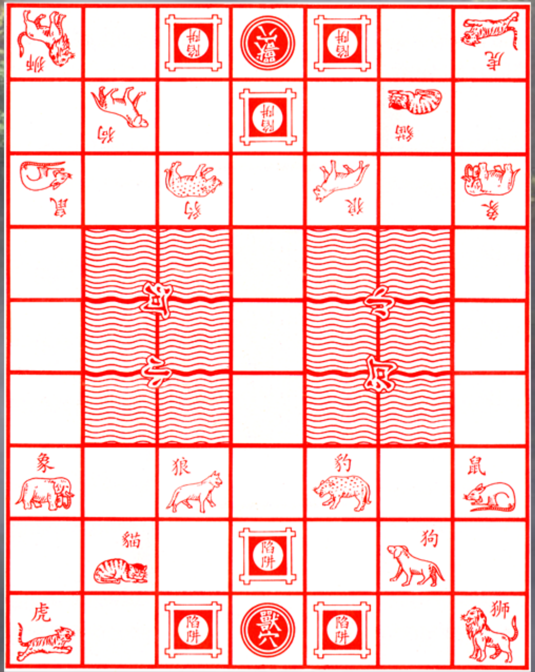
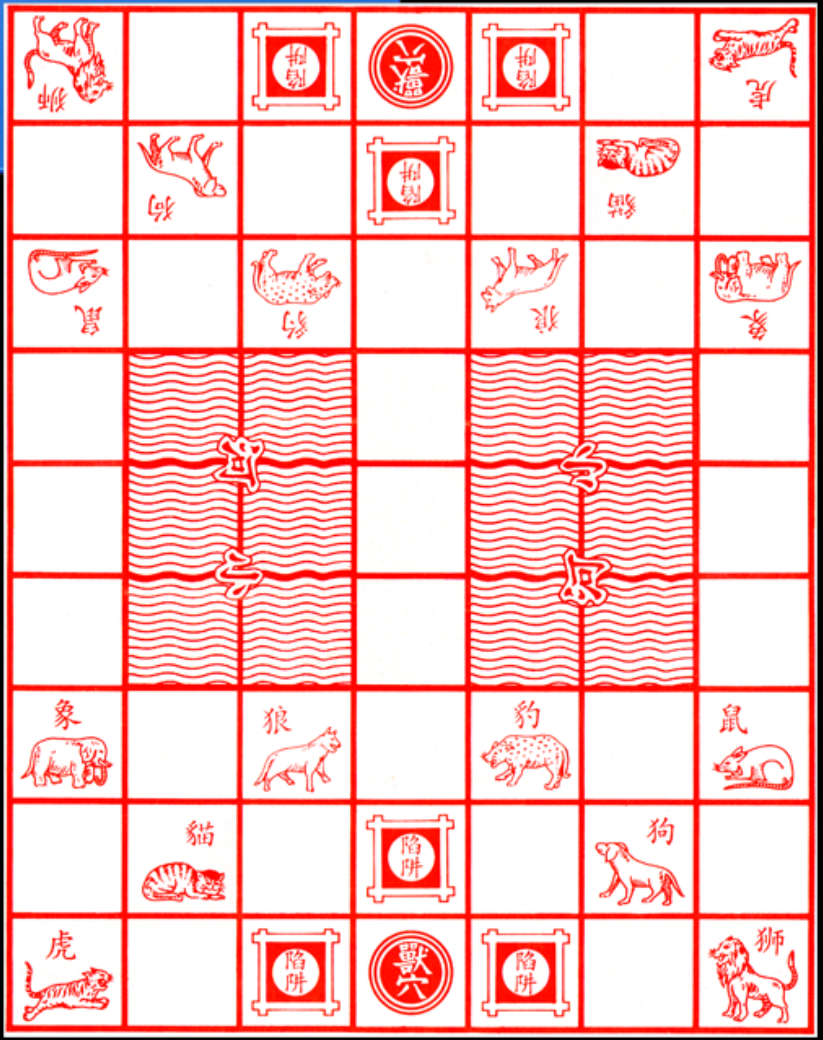
Jungle



Method Men

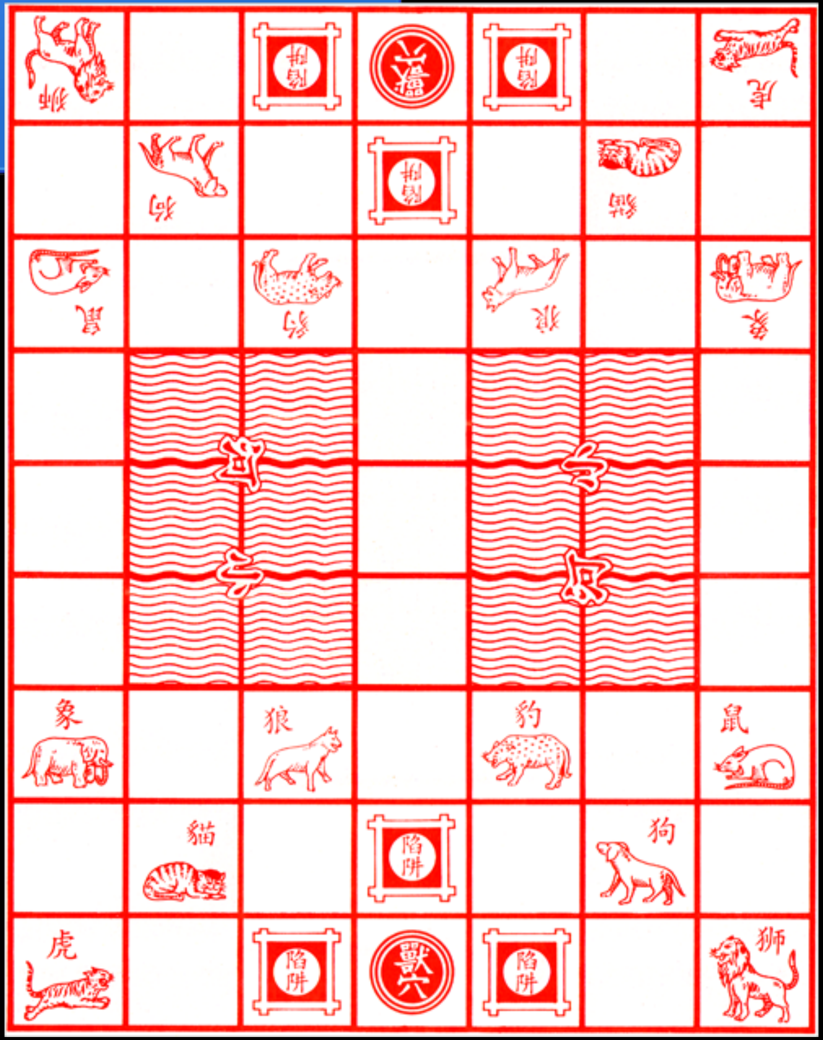
Overview

- Also known as: Dou Sou Qi, Children's Chess
- 7 x 9 board represents a jungle
- Board contains dens, traps set around dents and two rivers in the center
- Each player has 8 pieces representing different animals with different ranks



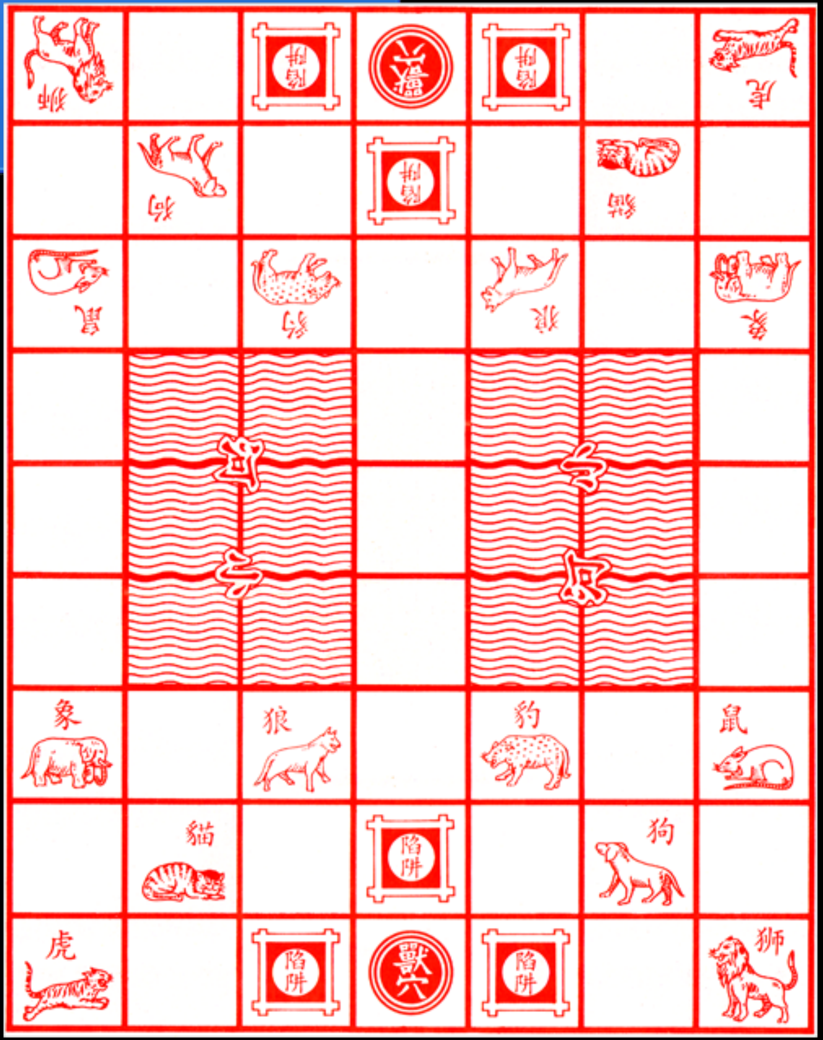
Objective

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

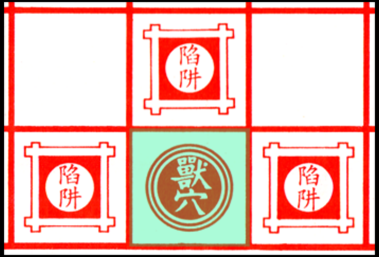


Pieces

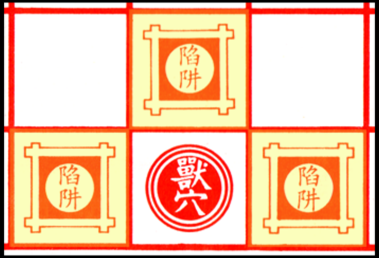
- 8 pieces, different animals and different ranks
- Higher rank = stronger piece
  - Elephant - 8
  - Lion - 7
  - Tiger - 6
  - Leopard - 5
  - Dog - 4
  - Wolf - 3
  - Cat - 2
  - Rat - 1



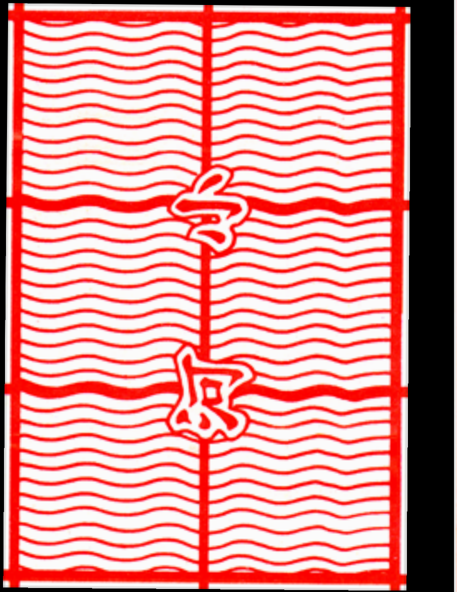
Board



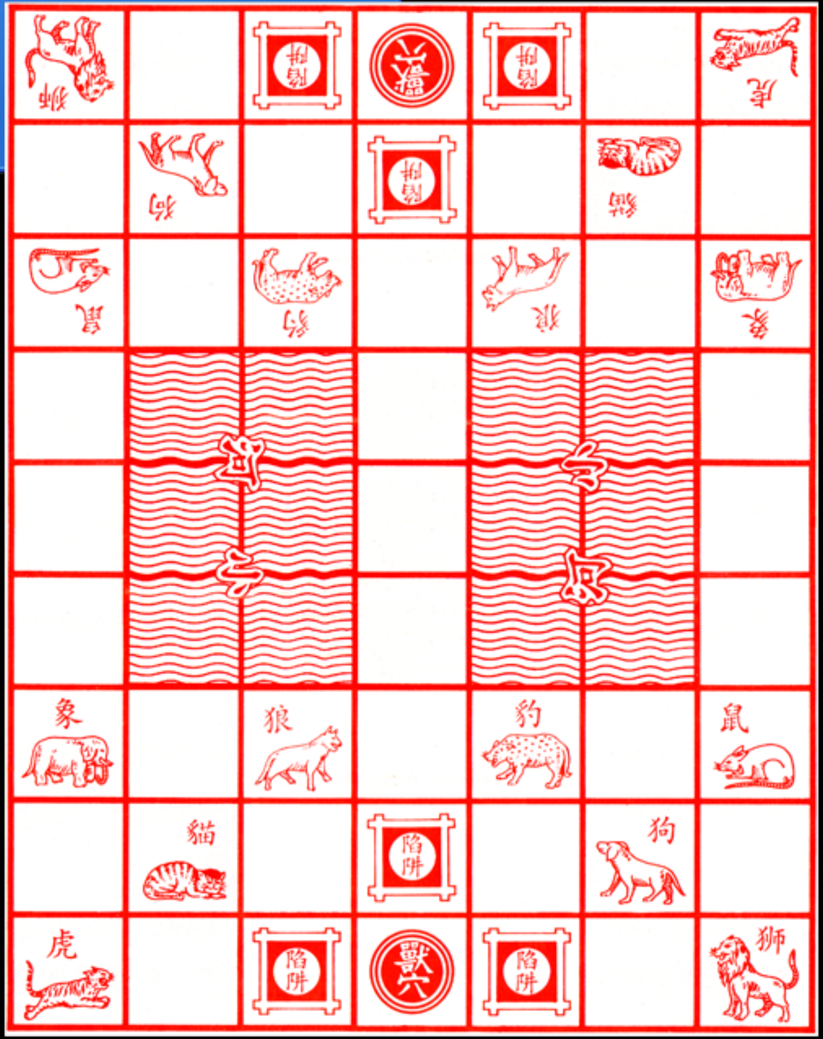
Den



Traps

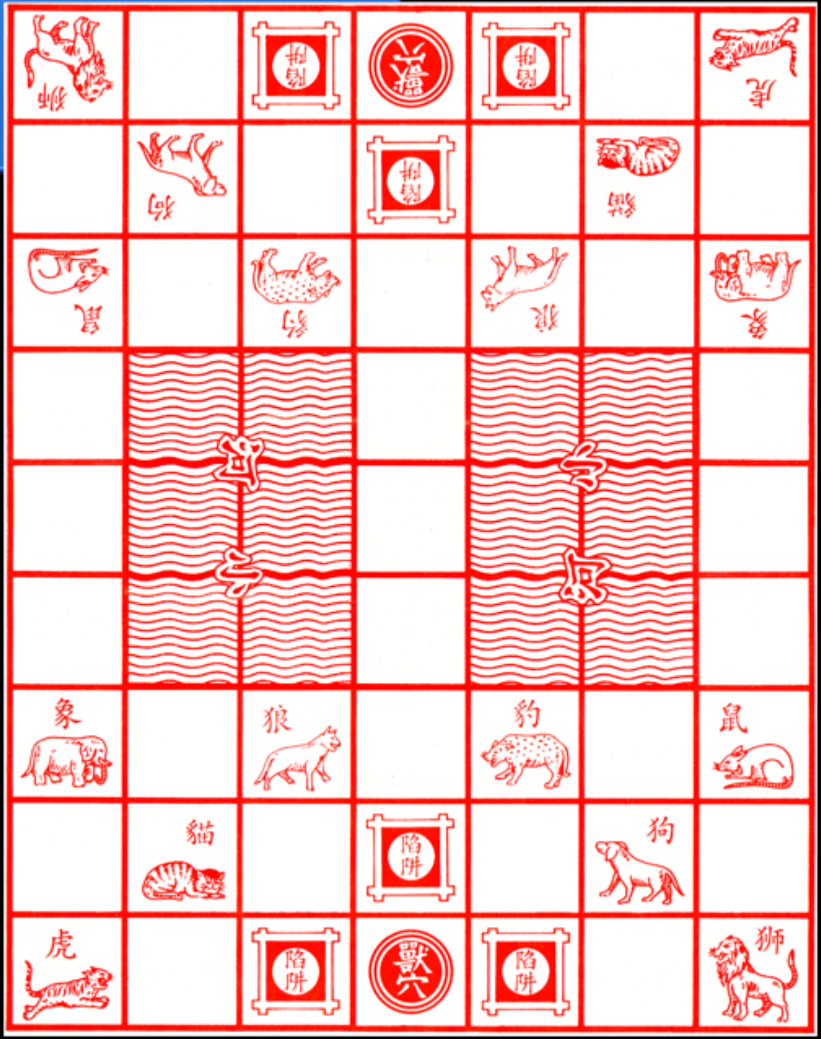


River



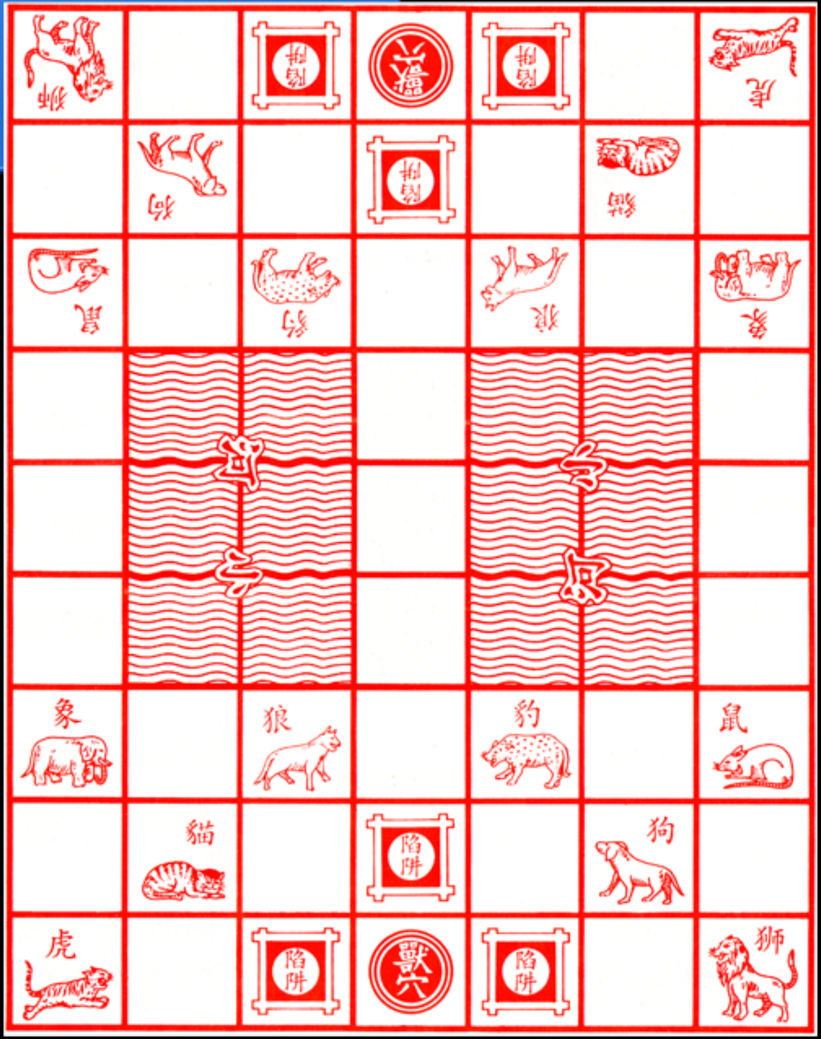
Movement

- 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



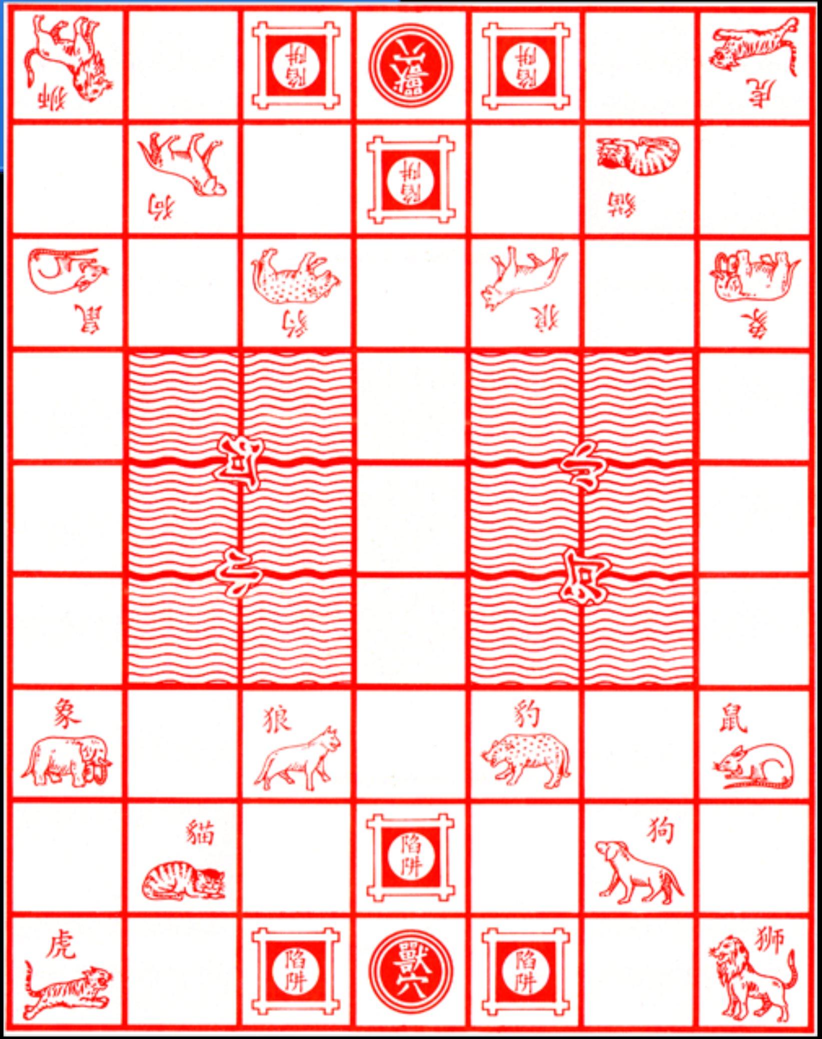
Water Tiles

- 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

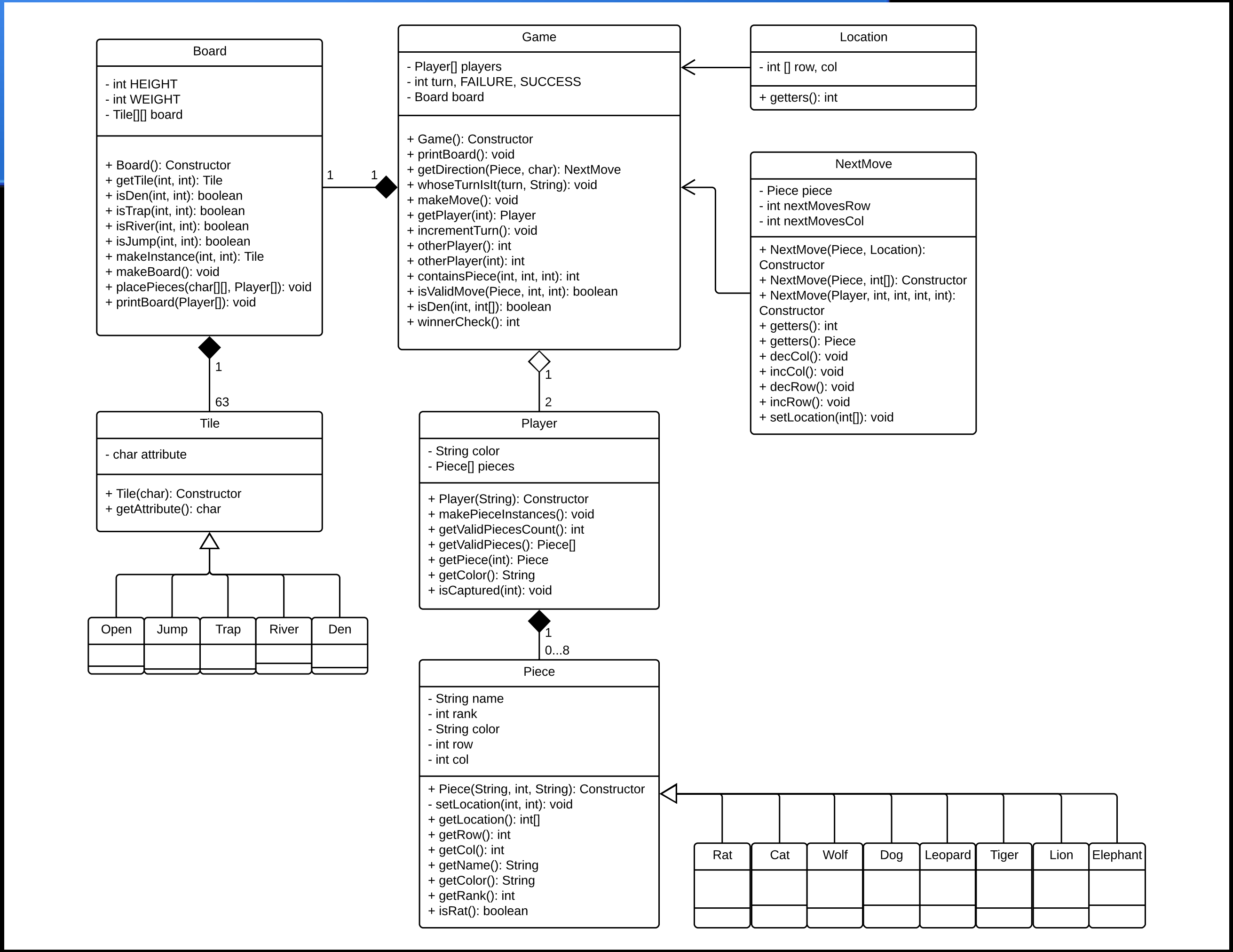


Specials

- 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



Board

- int HEIGHT
- int WEIGHT
- Tile[][] board

+ Board(): Constructor
+ getTile(int, int): Tile
+ isDen(int, int): boolean
+ isTrap(int, int): boolean
+ isRiver(int, int): boolean
+ isJump(int, int): boolean
+ makeInstance(int, int): Tile
+ makeBoard(): void
+ placePieces(char[][], Player[]): void
+ printBoard(Player[]): void

Game

- Player[] players
- int turn, FAILURE, SUCCESS
- Board board

+ Game(): Constructor
+ printBoard(): void
+ getDirection(Piece, char): NextMove
+ whoseTurnIsIt(turn, String): void
+ makeMove(): void
+ getPlayer(int): Player
+ incrementTurn(): void
+ otherPlayer(): int
+ otherPlayer(int): int
+ containsPiece(int, int, int): int
+ isValidMove(Piece, int, int): boolean
+ isDen(int, int[]): boolean
+ winnerCheck(): int

Location

- int [] row, col

+ getters(): int

NextMove

- Piece piece
- int nextMovesRow
- int nextMovesCol

+ NextMove(Piece, Location): Constructor
+ NextMove(Piece, int[]): Constructor
+ NextMove(Player, int, int, int, int): Constructor
+ getters(): int
+ getters(): Piece
+ decCol(): void
+ incCol(): void
+ decRow(): void
+ incRow(): void
+ setLocation(int[]): void

Tile

- char attribute

+ Tile(char): Constructor
+ getAttribute(): char

Open | Jump | Trap | River | Den

Player

- String color
- Piece[] pieces

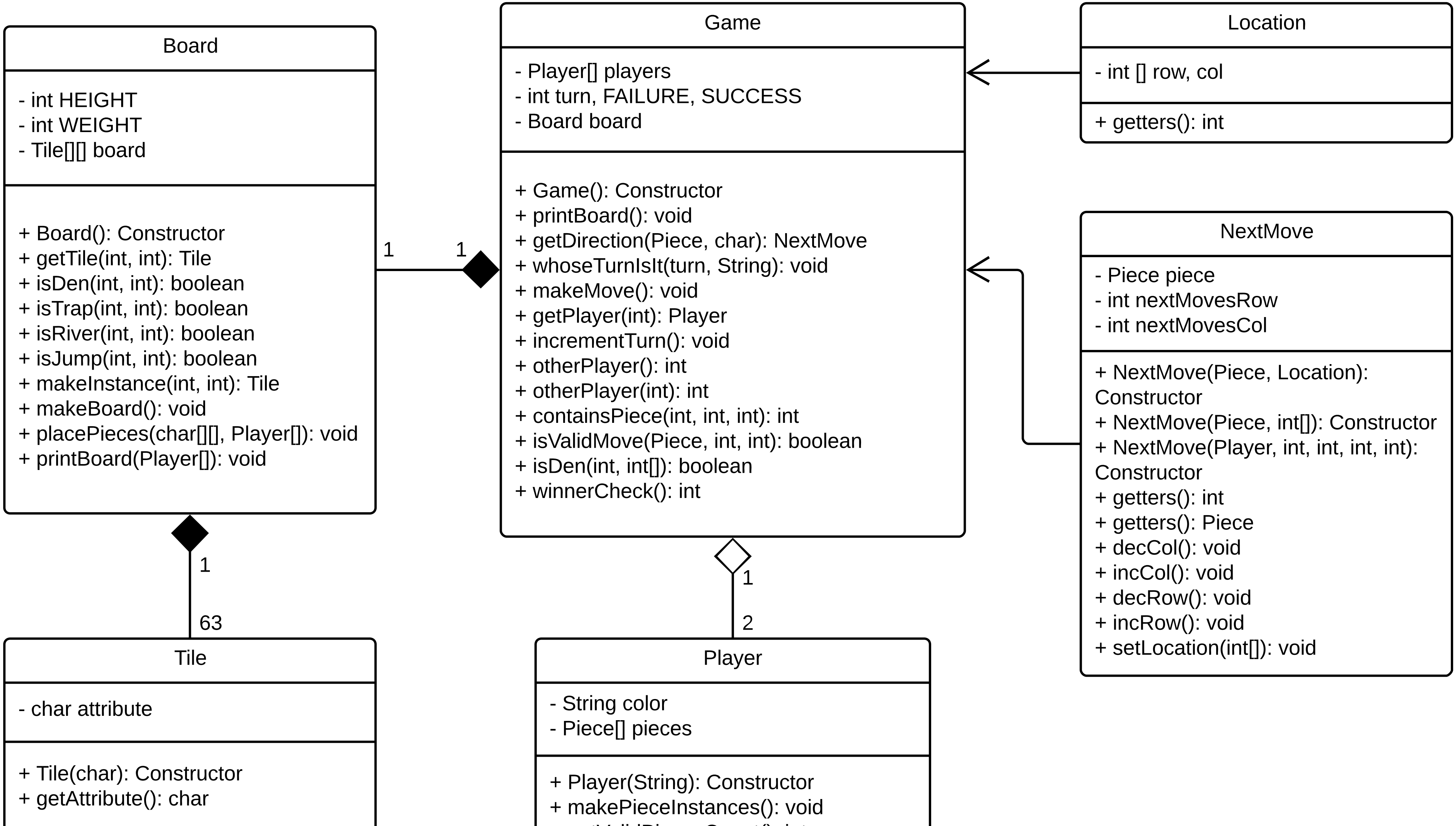
+ Player(String): Constructor
+ makePieceInstances(): void
+ getValidPiecesCount(): int
+ getValidPieces(): Piece[]
+ getPiece(int): Piece
+ getColor(): String
+ isCaptured(int): void

Piece

- String name
- int rank
- String color
- int row
- int col

+ Piece(String, int, String): Constructor
- setLocation(int, int): void
+ getLocation(): int[]
+ getRow(): int
+ getCol(): int
+ getName(): String
+ getColor(): String
+ getRank(): int
+ isRat(): boolean

Rat | Cat | Wolf | Dog | Leopard | Tiger | Lion | Elephant



Board

- int HEIGHT
- int WEIGHT
- Tile[][] board

+ Board(): Constructor
+ getTile(int, int): Tile
+ isDen(int, int): boolean
+ isTrap(int, int): boolean
+ isRiver(int, int): boolean
+ isJump(int, int): boolean
+ makeInstance(int, int): Tile
+ makeBoard(): void
+ placePieces(char[][], Player[]): void
+ printBoard(Player[]): void

Game

- Player[] players
- int turn, FAILURE, SUCCESS
- Board board

+ Game(): Constructor
+ printBoard(): void
+ getDirection(Piece, char): NextMove
+ whoseTurnIsIt(turn, String): void
+ makeMove(): void
+ getPlayer(int): Player
+ incrementTurn(): void
+ otherPlayer(): int
+ otherPlayer(int): int
+ containsPiece(int, int, int): int
+ isValidMove(Piece, int, int): boolean
+ isDen(int, int[]): boolean
+ winnerCheck(): int

Location

- int [] row, col

+ getters(): int

NextMove

- Piece piece
- int nextMovesRow
- int nextMovesCol

+ NextMove(Piece, Location): Constructor
+ NextMove(Piece, int[]): Constructor
+ NextMove(Player, int, int, int, int): Constructor
+ getters(): int
+ getters(): Piece
+ decCol(): void
+ incCol(): void
+ decRow(): void
+ incRow(): void
+ setLocation(int[]): void

Tile

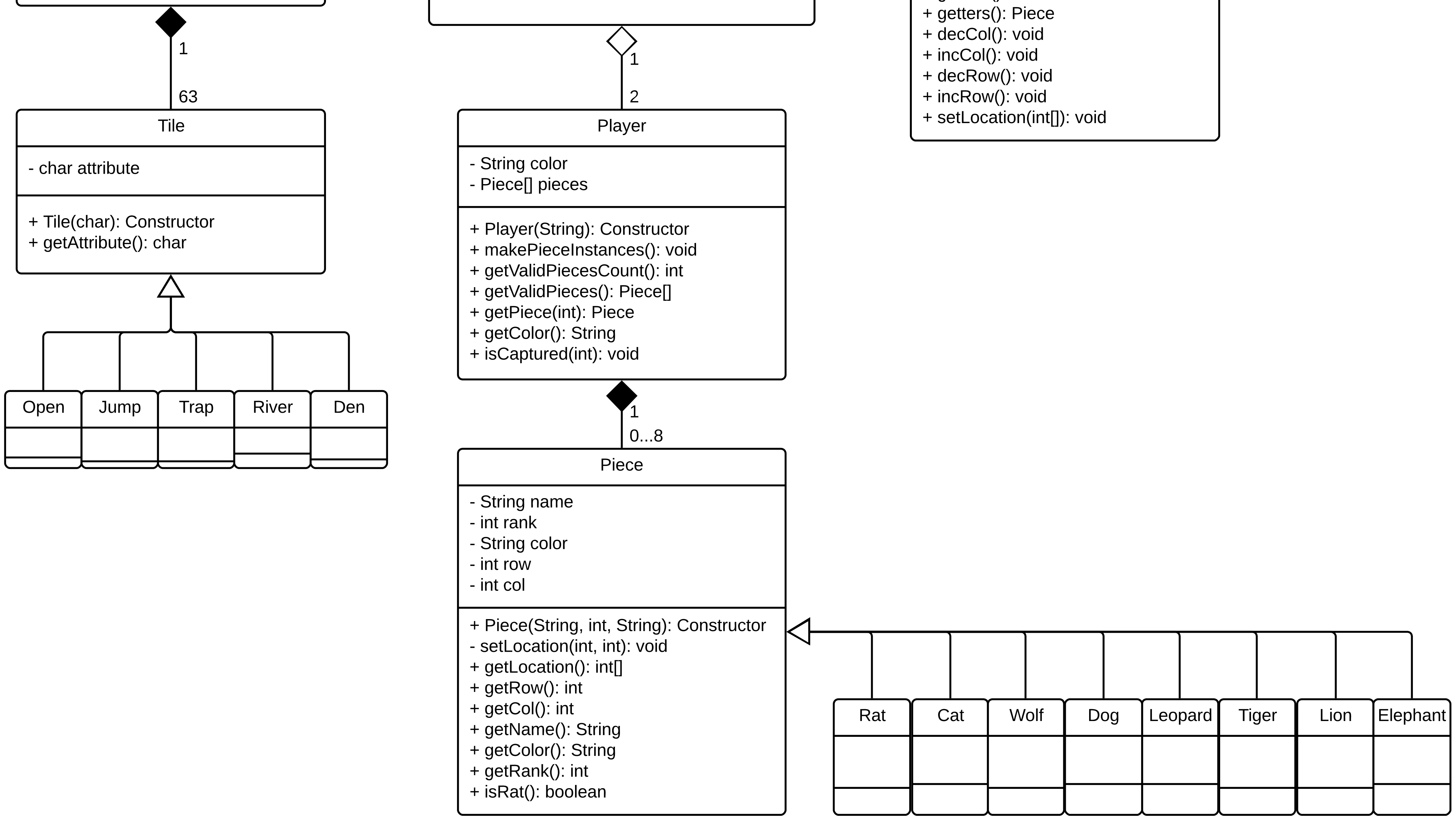
- char attribute

+ Tile(char): Constructor
+ getAttribute(): char

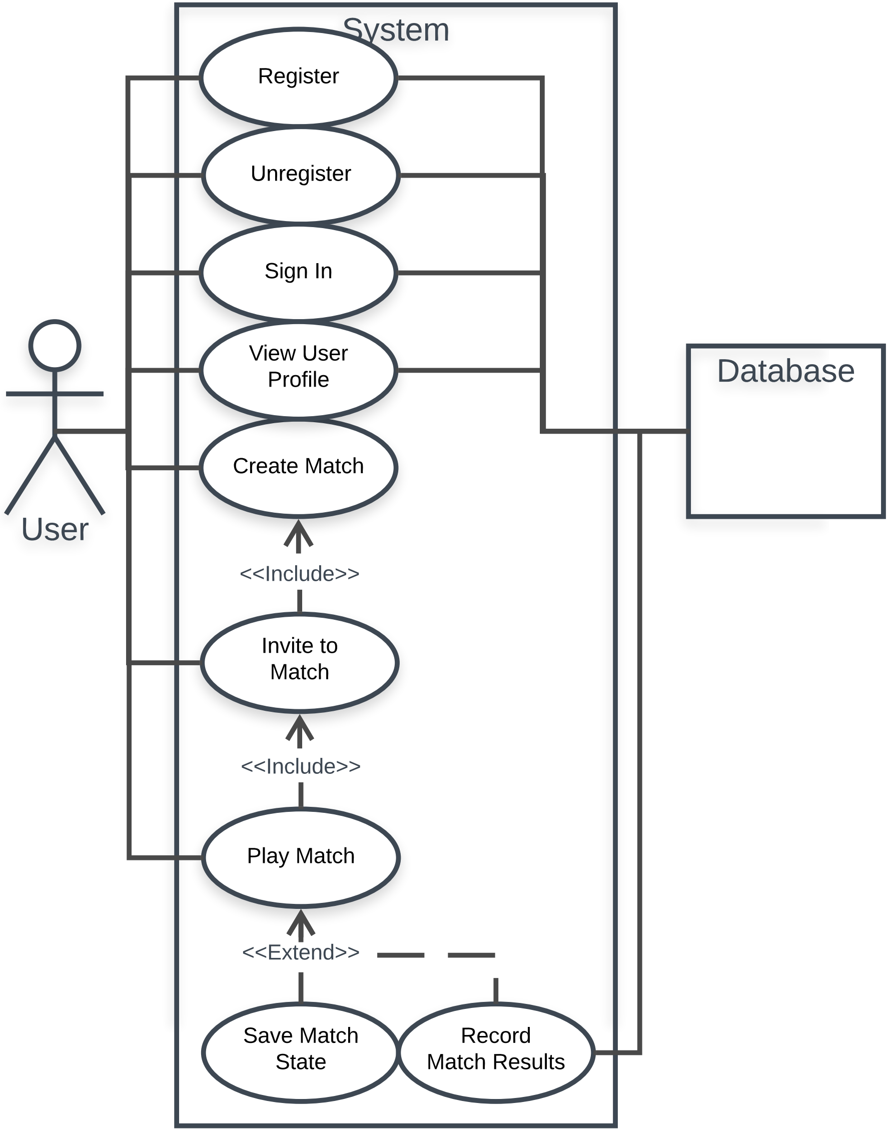
Player

- String color
- Piece[] pieces

+ Player(String): Constructor
+ makePieceInstances(): void
+ getValidPieceCount(): int


+ getters(): Piece
+ decCol(): void
+ incCol(): void
+ decRow(): void
+ incRow(): void
+ setLocation(int[]): void
1
63
Tile
- char attribute
+ Tile(char): Constructor
+ getAttribute(): char
Open
Jump
Trap
River
Den
1
2
Player
- String color
- Piece[] pieces
+ Player(String): Constructor
+ makePieceInstances(): void
+ getValidPiecesCount(): int
+ getValidPieces(): Piece[]
+ getPiece(int): Piece
+ getColor(): String
+ isCaptured(int): void
1
0...8
Piece
- String name
- int rank
- String color
- int row
- int col
+ Piece(String, int, String): Constructor
- setLocation(int, int): void
+ getLocation(): int[]
+ getRow(): int
+ getCol(): int
+ getName(): String
+ getColor(): String
+ getRank(): int
+ isRat(): boolean
Rat
Cat
Wolf
Dog
Leopard
Tiger
Lion
Elephant

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.

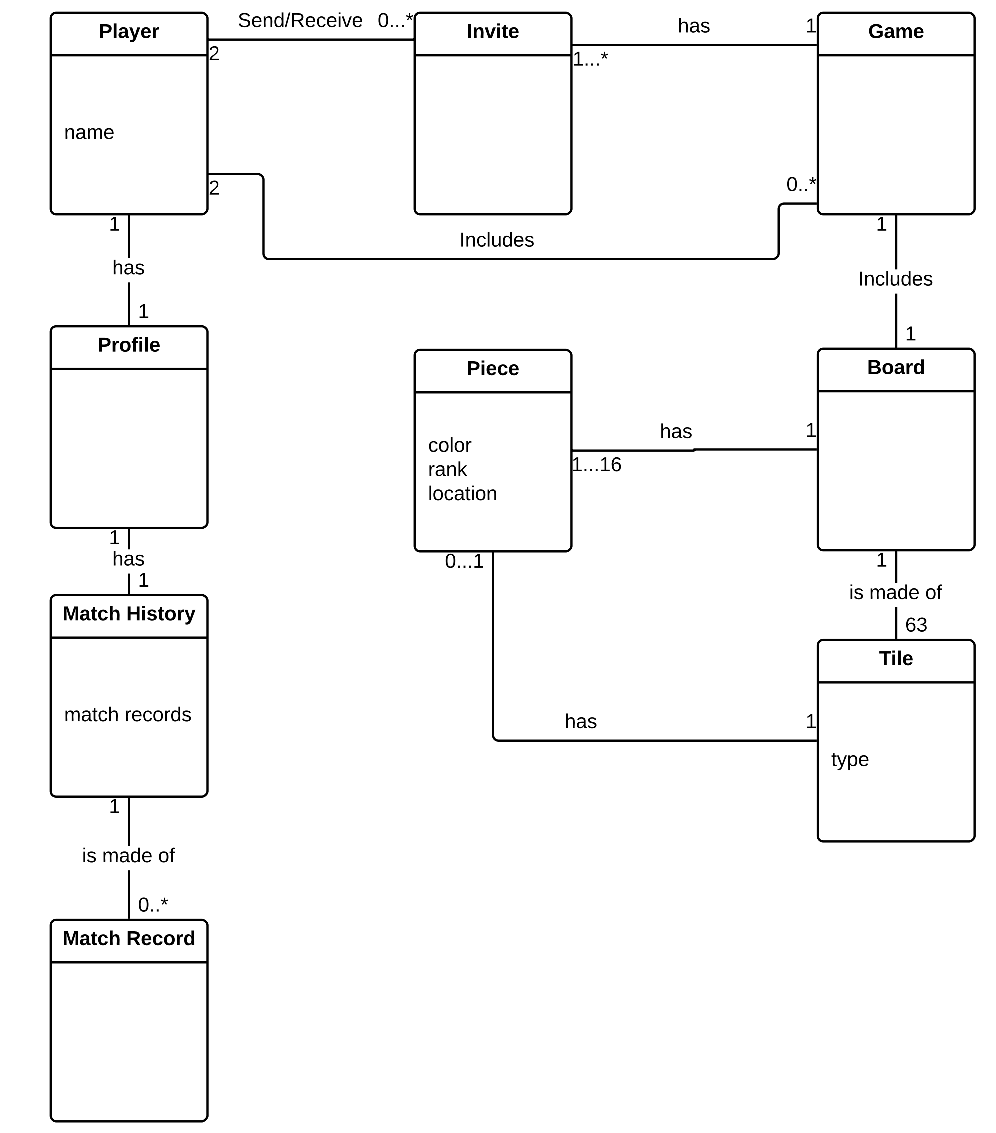
Player: Registered user of the system.

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.



Player
name

Send/Receive
2
0...*

Invite

has
1...*
1

Game

Includes
2
0..*

has
1
1

Profile

has
1
1

Match History
match records

is made of
1
0..*

Match Record

Includes
1
1

Board

Piece
color
rank
location

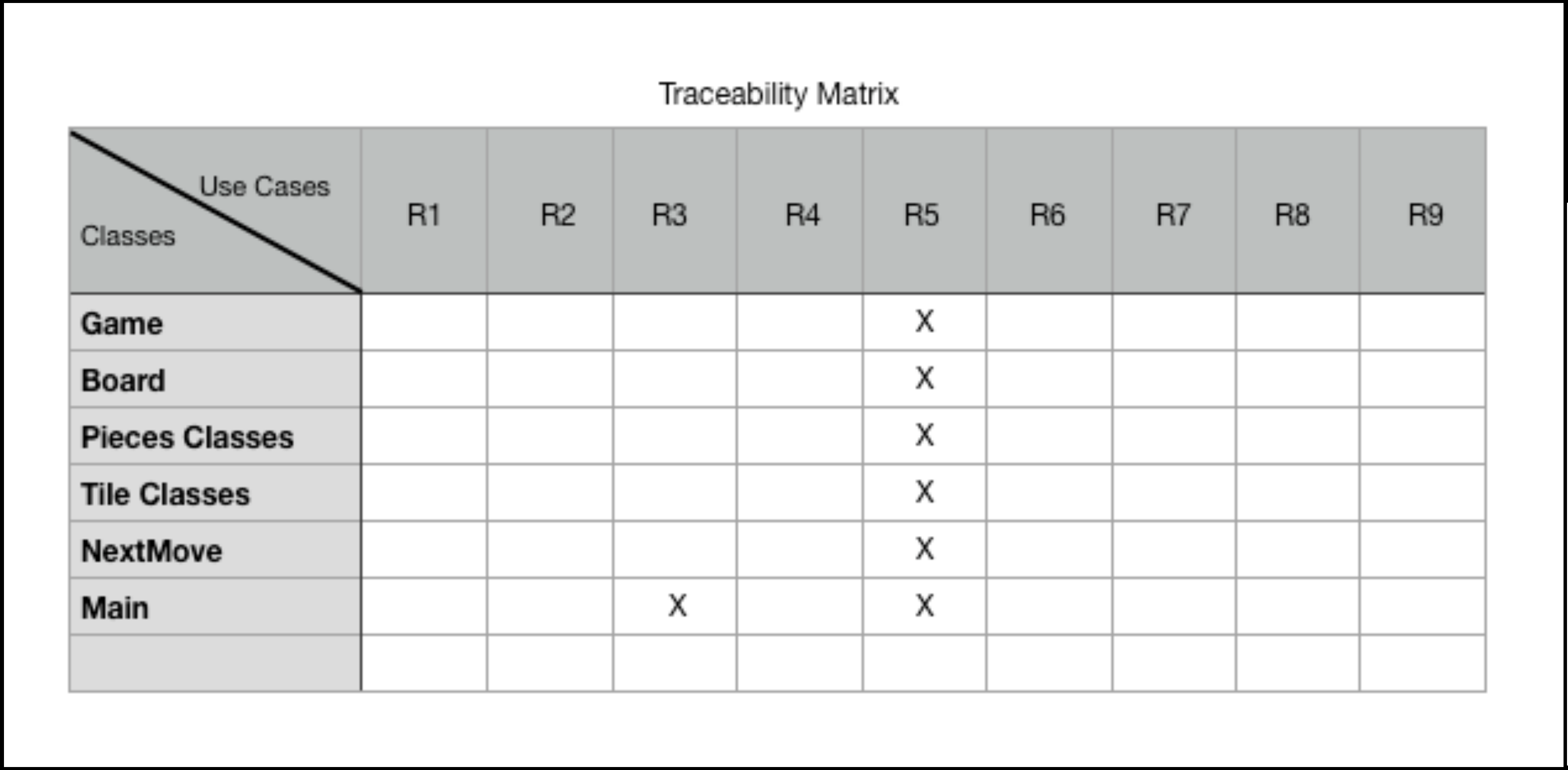
has
1...16
1

is made of
1
63

Tile
type

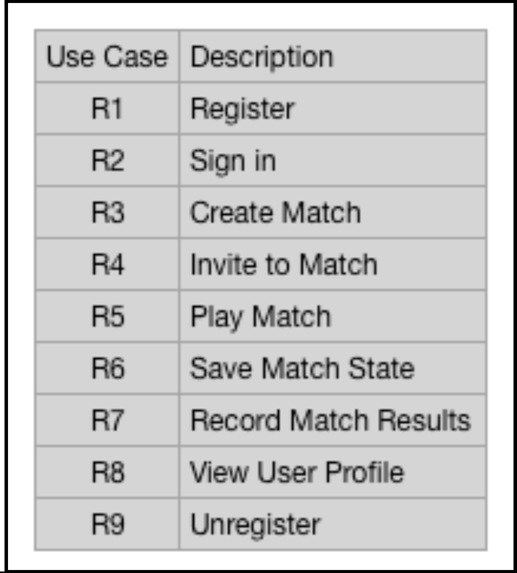
has
0...1
1

Traceability Matrix



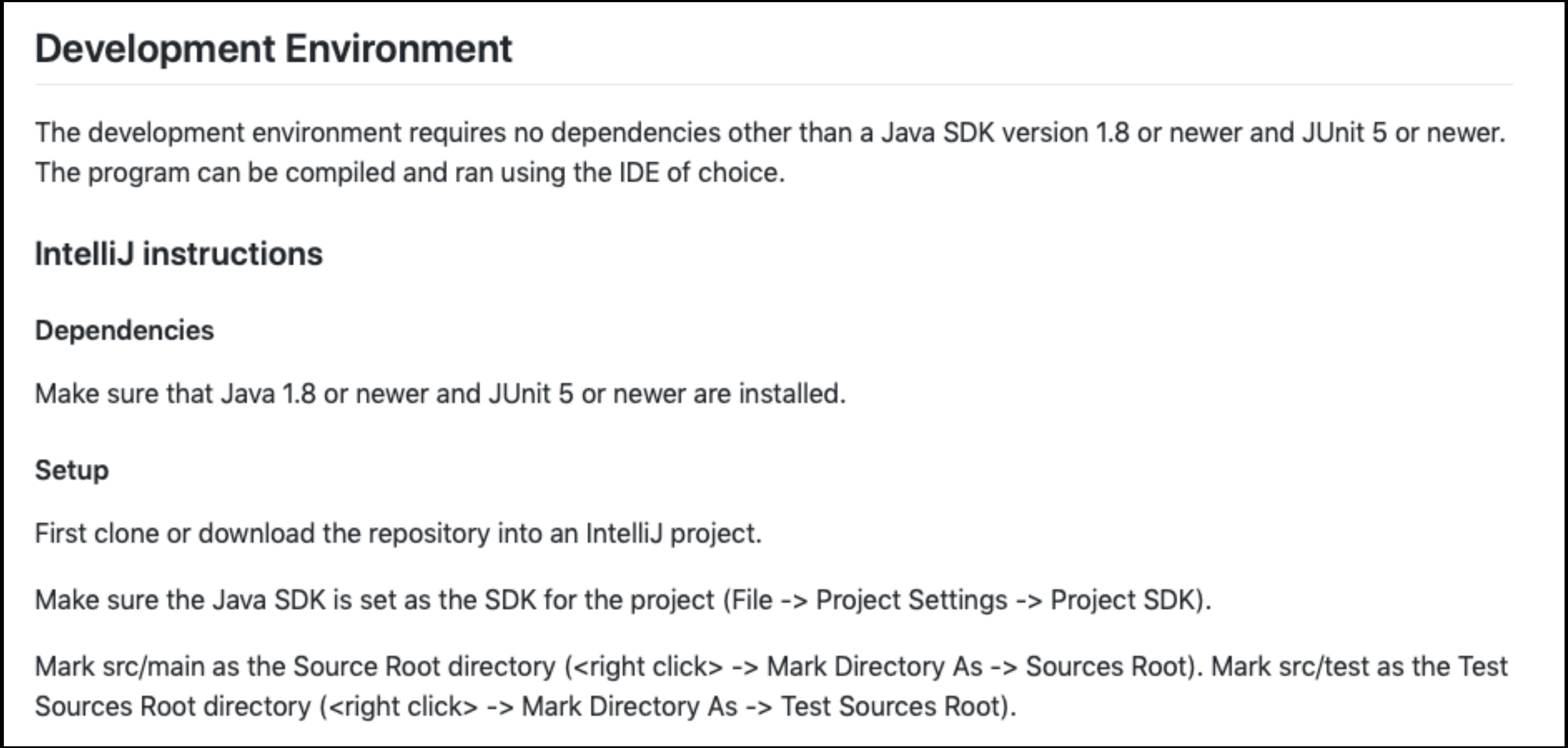
Traceability Matrix

Use Cases Classes	R1	R2	R3	R4	R5	R6	R7	R8	R9
Game					X				
Board					X				
Pieces Classes					X				
Tile Classes					X				
NextMove					X				
Main			X		X				



Use Case	Description
R1	Register
R2	Sign in
R3	Create Match
R4	Invite to Match
R5	Play Match
R6	Save Match State
R7	Record Match Results
R8	View User Profile
R9	Unregister

Development Manual



Development Environment

The development environment requires no dependencies other than a Java SDK version 1.8 or newer and JUnit 5 or newer. The program can be compiled and ran using the IDE of choice.

IntelliJ instructions

Dependencies

Make sure that Java 1.8 or newer and JUnit 5 or newer are installed.

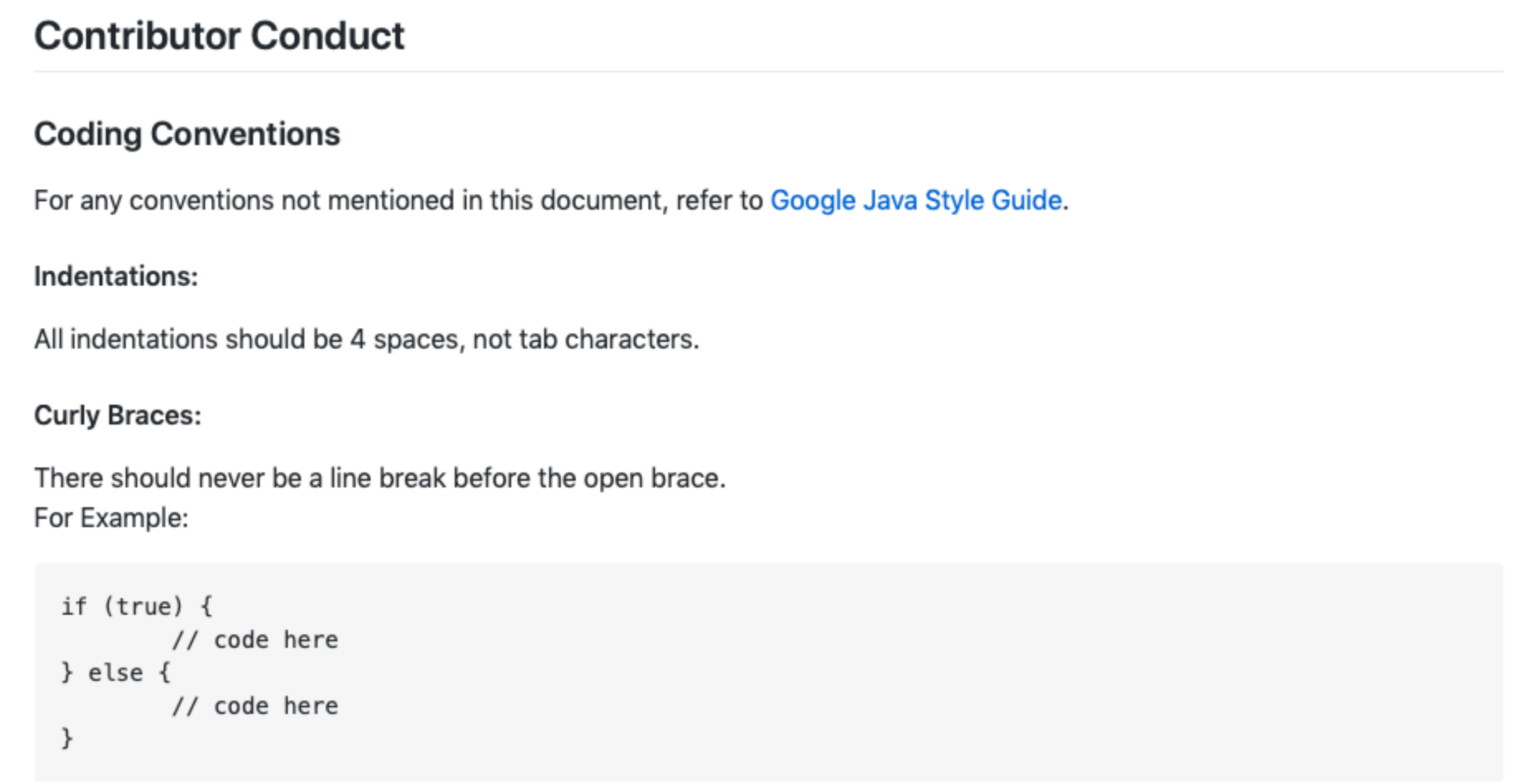
Setup

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).

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

Development Manual



Contributor Conduct

Coding Conventions

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

Indentations:

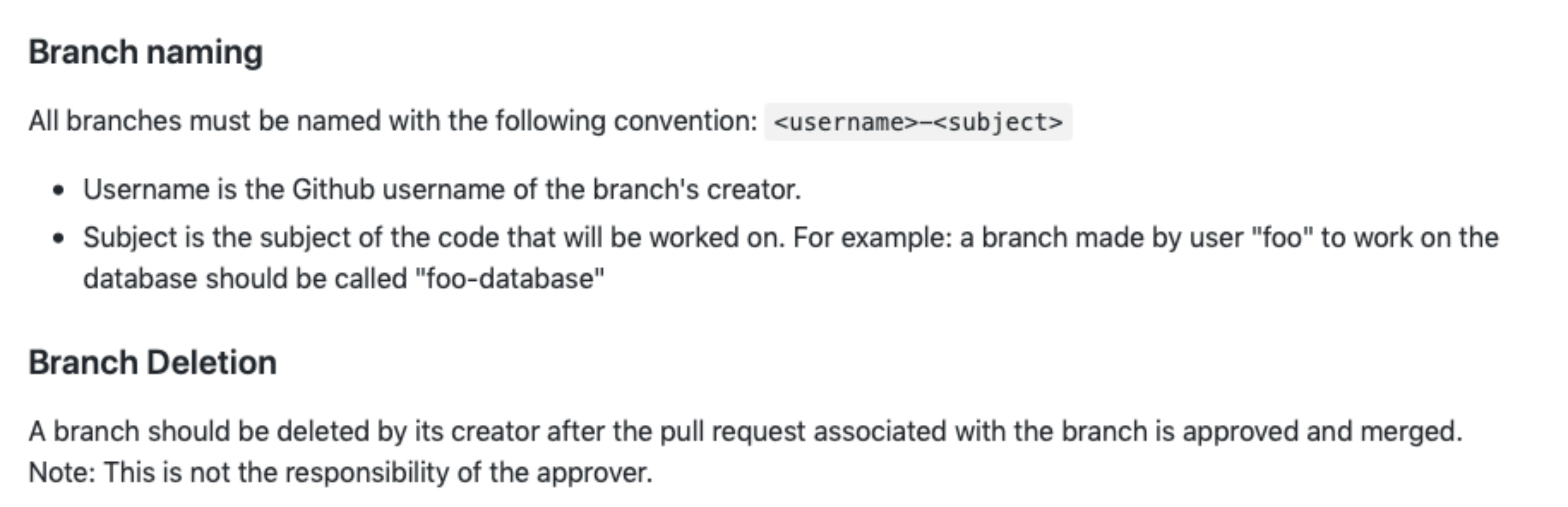
All indentations should be 4 spaces, not tab characters.

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

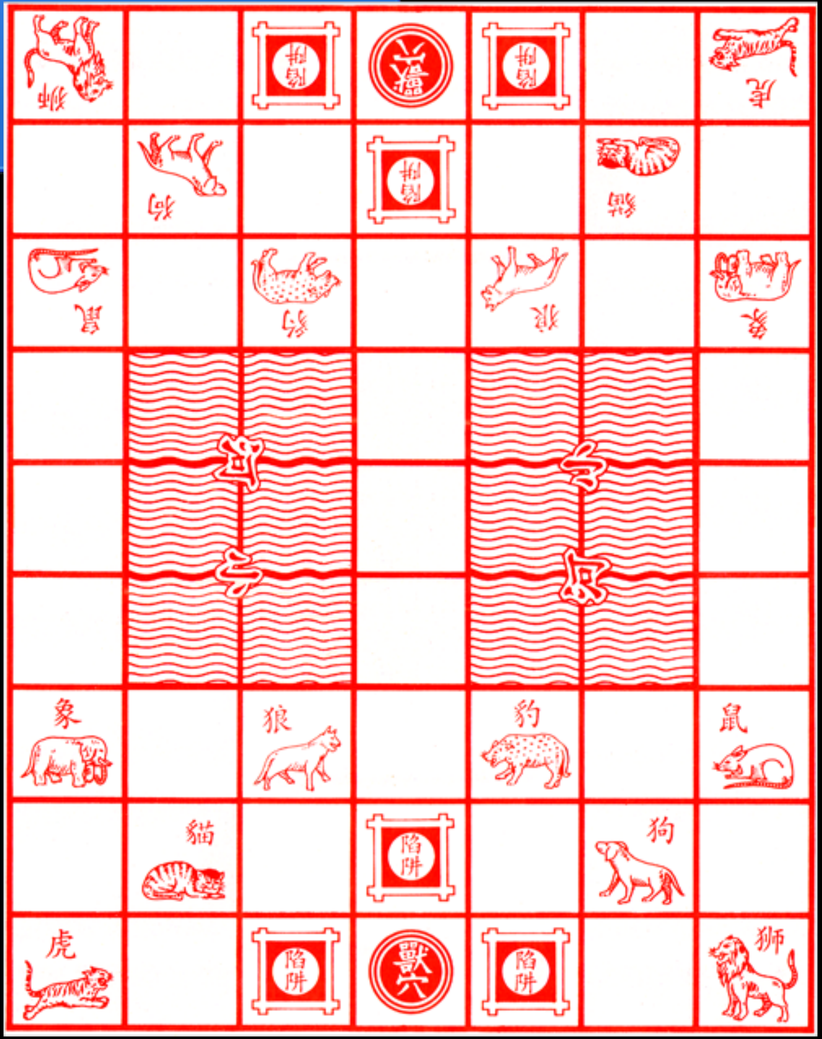
All branches must be named with the following convention: `<username>-<subject>`

- 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.

Demo


獅
陷阱
獸穴
陷阱
虎
狗
陷阱
貓
鼠
豹
狼
象
象
狼
豹
鼠
貓
陷阱
狗
虎
陷阱
獸穴
陷阱
獅