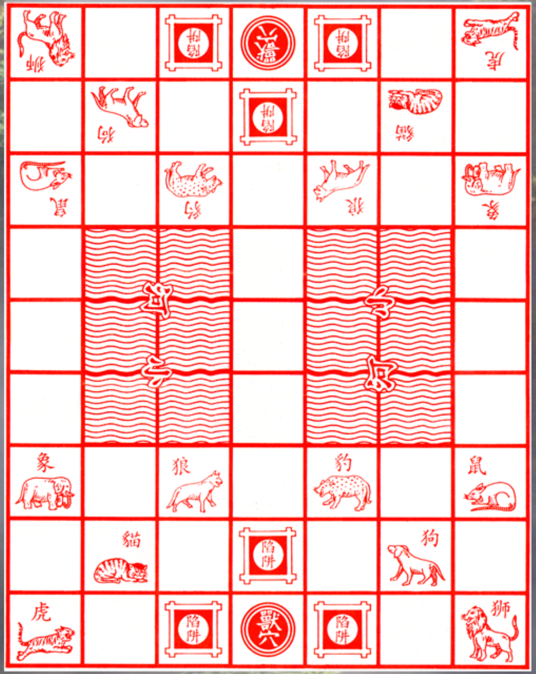
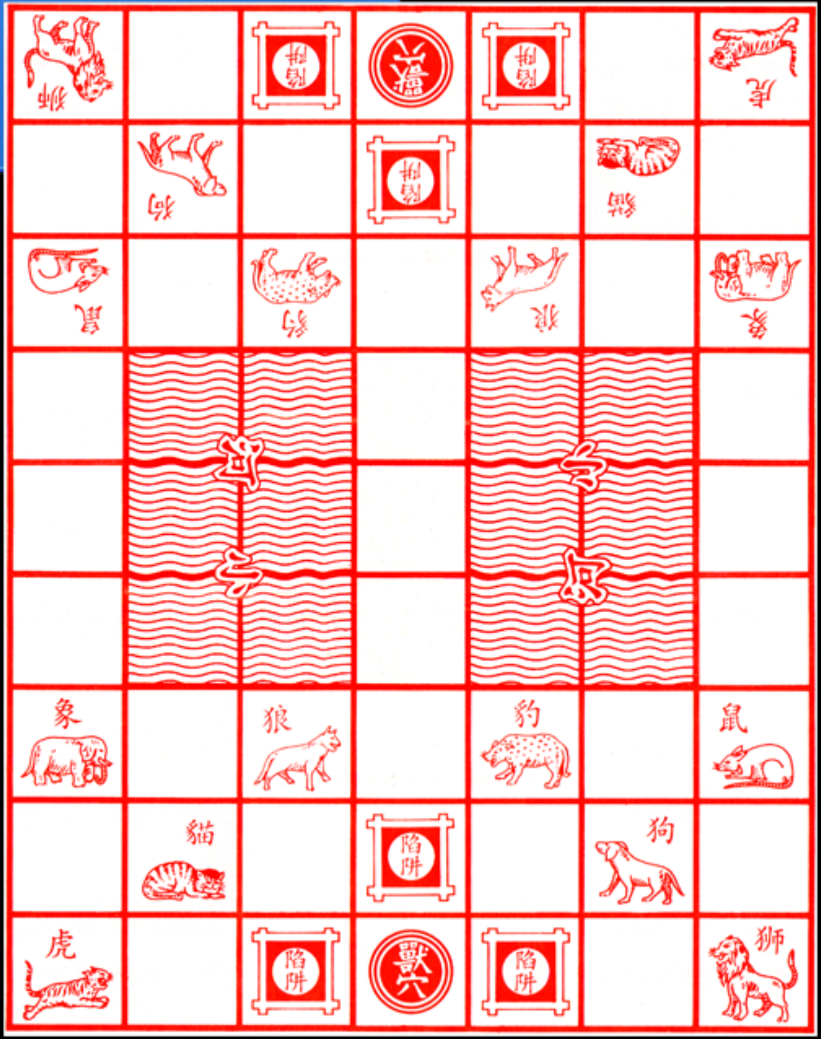
Jungle



Method Men

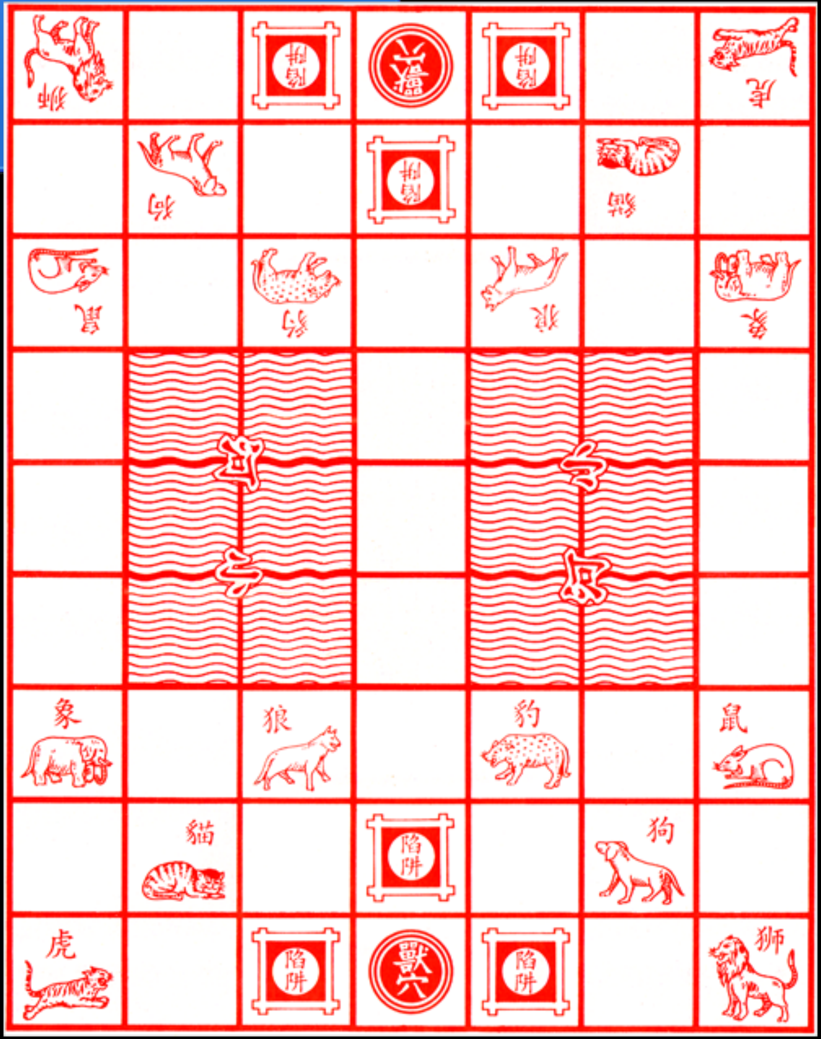
Game Overview

- Two ways to win:
  - Move a piece into the opponent's Den
  - Capture all the opponent's pieces

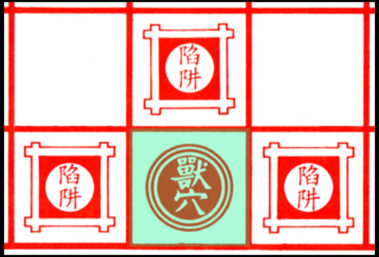


Pieces

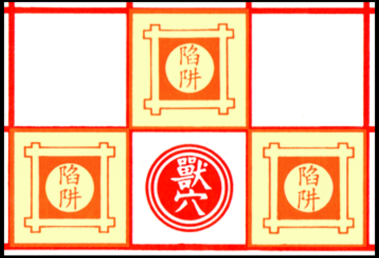
- 8 pieces, different animals and different ranks
- Higher rank = stronger piece
  - Elephant - 8
  - Lion - 7
  - Tiger - 6
  - Leopard - 5
  - Dog - 4
  - Wolf - 3
  - Cat - 2
  - Rat - 1



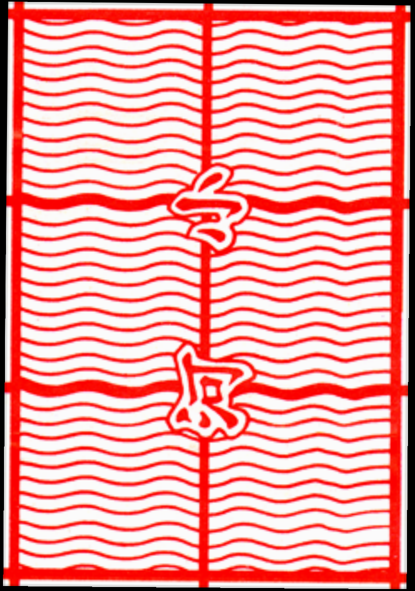
Board



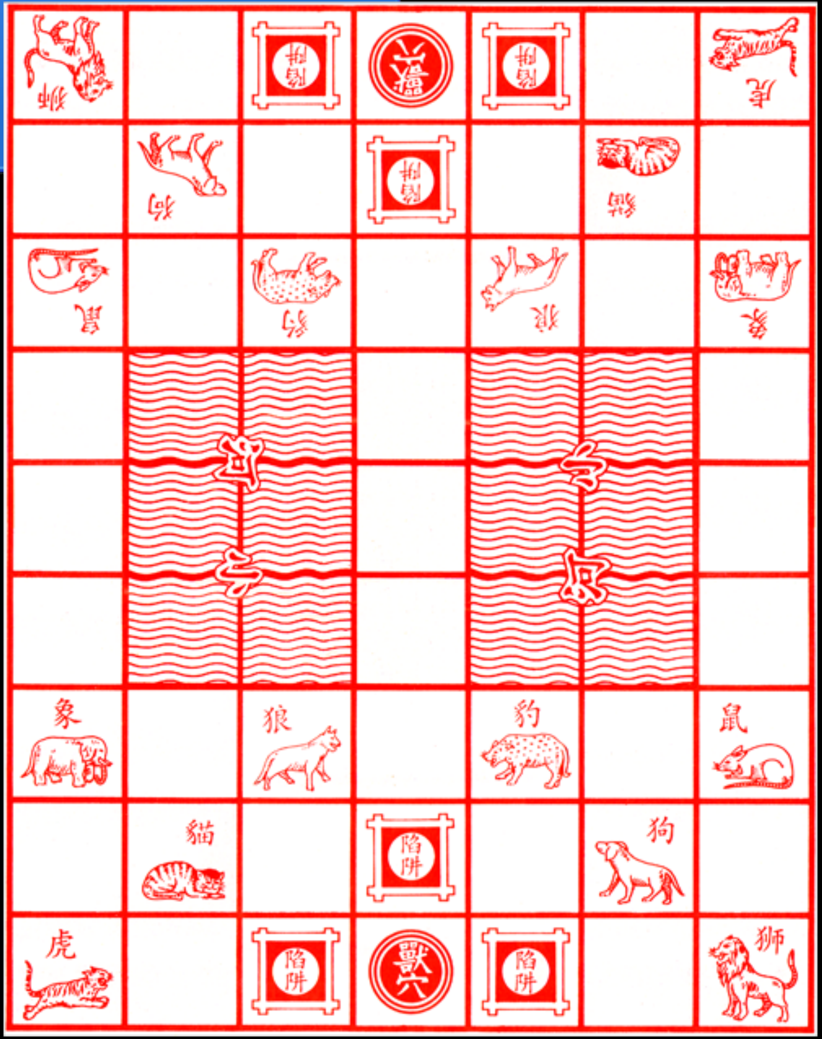
Den



Traps

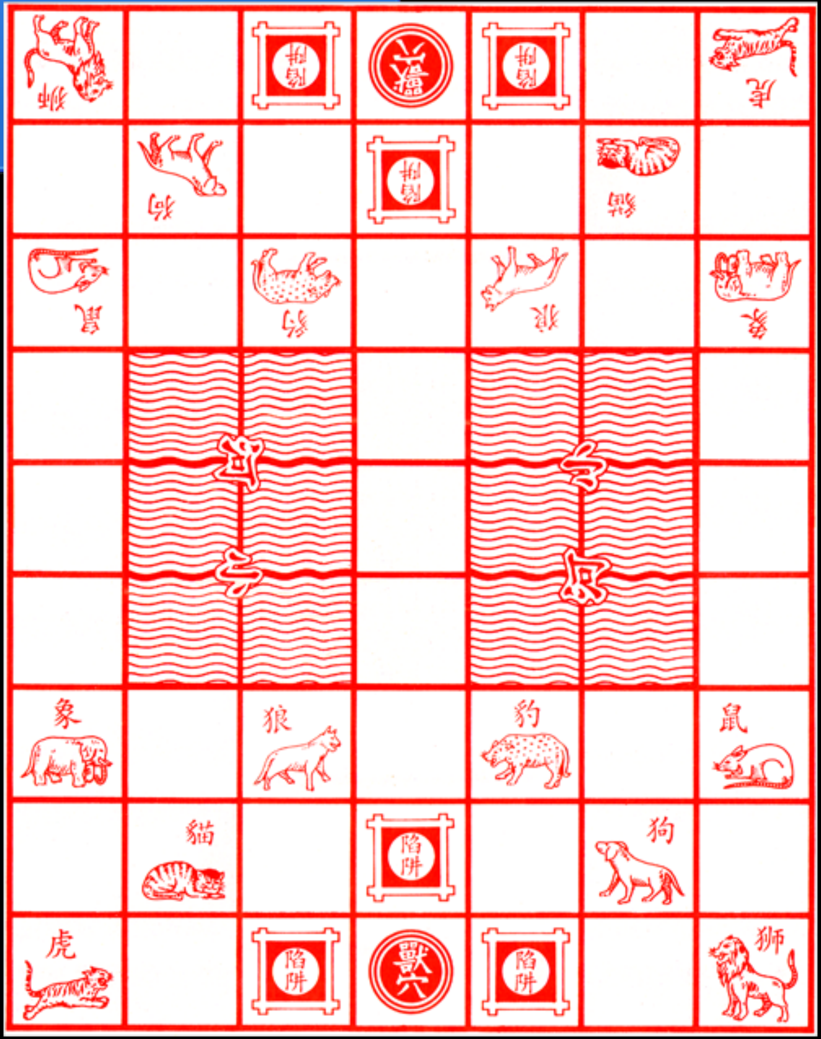


River



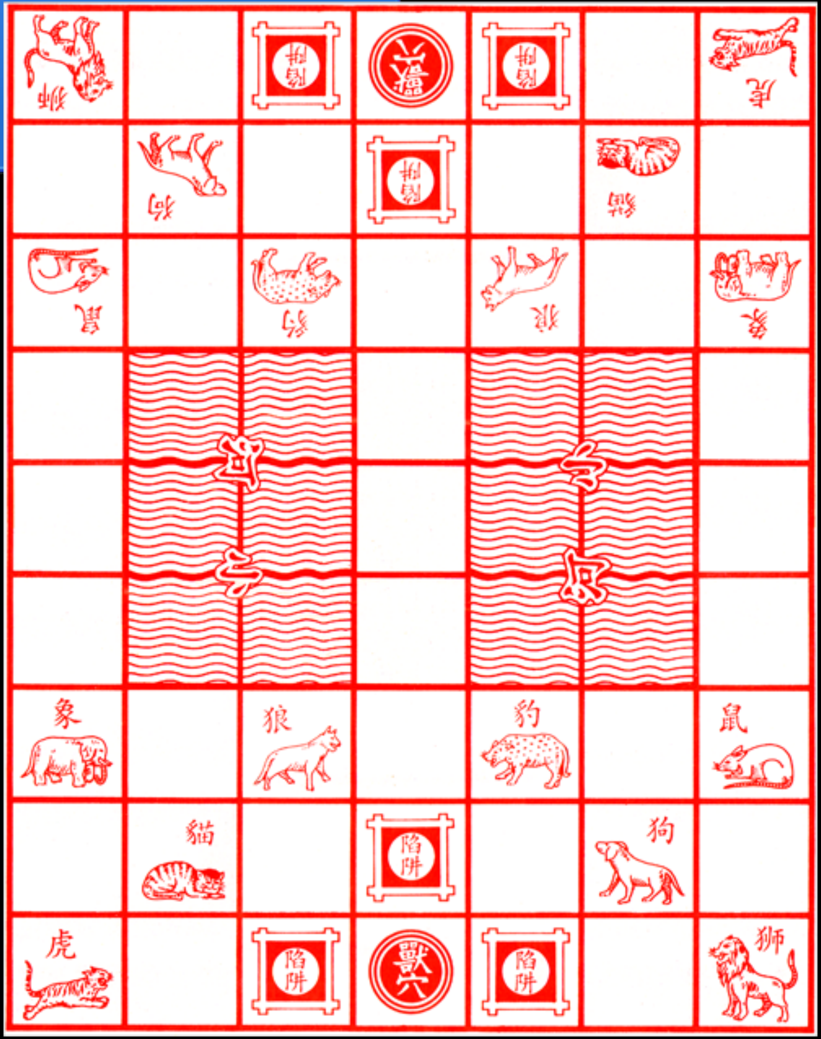
Movement

- 1 tile horizontally or vertically
- *Cannot move diagonally*
- Trap reduces rank to 0 - piece can be captured by any opponent piece
- Only Rat can enter water
- Overtake an equal or lower-valued enemy to capture



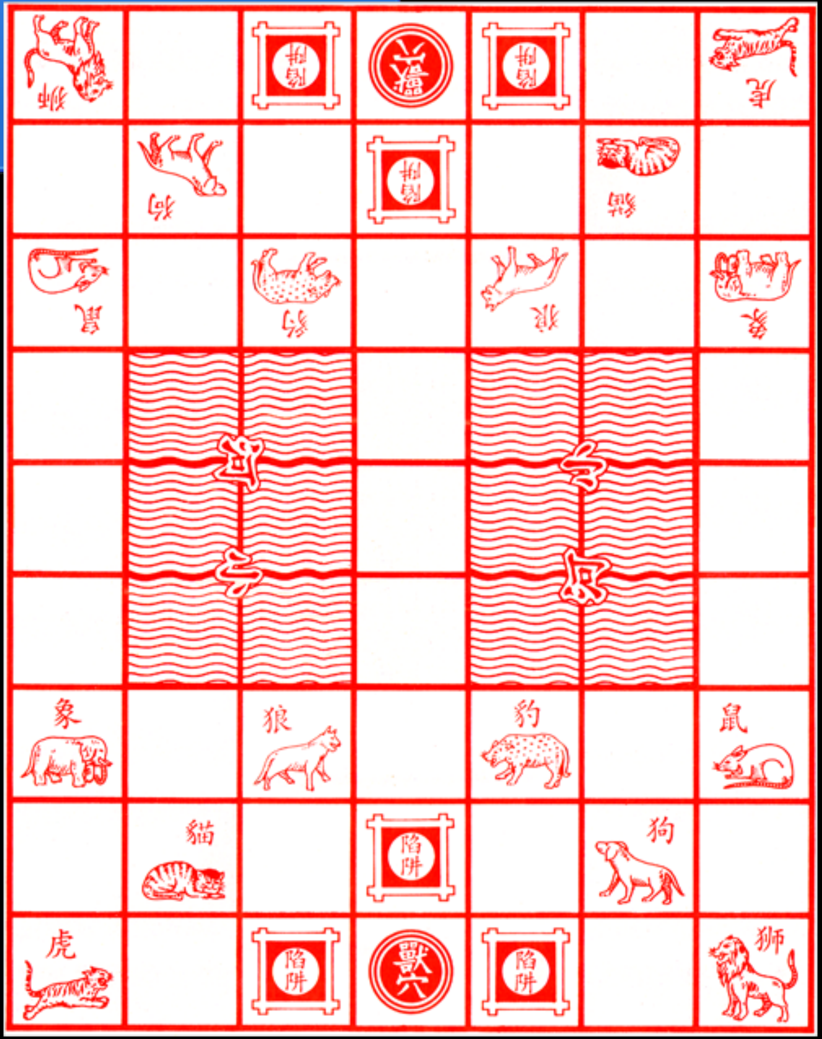
Water Tiles

- Only Rat can enter the water
  - Can swim around the water freely
  - *Cannot attack from water*
- Lion & Tiger can jump over water
  - May jump
    - Horizontally (3 tiles)
    - Vertically (4 tiles)
    - Jump is blocked if Rat is in path



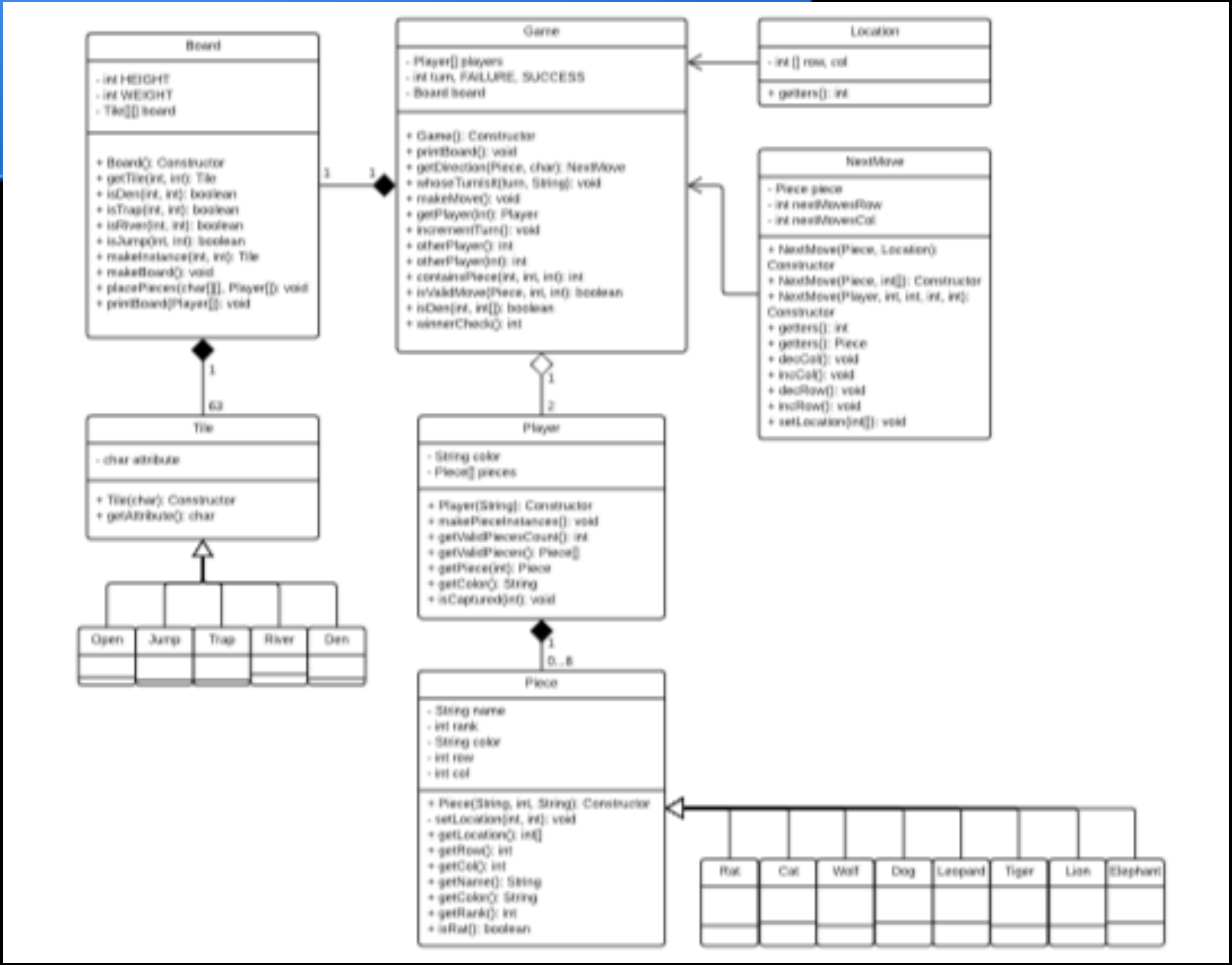
Specials

- Rat can capture Elephant, Elephant cannot capture Rat
- When a piece moves into an opponent trap, rank is reduced to zero and any enemy piece can capture
- Pieces can move into own traps, does not effect rank
- Cannot move into own Den



Class Diagram

- P3 version
- Refactored our game code

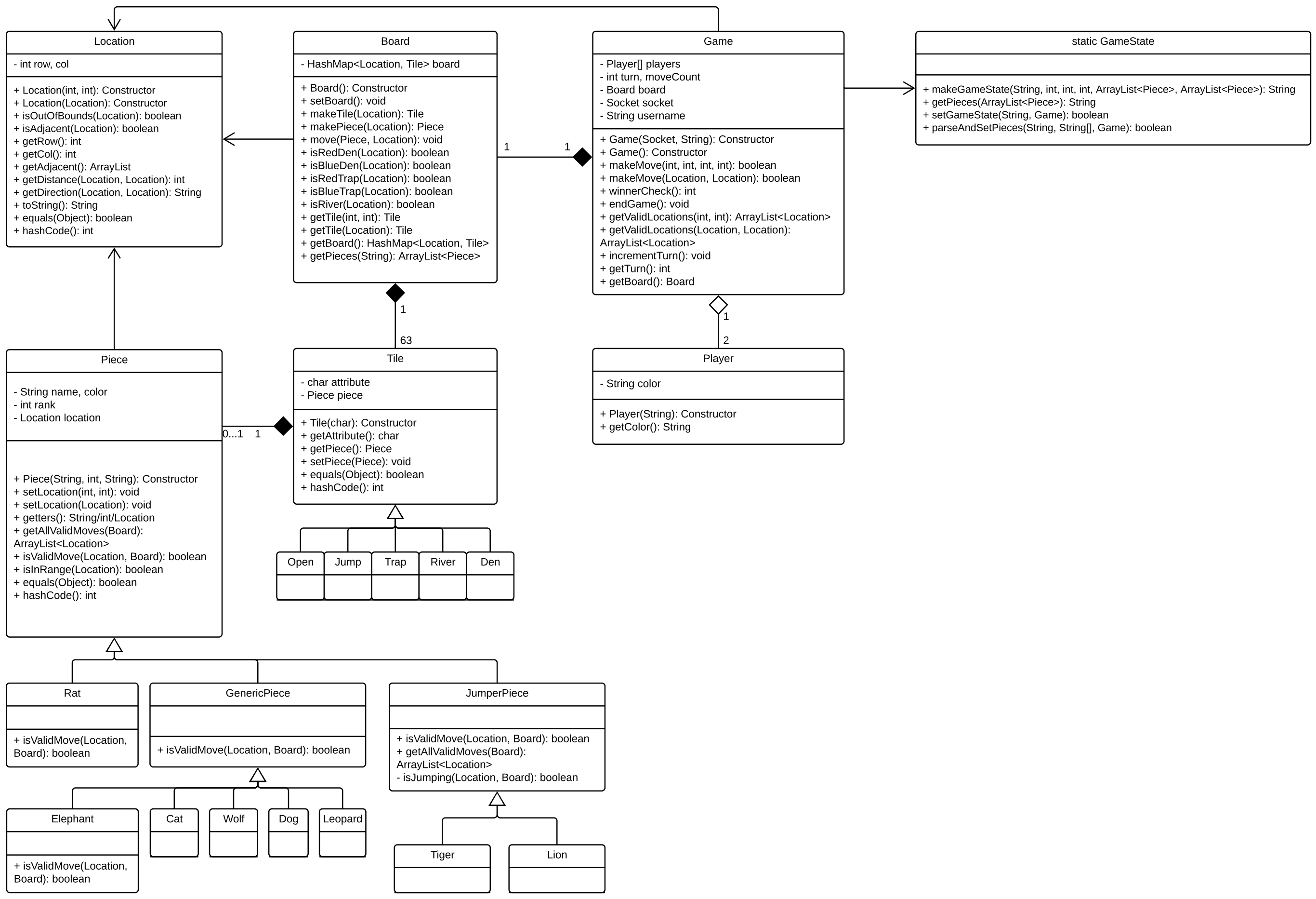


Refactoring

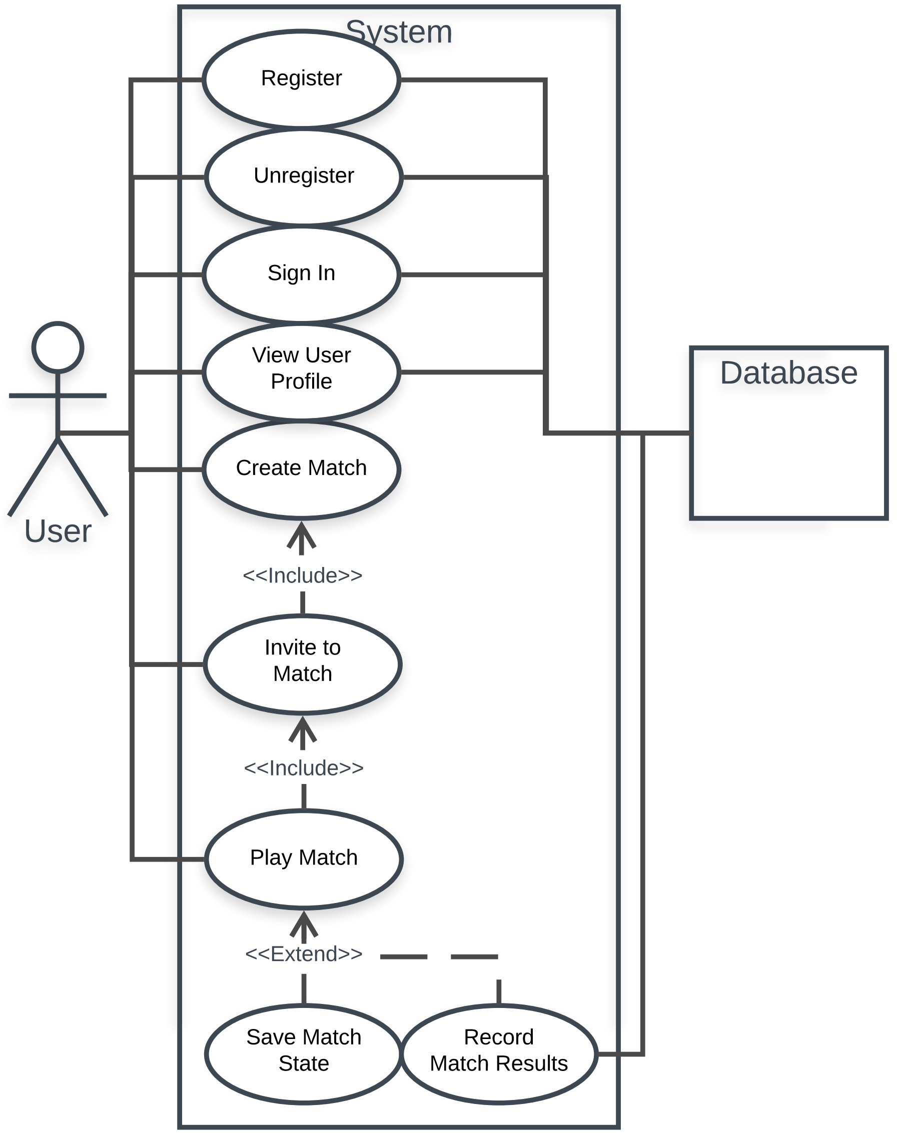
- Made Piece have subclasses: GenericPiece, Jumper piece that are extended by relevant pieces
- Move validation handled on a piece-by-piece basis
- Moved Pieces to be held by Tile > held by Board > held by Game
- Board holds a HashMap<Location, Tile> rather than Tile[][]
- Board now creates a Tile based on location, and sets Pieces based on location.
- Reduced Game.java class by ~600 lines, similarly simplified Player.java and Board.java

Class Diagram

- Class diagram for game logic portion of project after refactoring


Location
- int row, col
+ Location(int, int): Constructor
+ Location(Location): Constructor
+ isOutOfBounds(Location): boolean
+ isAdjacent(Location): boolean
+ getRow(): int
+ getCol(): int
+ getAdjacent(): ArrayList
+ getDistance(Location, Location): int
+ getDirection(Location, Location): String
+ toString(): String
+ equals(Object): boolean
+ hashCode(): int
Board
- HashMap<Location, Tile> board
+ Board(): Constructor
+ setBoard(): void
+ makeTile(Location): Tile
+ makePiece(Location): Piece
+ move(Piece, Location): void
+ isRedDen(Location): boolean
+ isBlueDen(Location): boolean
+ isRedTrap(Location): boolean
+ isBlueTrap(Location): boolean
+ isRiver(Location): boolean
+ getTile(int, int): Tile
+ getTile(Location): Tile
+ getBoard(): HashMap<Location, Tile>
+ getPieces(String): ArrayList<Piece>
Game
- Player[] players
- int turn, moveCount
- Board board
- Socket socket
- String username
+ Game(Socket, String): Constructor
+ Game(): Constructor
+ makeMove(int, int, int, int): boolean
+ makeMove(Location, Location): boolean
+ winnerCheck(): int
+ endGame(): void
+ getValidLocations(int, int): ArrayList<Location>
+ getValidLocations(Location, Location): ArrayList<Location>
+ incrementTurn(): void
+ getTurn(): int
+ getBoard(): Board
static GameState
+ makeGameState(String, int, int, int, ArrayList<Piece>, ArrayList<Piece>): String
+ getPieces(ArrayList<Piece>): String
+ setGameState(String, Game): boolean
+ parseAndSetPieces(String, String[], Game): boolean
1
1
1
63
1
2
Piece
- String name, color
- int rank
- Location location
+ Piece(String, int, String): Constructor
+ setLocation(int, int): void
+ setLocation(Location): void
+ getters(): String/int/Location
+ getAllValidMoves(Board): ArrayList<Location>
+ isValidMove(Location, Board): boolean
+ isInRange(Location): boolean
+ equals(Object): boolean
+ hashCode(): int
0...1
1
Tile
- char attribute
- Piece piece
+ Tile(char): Constructor
+ getAttribute(): char
+ getPiece(): Piece
+ setPiece(Piece): void
+ equals(Object): boolean
+ hashCode(): int
Player
- String color
+ Player(String): Constructor
+ getColor(): String
Open
Jump
Trap
River
Den
Rat
+ isValidMove(Location, Board): boolean
GenericPiece
+ isValidMove(Location, Board): boolean
JumperPiece
+ isValidMove(Location, Board): boolean
+ getAllValidMoves(Board): ArrayList<Location>
- isJumping(Location, Board): boolean
Elephant
+ isValidMove(Location, Board): boolean
Cat
Wolf
Dog
Leopard
Tiger
Lion

Use Case Diagram



Domain Model/Glossary

Glossary:

Board: The 7x9 tile area on which the Game is played between two players.

Game: One Jungle game set up between two players.

Invite: Message sent from one Player to another Player as a request to play a game. Can be accepted or rejected.

Match history: A collection of match records for a specific player, which is located within their profile.

Match record: A record of the outcome of a specific match.

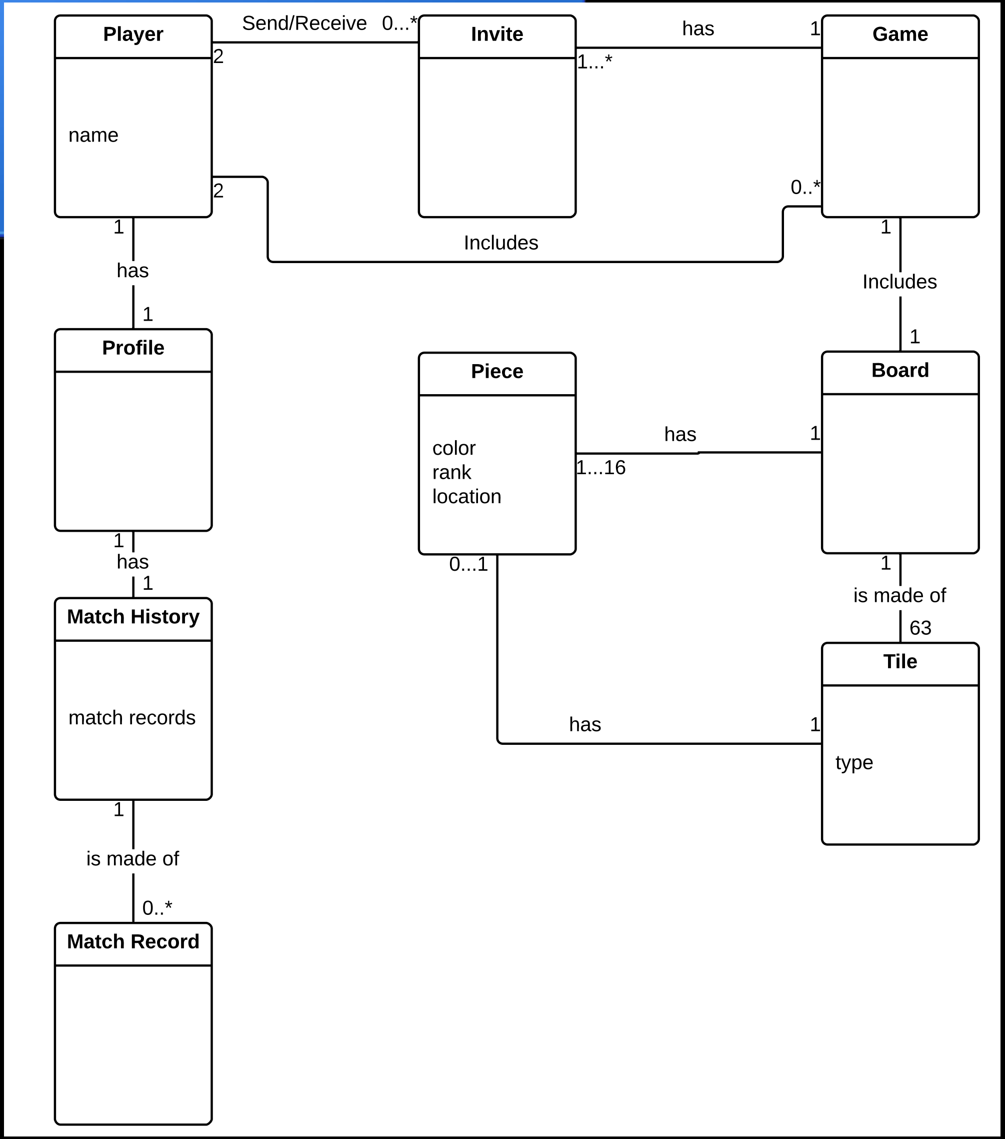
Player: Registered user of the system.

Piece: A player-controlled, movable object which has a color (associated with one player), rank (from 1 to 8), and location on the board.

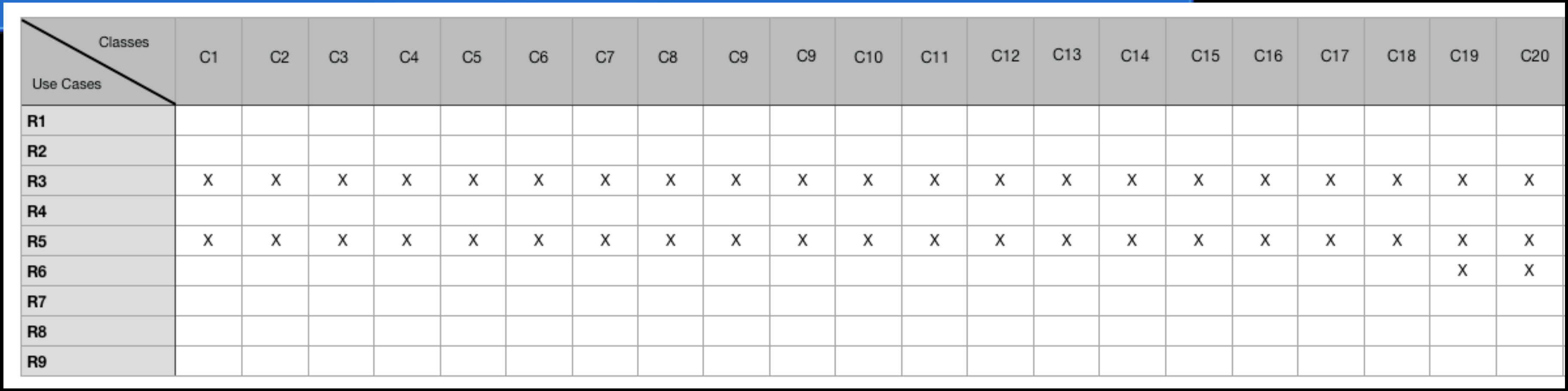
Profile: A virtual place for certain player information to be located and viewed.

Tile: A single square, of 63, on the board. Tiles can be of many types, and can hold pieces.

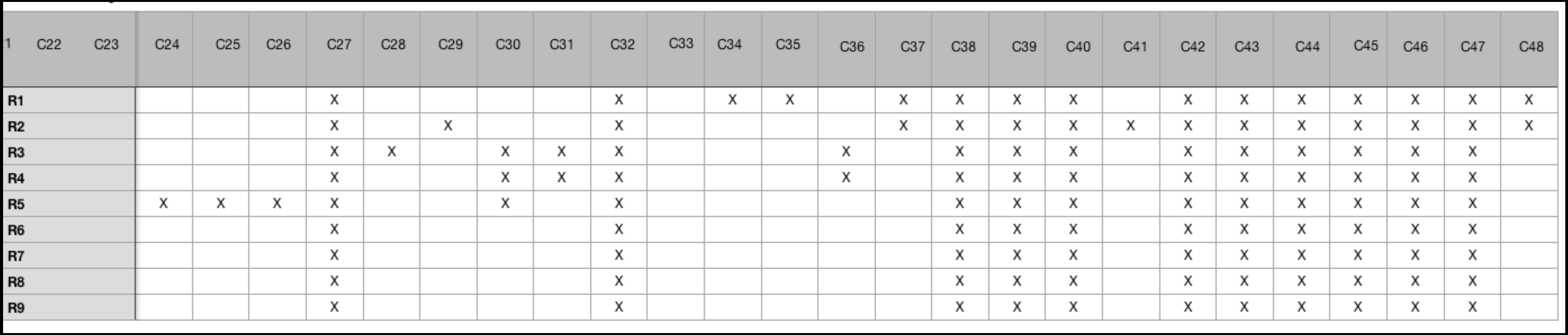
Tile types: Den, trap, river, basic. Consult the Jungle game rules for more information on implications of each type.



Traceability Matrix

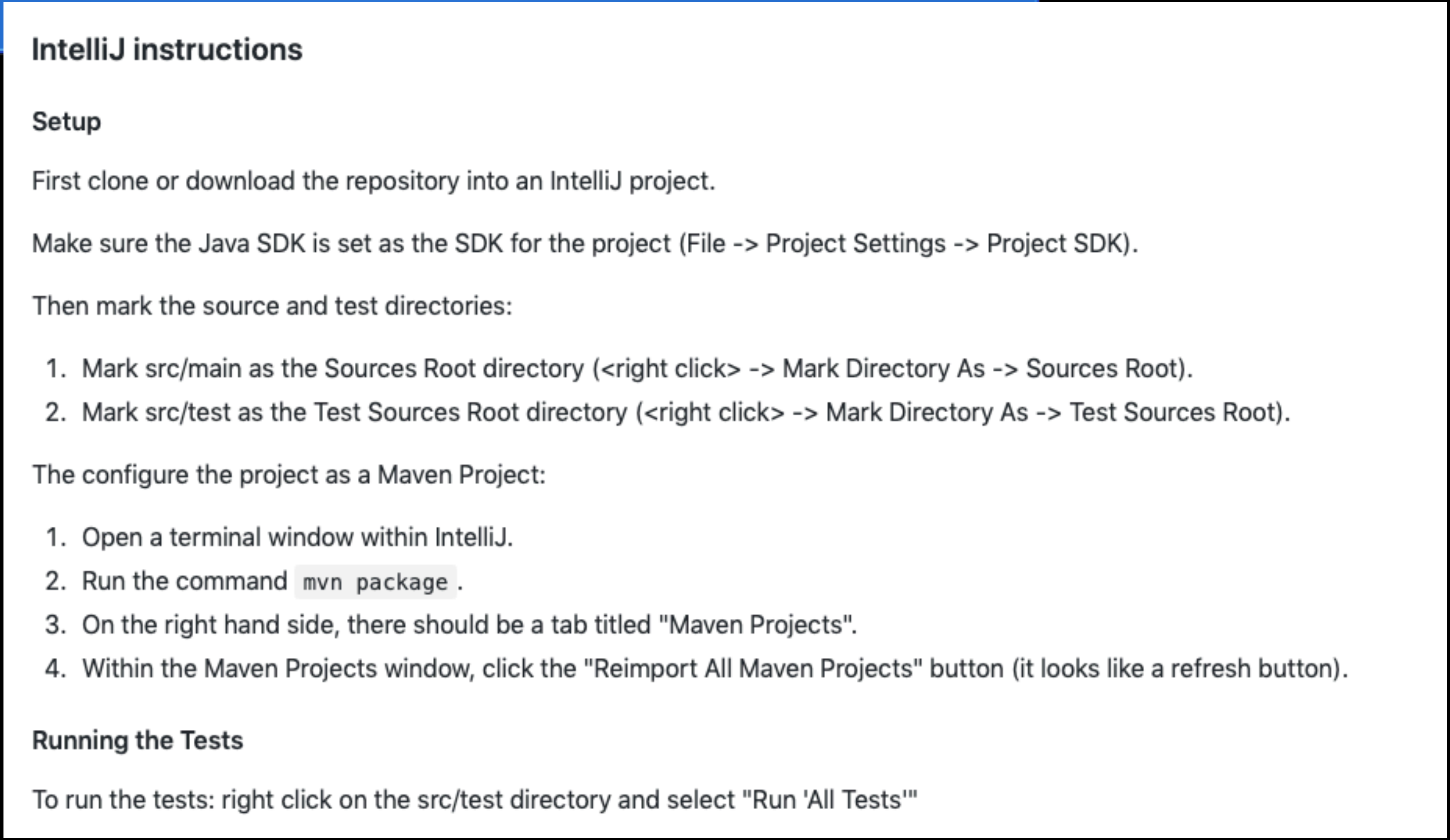


Classes / Use Cases	C1	C2	C3	C4	C5	C6	C7	C8	C9	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
R1																					
R2																					
R3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R4																					
R5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R6																				X	X
R7																					
R8																					
R9																					



1 C22 C23	C24	C25	C26	C27	C28	C29	C30	C31	C32	C33	C34	C35	C36	C37	C38	C39	C40	C41	C42	C43	C44	C45	C46	C47	C48
R1				X					X		X	X		X	X	X	X		X	X	X	X	X	X	X
R2				X		X			X					X	X	X	X	X	X	X	X	X	X	X	X
R3				X	X		X	X	X				X		X	X	X		X	X	X	X	X	X	
R4				X			X	X	X				X		X	X	X		X	X	X	X	X	X	
R5	X	X	X	X			X		X						X	X	X		X	X	X	X	X	X	
R6				X					X						X	X	X		X	X	X	X	X	X	
R7				X					X						X	X	X		X	X	X	X	X	X	
R8				X					X						X	X	X		X	X	X	X	X	X	
R9				X					X						X	X	X		X	X	X	X	X	X	

Development Manual



IntelliJ instructions

Setup

First clone or download the repository into an IntelliJ project.

Make sure the Java SDK is set as the SDK for the project (File -> Project Settings -> Project SDK).

Then mark the source and test directories:

1. Mark src/main as the Sources Root directory (<right click> -> Mark Directory As -> Sources Root).
2. Mark src/test as the Test Sources Root directory (<right click> -> Mark Directory As -> Test Sources Root).

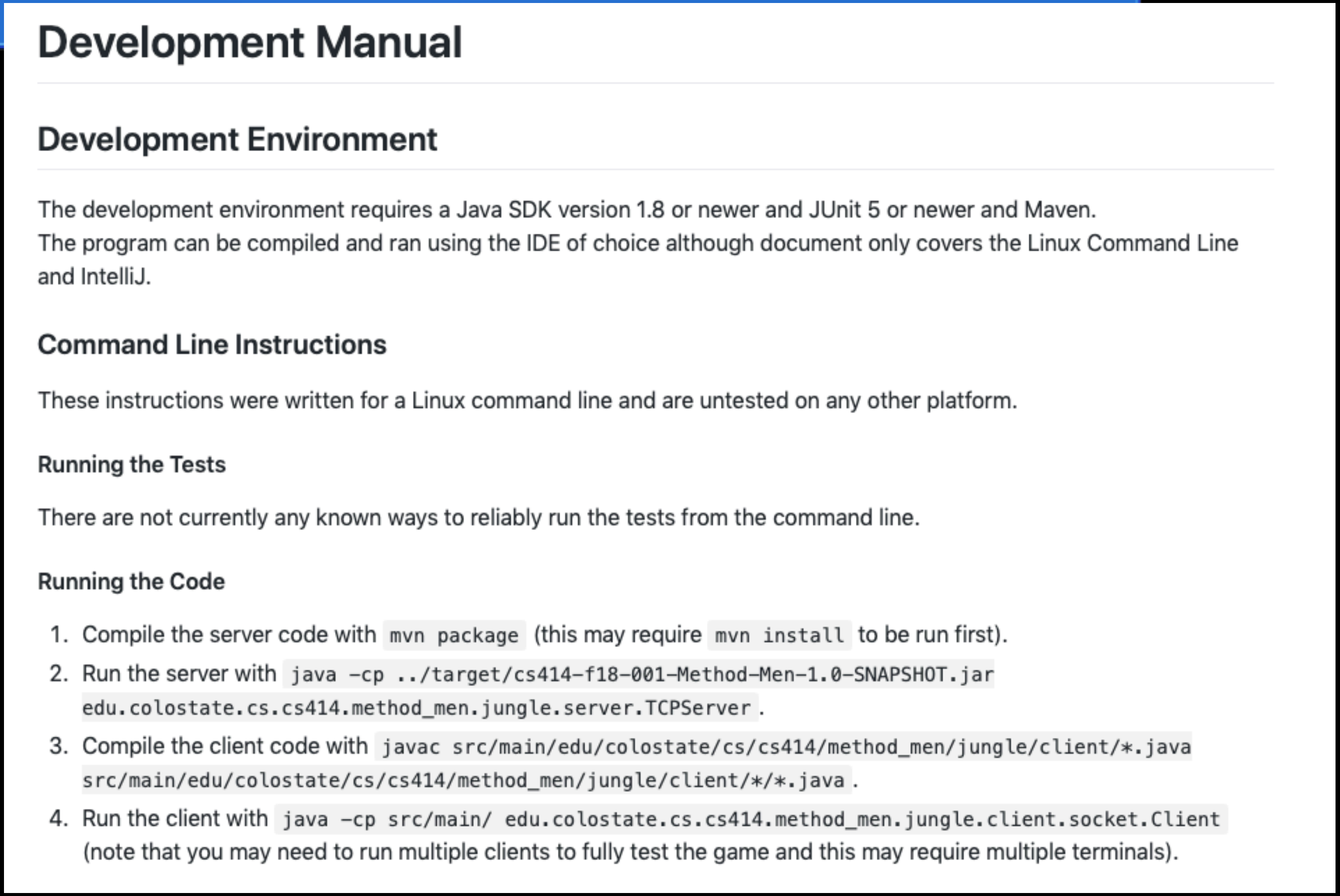
The configure the project as a Maven Project:

1. Open a terminal window within IntelliJ.
2. Run the command `mvn package`.
3. On the right hand side, there should be a tab titled "Maven Projects".
4. Within the Maven Projects window, click the "Reimport All Maven Projects" button (it looks like a refresh button).

Running the Tests

To run the tests: right click on the src/test directory and select "Run 'All Tests'"

Development Manual



Development Manual

Development Environment

The development environment requires a Java SDK version 1.8 or newer and JUnit 5 or newer and Maven.
The program can be compiled and ran using the IDE of choice although document only covers the Linux Command Line and IntelliJ.

Command Line Instructions

These instructions were written for a Linux command line and are untested on any other platform.

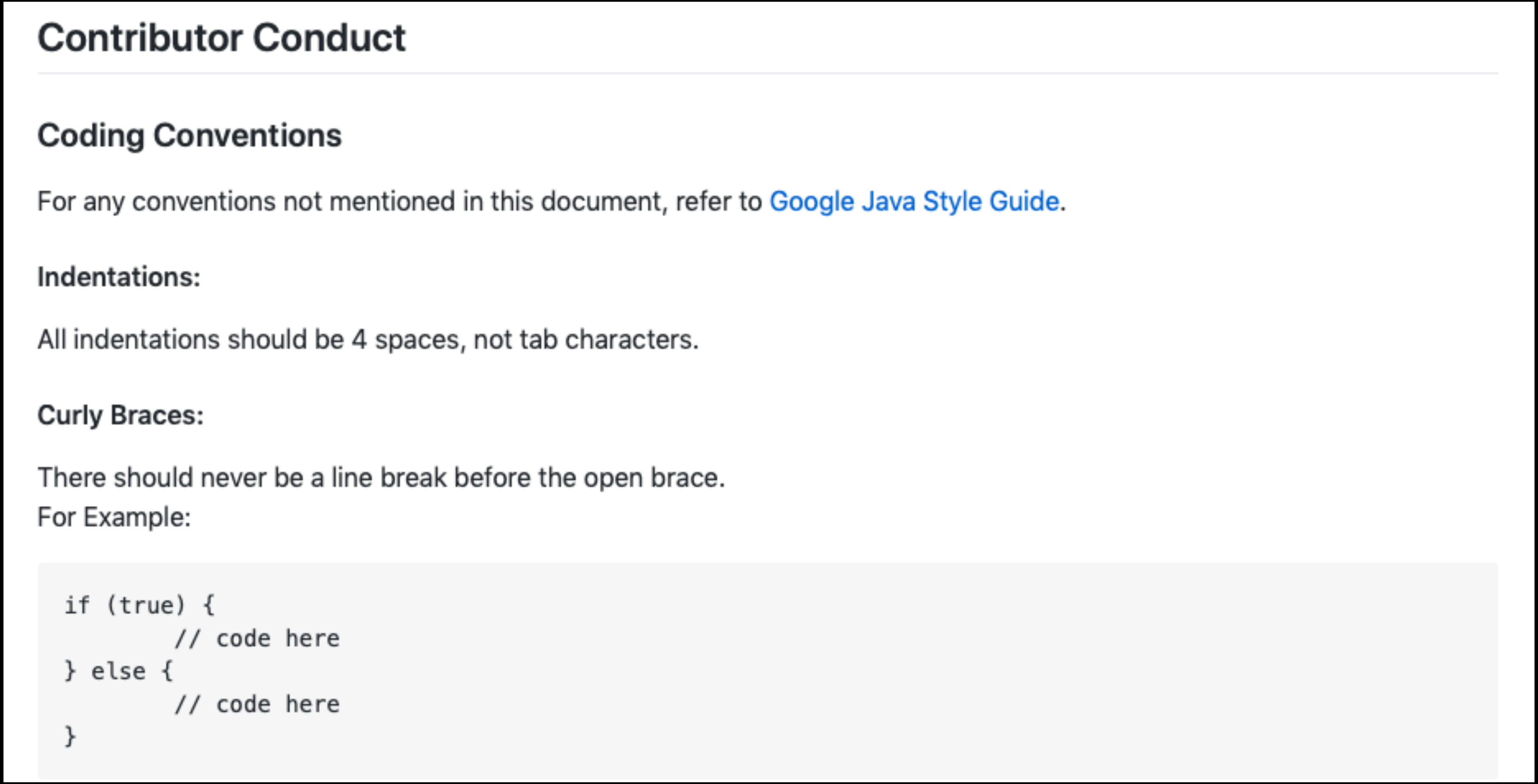
Running the Tests

There are not currently any known ways to reliably run the tests from the command line.

Running the Code

1. Compile the server code with `mvn package` (this may require `mvn install` to be run first).
2. Run the server with `java -cp ../target/cs414-f18-001-Method-Men-1.0-SNAPSHOT.jar edu.colostate.cs.cs414.method_men.jungle.server.TCPServer` .
3. Compile the client code with `javac src/main/edu/colostate/cs/cs414/method_men/jungle/client/*.java src/main/edu/colostate/cs/cs414/method_men/jungle/client/*/*.java` .
4. Run the client with `java -cp src/main/ edu.colostate.cs.cs414.method_men.jungle.client.socket.Client` (note that you may need to run multiple clients to fully test the game and this may require multiple terminals).

Development Manual



Contributor Conduct

Coding Conventions

For any conventions not mentioned in this document, refer to Google Java Style Guide.

Indentations:

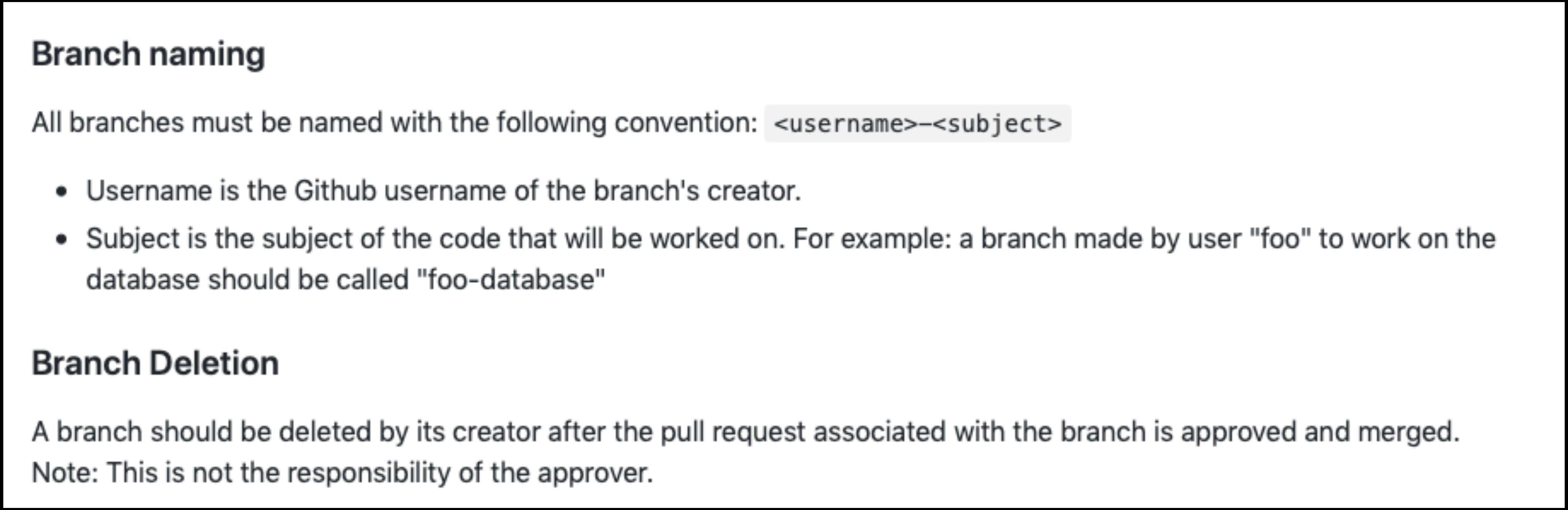
All indentations should be 4 spaces, not tab characters.

Curly Braces:

There should never be a line break before the open brace.
For Example:

```
if (true) {
        // code here
} else {
        // code here
}
```

Development Manual



Branch naming

All branches must be named with the following convention: `<username>-<subject>`

- Username is the Github username of the branch's creator.
- Subject is the subject of the code that will be worked on. For example: a branch made by user "foo" to work on the database should be called "foo-database"

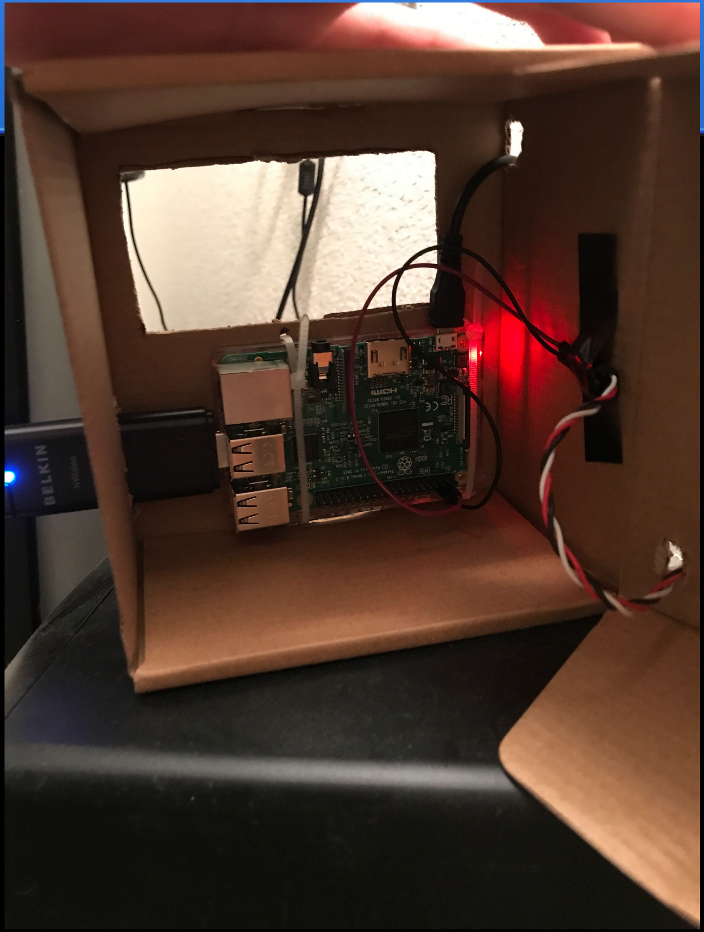
Branch Deletion

A branch should be deleted by its creator after the pull request associated with the branch is approved and merged. Note: This is not the responsibility of the approver.

Technologies used

- Git/GitHub, ZenHub
- Travis CI
- Slack main channel of communication
- IntelliJ IDE, Java
- Server/Client: TCP sockets
- GUI: Java Swing
- Database: MariaDB
- Database hosted on Marcel’s Raspberry Pi - mostly so we could connect to it when off campus

Database



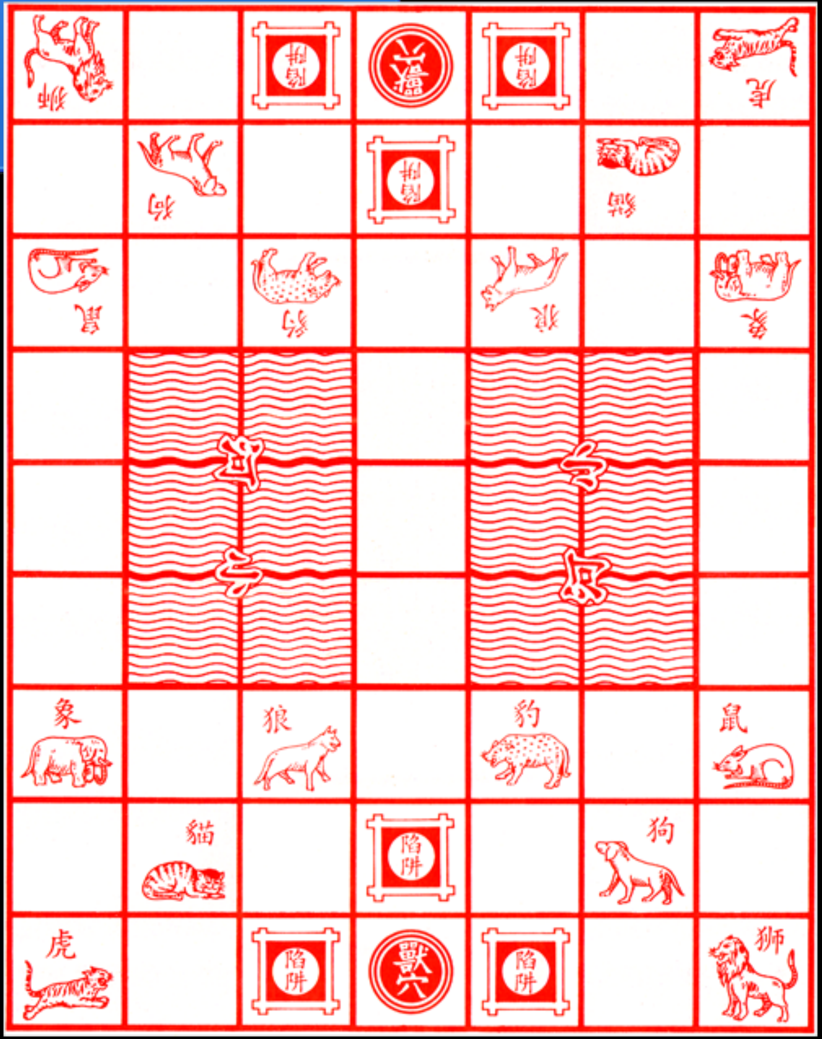
Difficulties Encountered

- Getting organized, not stepping on each other's toes
- Learning new technologies: SQL, Servers, Swing
- Had a bug for a long time where the server would eat up about 300% of CPU because of threading issues
- Learning JavaSwing, figuring out how to place things in particular spots in a JPanel, how to make buttons work
- Writing readable code, refactoring code written by someone else
- Going from a local game to an online game with two players

Lessons Learned

- Coding is easier and more efficient with better planning beforehand
- Good communication is vital
- Create issues that are small, bite-sized pieces
- Better to overestimate how long something will take to code
- Start earlier on the different pieces of the project

Demo


象
狼
豹
鼠
貓
陷阱
狗
虎
陷阱
獸穴
陷阱
獅