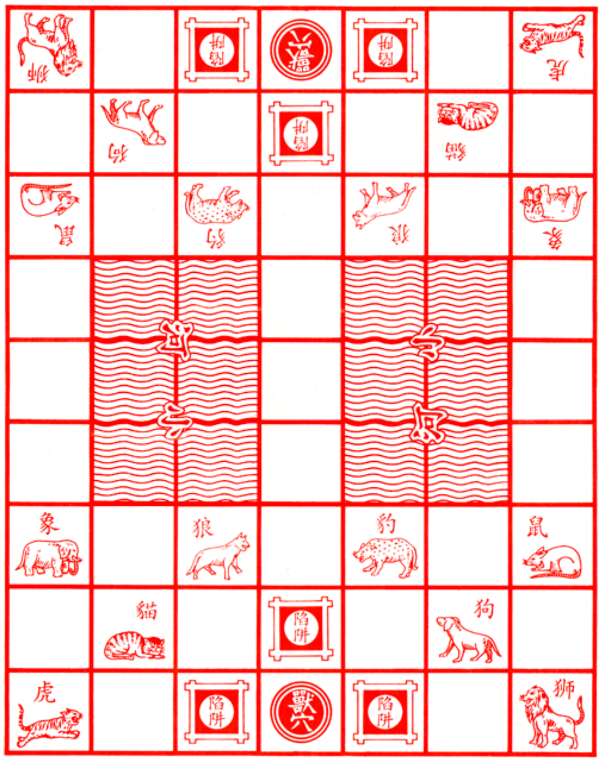

# Jungle

Refactoring and Design Pattern List

## Refactoring and Design Pattern List

During the life of our project we have done our best to keep a constant eye on code smell and pattern usage, from the beginning. However some of the most important or common refactoring done is below. The basic list of patterns we have used to this point, including changes across the iterations is also listed below.

### *Design Patterns*

- Observer
- Factory/Abstract Factory
- Model-View-Controller
- Singleton
- Factory Method
- Abstract Factory Method
- Prototype
- Facade
- Decorator

### *Refactoring:*

- Replaced branching game logic implementation with polymorphism.
- Our team spent time updating the original project structure to make sure that each of the three core parts (UI, Server, and Business Logic) had a clear structure and interface providing interaction between them.
- We reduced coupling in these internal parts be forming a clear hierarchy to the structure. (e.g. in the business logic squares control pieces, lowering the level of coupling for the gameboard)

- ➢ We spent quite some time double checking how all our code was structured so we could add in patterns (such as factory) to help provide direct and easily managed code.
- ➢ A large amount of our potential refactoring is often prevented by our team's focus on clear design and pull management. The group as a whole has done a good job of managing that each small section of code pulled into the master branch is clear and straightforward. We have sought out the extra time to check if shortcuts are taken at the time of creation rather than when errors arise. In this regard most refactoring is either small changes at the time of creation which encourage the use of strong code habits and naming conventions or our refactoring is large scale restructuring of the methods and classes to ensure fast and clear usage, as covered above.