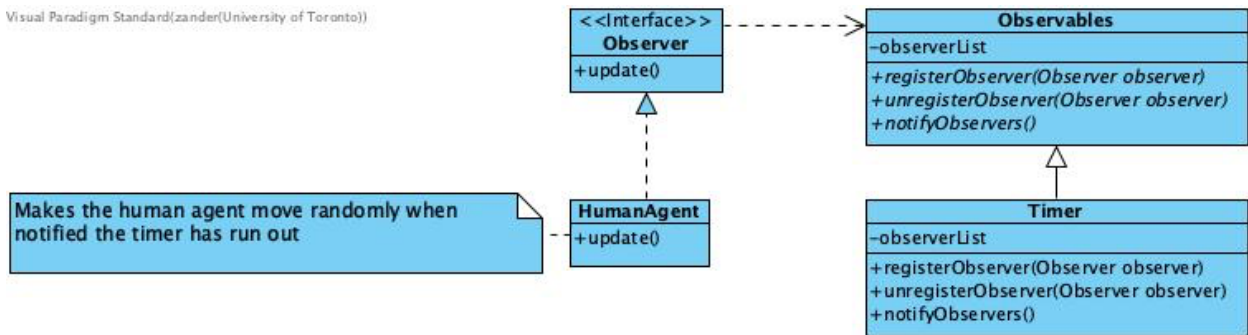
# Observer

## Usage

The Human player is given an allocated time to make their move; if they do not, the HumanAgent is notified and a move is made for them. This feature uses the basic Observables/Observer class and interface while the human agent will be given a timer. Timers are not given to non-human agents since they may take some time and ruin their functionality.

## UML Diagram

Visual Paradigm Standard(zander(University of Toronto))

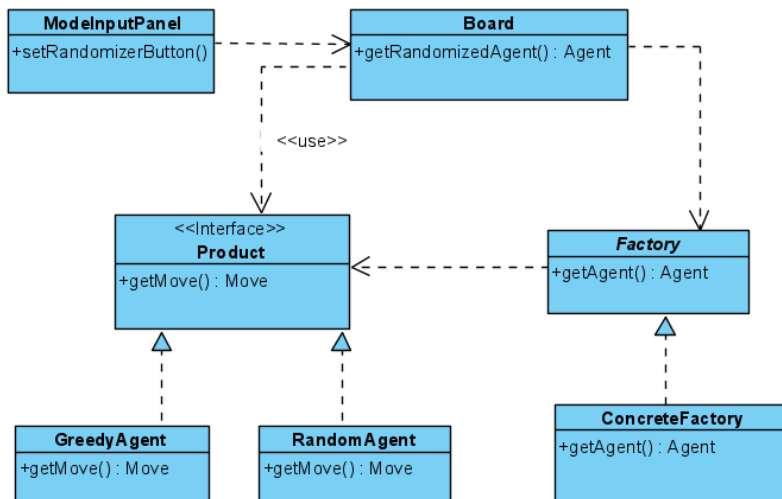


# Factory

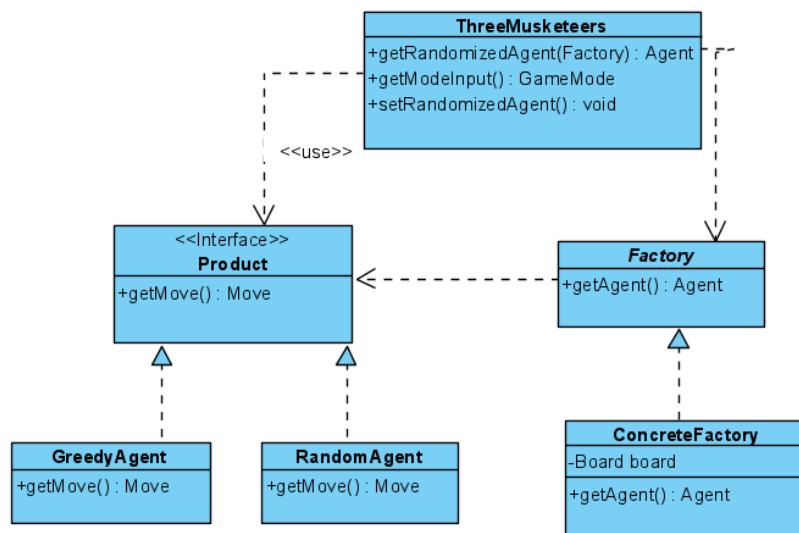
## Usage

Set up the game state when the player selects modes, with the proper agents playing each side. This will also facilitate the implementation of a random mode feature, in which the human player is assigned to a random side opposing a random non-human agent.

## UML Diagram



### FACTORY DESIGN PATTERN V2

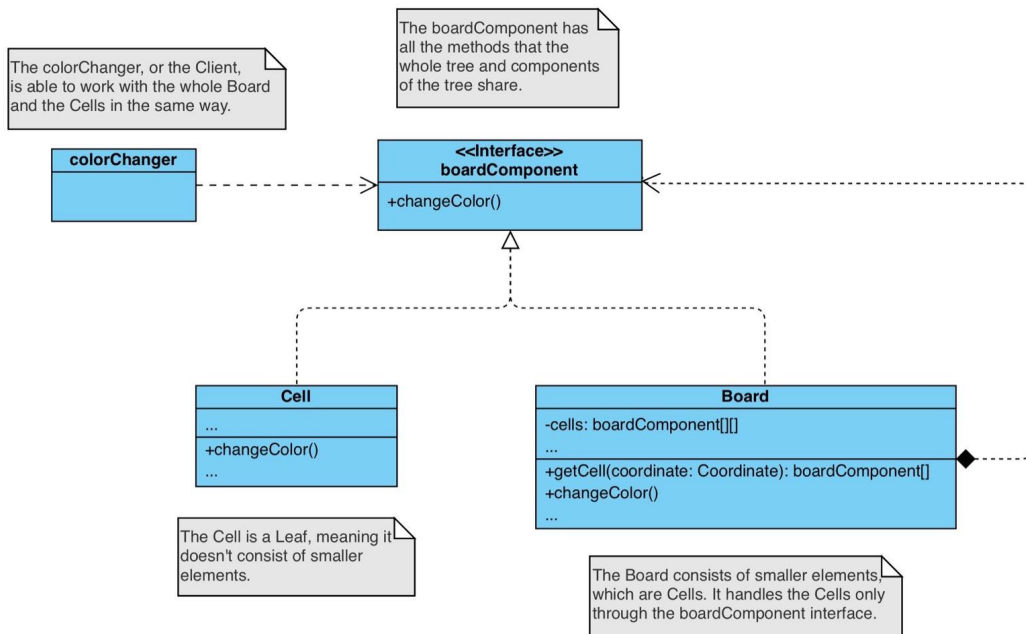


# Composite

## Usage

After each turn, the board and the cells on the board change color depending on whether it is a Guard or Musketeer turn. The boardComponent contains the methods that are shared between the Board (the whole tree) and the Cells (the components of the tree).

## UML Diagram

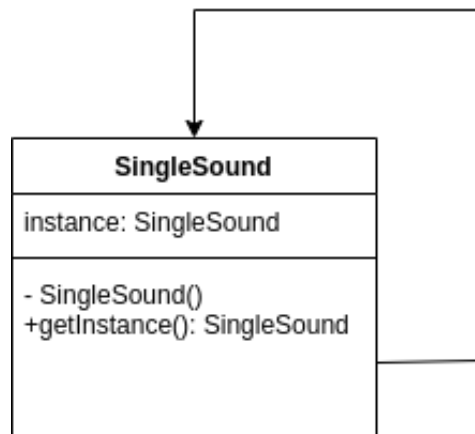


# Singleton

## Usage

Using this design pattern, users will have a more immersive experience while playing the Three Musketeers. Here, users will hear a sound as their piece moves, which resembles the experience of playing in real life. We will implement this by having a singleton class for sound effects and we will have a method that other classes can call to use the sound effect at a given point in the game.

## UML Diagram

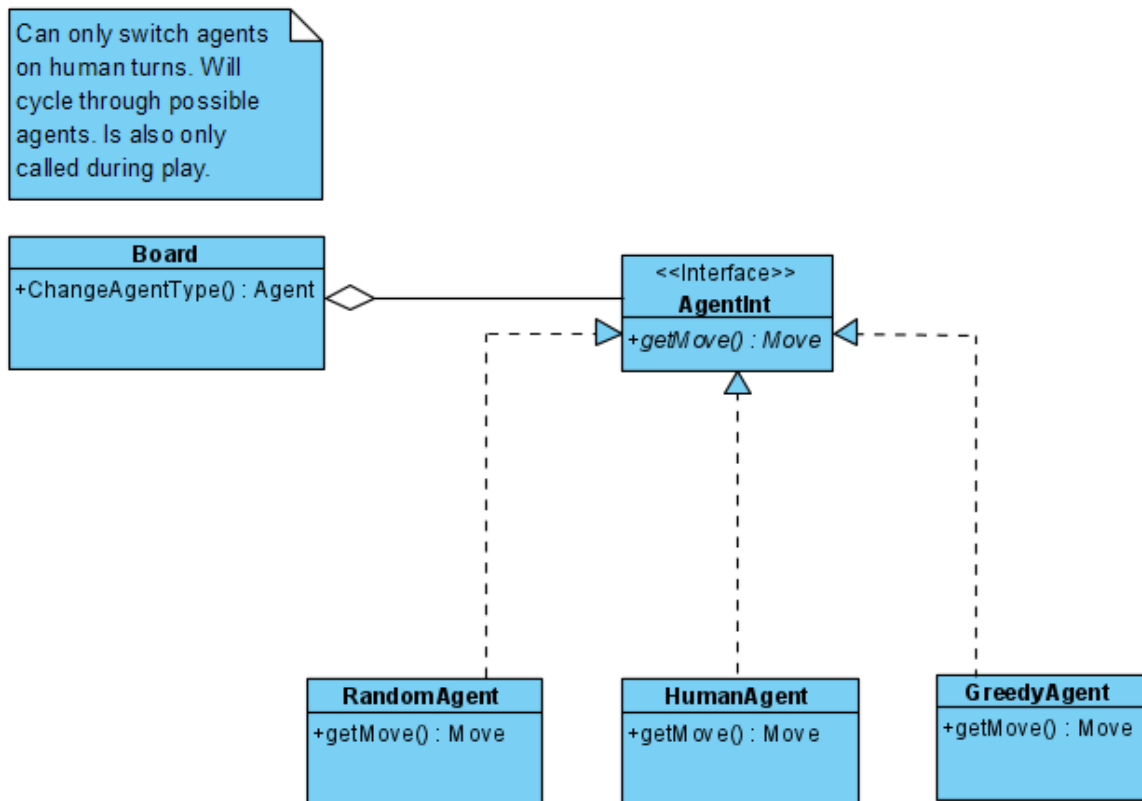


# Strategy

## Usage

The strategy pattern will be used to allow the human player to switch the game mode mid-game. A “switch” button will be implemented in the GUI, which can only be pressed during human turns and will not end the turn. There must always be at least one human player.

## UML Diagram



# Memento

## Usage

To create a memento of the game state before undoing a move so that the game state can be stored and then reconstructed when the user redoes a move. The board will be the originator, a new BoardMemento class will store the game state, and the main game class ThreeMusketees can be the caretaker. This solves the problem of game states being lost forever when a move is undone, allowing the implementation of a redo feature.

## UML Diagram

