

Sprint 2 - Design Patterns

Thomas Kolman

Razi Messinger

Nasim Bondar Sahebi

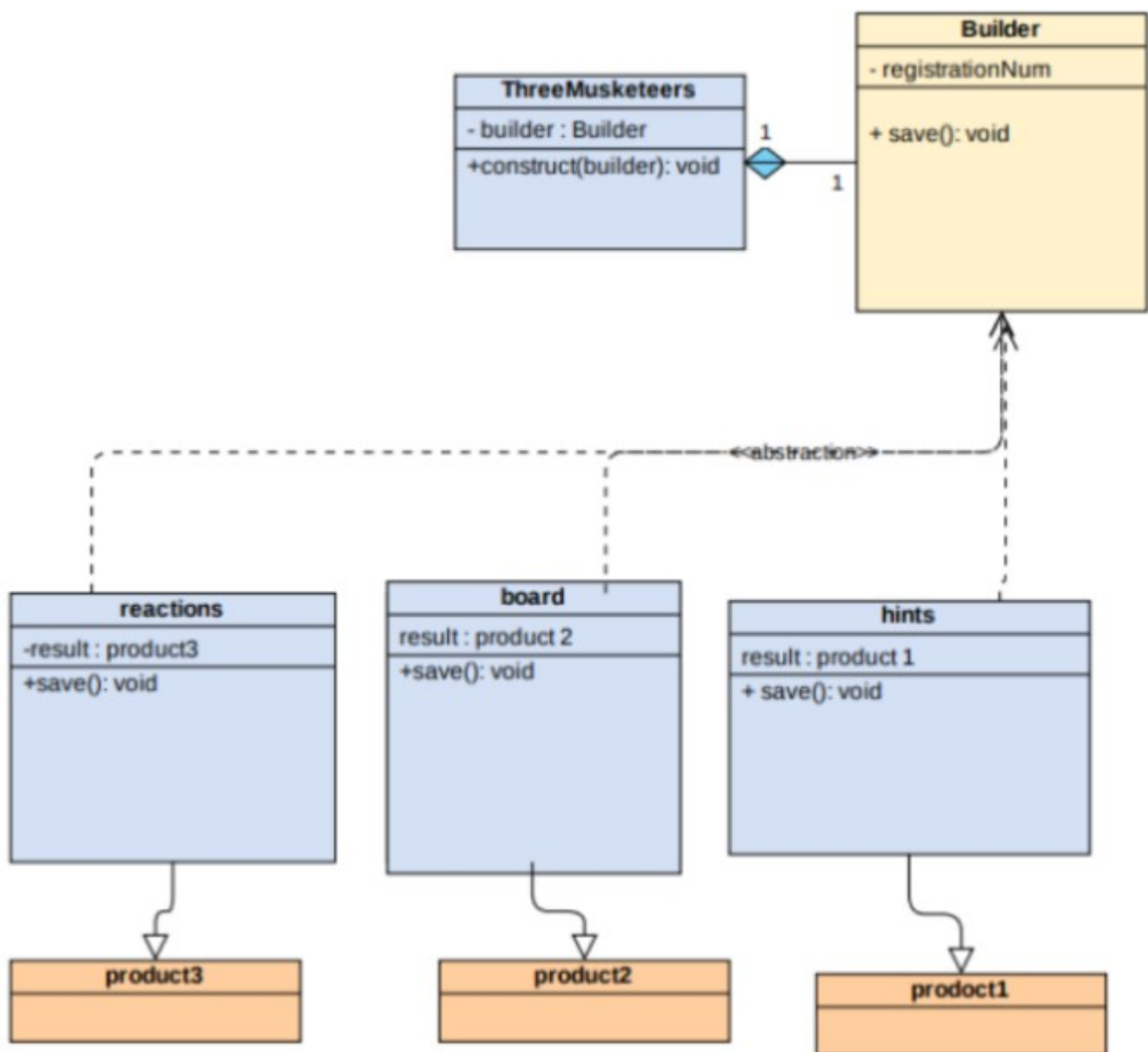
Justin Zhang

Table of contents

- 1) Builder pattern
- 2) Observer pattern
- 3) Factory pattern
- 4) Adapter pattern
- 5) Strategy pattern
- 6) Singleton pattern

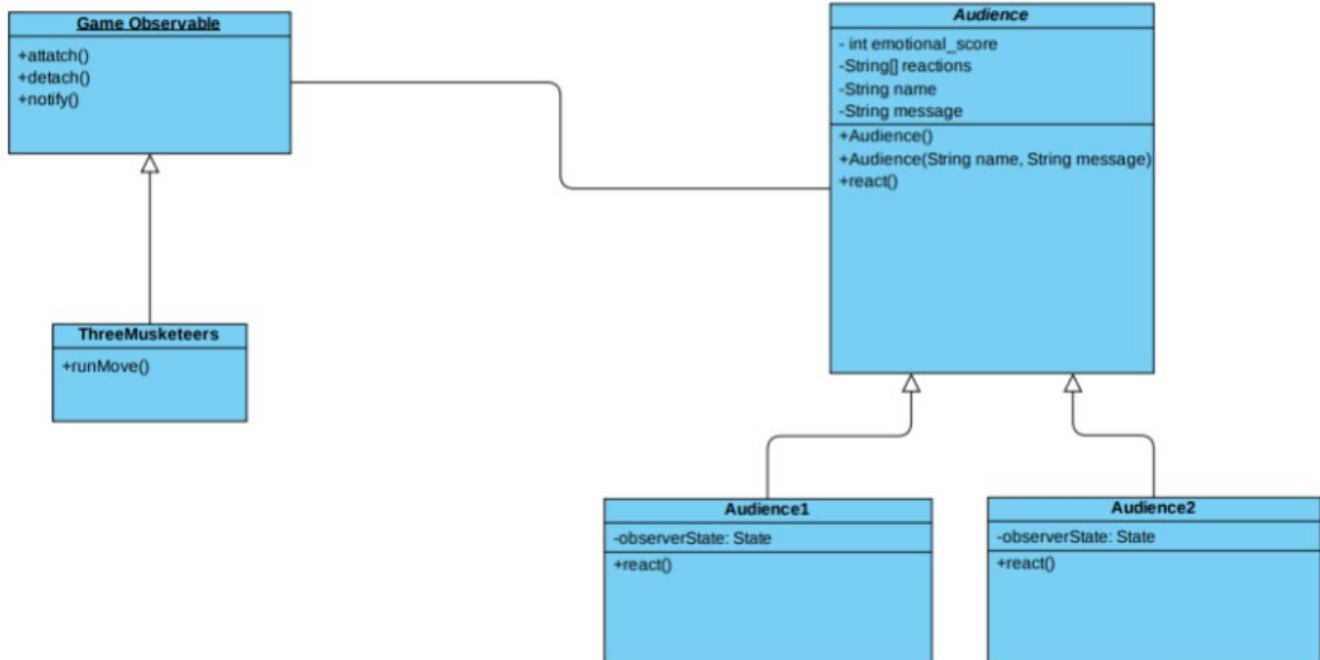
1) Builder pattern

The builder pattern defines an instance for the object but lets subclasses decide which class to start. It also refers to the newly created object through a common interface. To save the game, we use the builder interface and three classes of hints and reaction and board, which is implemented by the builder. Each of the three classes share the save() method inside.

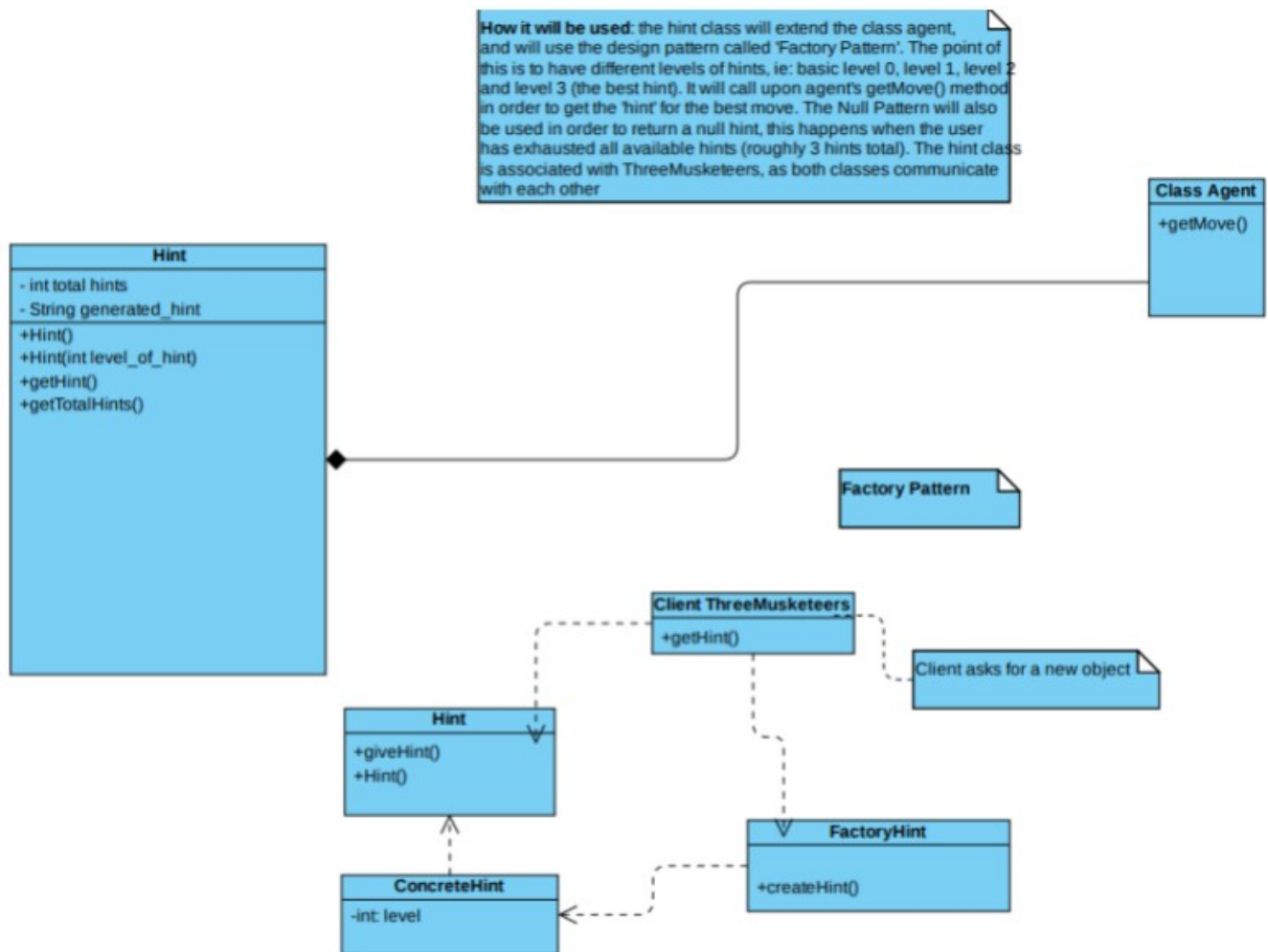


2) Observer pattern

How it will be used: The audience class will utilize the Observer Pattern.. The audience is the observer, and it is observing the class Game Observable. Audience communicates with class ThreeMusketeers as it utilizes the react() Method which will have a reaction based on moves performed as the game is played. Class ThreeMusketeers, which extends class Game Observable, will notify all observers (in this case, the class audience which has many different members in the audience, ie: audience1, audience2, audience3.

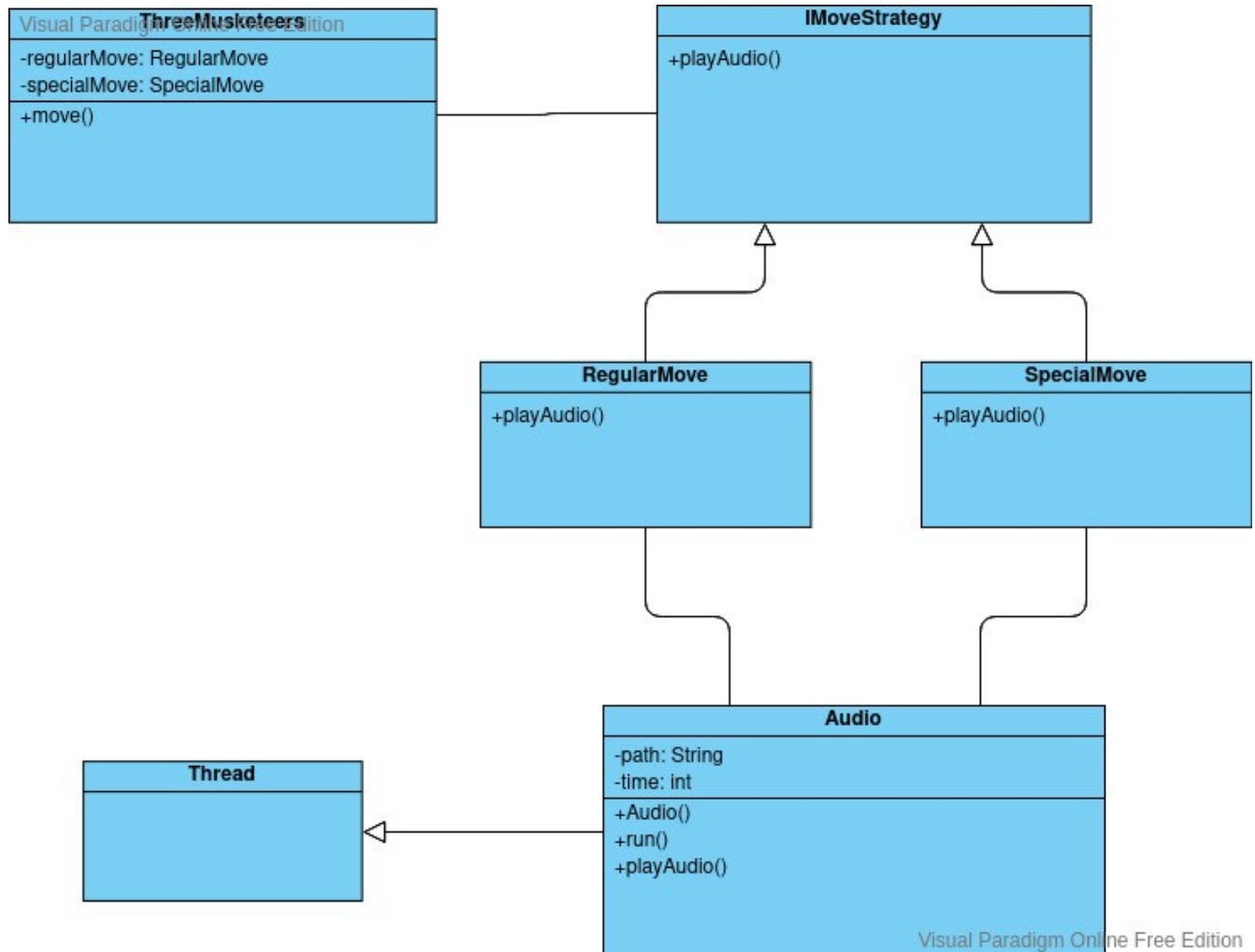


3) Factory Pattern



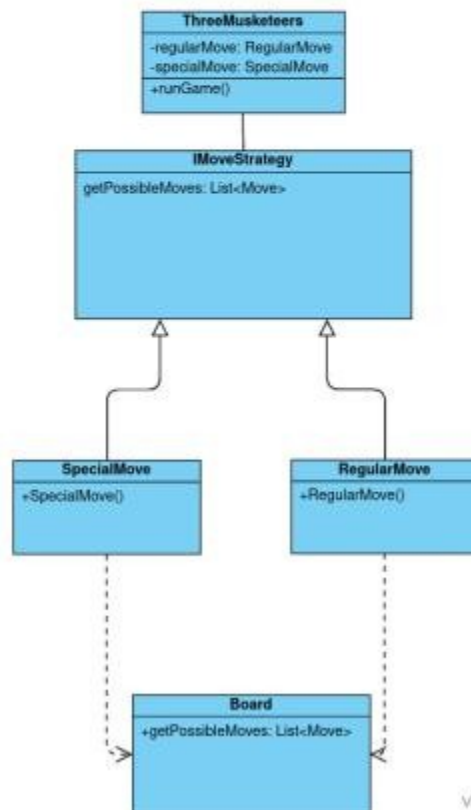
4) Adapter Patterns

The `IMoveStrategy` defines an interface for the `Regular` and `Special` move classes, which each inherit the `playAudio` method. The `move()` method in `ThreeMusketeers` accesses the `IMoveStrategy` objects, calls the `playAudio()` method, which in turn calls the `Audio` method.



5) Strategy Pattern

How it's used: The special move class uses the command pattern. It associates itself with the ThreeMusketeers class. There are many kinds of moves, specifically: RegularMove and SpecialMove which are subclasses of move. They are special kinds of moves like 'killing 2 pieces / moving twice'. RegularMove is just 1 move, ie: for guard it is moving a guard once, and for musketeer it is killing a guard.



Visual Paradigm Online Free Edition

6) Singleton pattern

How it's used: These classes use the singleton design pattern. The singletons are the RandomAgent, HumanAgent and GreedyAgent. Agent is an abstract class, and Greedy and Random and Human Agent implement Agent. One instance of each object (everything is static), there will be only one object of each agent. Run game essentially takes all the agent's and utilizes them. The point of this is to be able to switch agent's mid game (just like how you could state which agent at the start of the game, the user is also able to change agents mid-way through).

