Xiàngqí: Chinese Chess

Developer's Guide

Gary Pollice

ABSTRACT

Xiàngqí is a game of strategy that is similar to the game that we know as chess. It has a different type of board and pieces that are similar to standard chess pieces, but with quite different rules.

This manual provides helpful hints for developers who will produce software implementations of the game for CS4233: Object-Oriented Analysis and Design.

Last Modified: 13-Feb-2017

Date	Version	Changes
29-Nov-2016	0.1	Initial version draft started
28-Dec-2016	0.2	Through Beta Xiàngqí
6-Feb-2017	0.3	Add Gamma Xiàngqí
13-Feb-2017	0.4	Clarify check and legal moves.
19-Feb-2017	0.5	Additional clarifications based upon comments in the course discussion forum

Table of Contents

xiàngqí: Chinese Chess

1. Introduction

Xiàngqí, also written as Xiangqi and Xiàng Qí, is also known as Chinese Chess. It contains many components and concepts that are identical or similar to western Chess as played in western countries. It has a board, several pieces of different types, and the same goal—to capture the General (King) through checkmate. Some of the differences are the presence of a river that flows horizontally across the middle of the board, the Cannon piece that can only capture another piece if it jumps over a piece, and restrictions on which parts of the board pieces may occupy.

Xiàngqí uses a board similar to a chessboard, but instead of occupying the squares of the board, the pieces occupy the intersection at the corners of the squares, including the intersections at the edges of the board. This larger board has 8 squares horizontally and 9 squares vertically. This yields an effective 9 X 10 playing field.

Games are played as a sequence of *turns*. Each turn consists of two *moves*, the first being the Red player's move and the second being the Black player's move. Red always makes the first move.

Like chess, Xiàngqí has several variations [1] that add additional challenges to the game; although most players play the standard game. The game has a following around the world, with tournaments, rankings, books, and other publications supporting it.

This term you will implement a service that manages a game of Xiàngqí. This means that a Xiàngqí game program can use it to ensure that moves made by players—real or virtual—are correct. The service determines when a player wins and reports the results of the game and provides an optional record of the moves. In order to support a client designed to teach and analyze Xiàngqí games, the service may provide an undo function.

2. Game Mechanics

This section describes the board and the notation we use to reference locations on the board. It then describes the game pieces and how each piece moves in the standard game. Before continuing with this section, read the Wikipedia Xiàngqí rules [1]. The material that follows assumes that you already have read those rules.

2.1. The board

The board consists of an 8 X 9 grid of squares; that means eight columns and nine rows. Since the pieces occupy the intersection points, the board has an effective space of 9 X 10 locations [Figure 1]. There are nine *files* and ten *ranks*.

The blue squares that go across the middle of the board represent the river that runs through the kingdom. Some pieces may not cross the river. The two shaded square areas represent the Red and Black palaces. Certain pieces are restricted to occupy one of the nine locations within the player's palace.

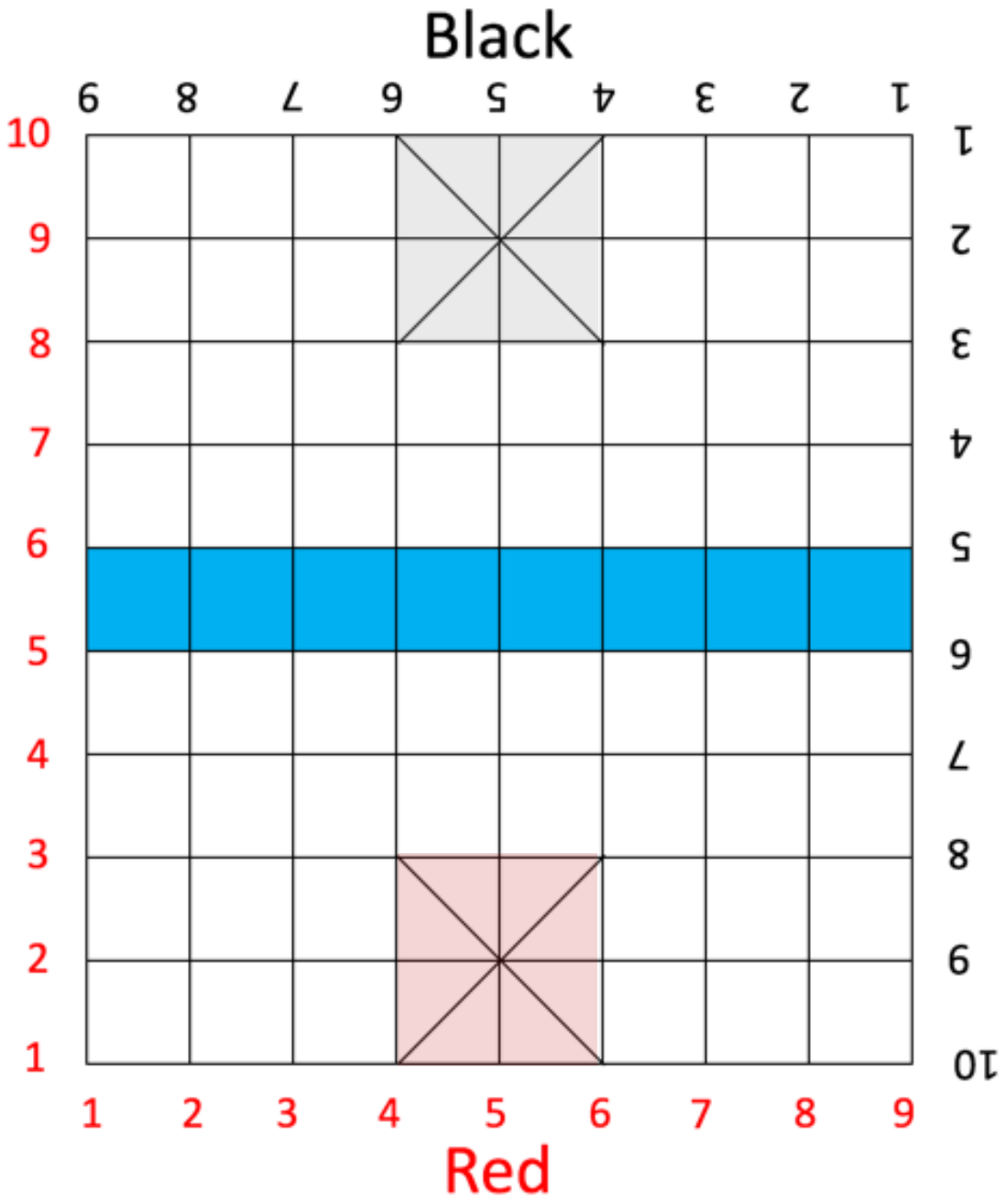

Figure 1. The Xiàngqí board

2.1.1. Board notation

Several notation methods exist for identifying locations on the Xiàngqí board and moves during a game. We will use a method similar to the first method described in [1]. A location is represented as rf, where r is the rank and f is the file. For example, the central point of a player's palace is 25 (2 is the rank and 5 is the file). The furthest point in the opponent's palace is 105 (10 is the rank and 5 is the file). Section 2.6 describes the notation for making moves.

The notation method we use means that every location on the board has two notations, depending upon the viewpoint of the player the notation refers to. For example, the middle of the Red player's palace is 25 from Red's viewpoint and 95 from that of Black.

2.2. Pieces

The standard pieces of Xiàngqí have a close resemblence to western chess pieces, with a couple of exceptions. However, the movement is quite different for most of them. The following subsections describe each piece, the Chinese names and symbols that are typically used, the name and symbol we use for it in our implementation (in the heading for each piece), the starting position(s), and the movement of the piece in the standard Xiàngqí game. For some pieces in Xiàngqí, different names symbols are used for the same piece, depending upon which side (Red or Black) the piece belongs to. We use one name and symbol (letter) to identify pieces in our game as a simplification. The information for the pieces is taken from [1] [2] [3].

2.2.1. General (G)

As in western chess, the King is the key player. When the King is placed in a position of *checkmate* (under attack and cannot get away), the game is over and the mating player wins. The King's characteristics are:

Chinese names	Jiàng (Governer), Shuài (General)
Chinese Symbols	
Starting position	15
Movement	The general may move and capture one point orthogonally, with the following exception. The two generals may not face each other along the same file with no intervening pieces. If that happens, the 飛將 ("flying general") move may be executed, in which the general to move may cross the board to capture the enemy general. In practice, this rule is only used to enforce checkmate. The general may not leave the palace except when executing the flying general move.

2.2.2. Advisor (A)

The Advisor is always close to the King.

Chinese names	Shì (Scholar, Guard, Gentleman, Officer, Official)
Chinese Symbols	
Starting position	14, 16
Movement	Move and capture one point diagonally and may not leave the palace, which confines them to five points on the board.

2.2.3. Elephant (E)

The Elephant is a defensive piece with limited movement capabilities.

Chinese names	Xiàng (Elephant, Minister)
Chinese Symbols	
Starting position	13, 17
Movement	Move and capture onexactly two points diagonally and may not jump over intervening pieces. If an Elephant cannot move due to a diagonally adjacent piece, it is known as "blocking the elephant's eye." The Elephant may not cross the river, it is mainly a defensive piece.

2.2.4. Horse (H)

The Horse is similar to the Knight in western chess. However, it may not jump over pieces, as the Knight does. This means that it may be blocked from a move that the Knight might make in western chess.

Chinese names	*Mǎ* for Black and *Mà* for Red
Chinese Symbols	
Starting position	12, 18
Movement	A horse moves and captures by moving one point orthogonally and then one point diagonally away from its former position, in this specific order. This is the same as the Knight in western chess, with the exception that *the Horse may not jump over any piece*. Each point that the Horse stops on (the orthogonal point) must be unoccupied.

2.2.5. Chariot (R)

The Chariot is the most powerful piece on the board. It is equivalent to the Rook in western chess.

Chinese names	*Jū* or *chē*
Chinese Symbols	
Starting position	11, 19
Movement	The Chariot moves exactly like the Rook in western chess: as many spaces as it wishes horizontally or vertically, until it meets another piece or the edge of the board. When it meets another piece, it may capture the piece and take that piece's location on the board.

2.2.6. Cannon (C)

The Cannon is unique in Xiàngqí. It is similar to the Chariot, but has a bit of a strange behavior in that it must jump over *exactly one* piece in order to capture an opposing piece.

Chinese names	*Pào* (Cannon) or *pào* (Catapult)
Chinese Symbols	
Starting position	32, 38
Movement	The Cannon moves just like the Chariot except when capturing another piece. In order to capture another piece, it must jump over *exactly one* piece and move to the spot of the piece that it captures.

2.2.7. Soldier (S)

The Soldier, as one would expect is like the Pawn in western chess. There are some differences which make it even less powerful. First, there is no promotion of pawns to other pieces. Second, when the Soldier crosses the river, it may move horizontally. Third, there is no *en passant* capture as in western chess, all captures are either straight ahead or to the side after the soldier crosses the river.

Chinese names	*Bīng* (Soldier) or *zú* (Pawn)
Chinese Symbols	
Starting position	41, 43, 45, 47, 49
Movement	Before crossing the river, the Soldier can only move forward one step. Once it crosses the river, it may move horizontally one step. It can *never* move backwards.

2.3. The starting board

Figure 2 shows the starting position of the board for the standard Xiàngqí game.

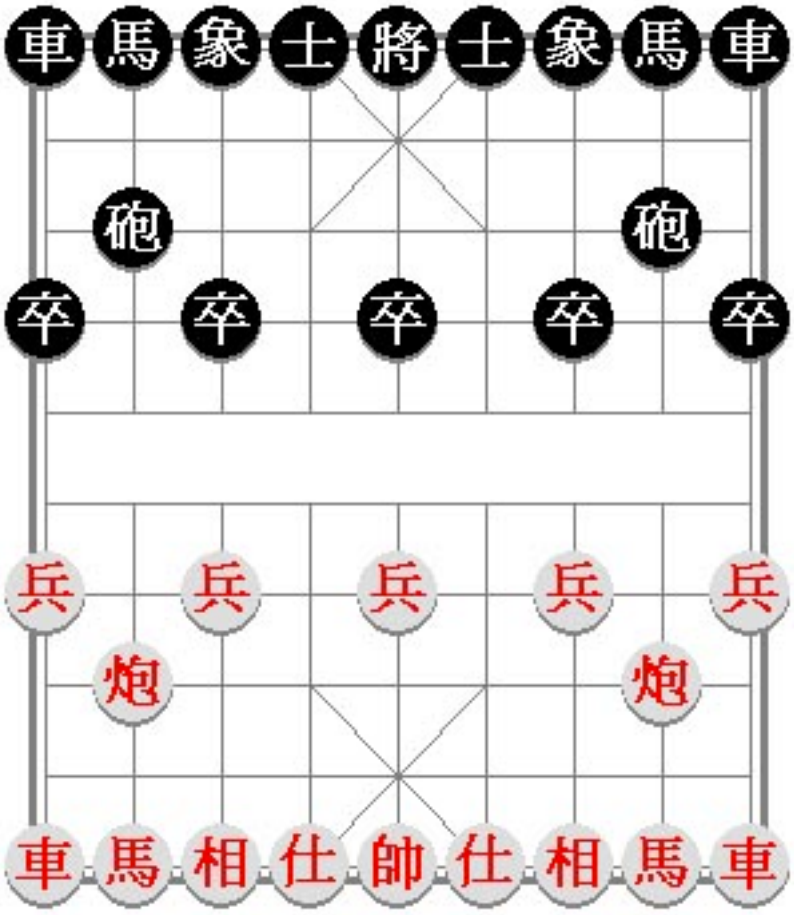

Figure 2. The starting board position [2]

2.4. Capturing

As in western chess, a piece captures by simply moving to the location of an opposing piece. The restrictions on how a piece may do this are mentioned in the piece descriptions when it is not the obvious method of simply moving to the location (specifically, see the description of the Cannon).

2.5. Game outcomes

There are three possible outcomes for a game; Red wins, Black wins, or Draw. The object of this game is to force capture of the enemy General. This may be by checkmate (he is under attack and has no means of escape) or by stalemate (he is not under immediate threat, but there is no legal, safe move). Note that this definition is different than stalemate in western chess, where stalemate results in a Draw. In Xiàngqí a stalemate results in a loss for the player who is unable to move.

When the General is being threatened with capture, he is said to be in "check," and the player must move in such a way that the General is no longer threatened. If he cannot, he has lost the game.

There is a special rule about the Generals. The two may never face each other on the same line across the board, with no intervening pieces between them. It is said that they may not *see* each other. This becomes important, especially in the end game, as the position of one General limits the movements of the other, and can protect a piece invading the enemy fortress.

If a player makes a move that leaves the opposing Generals facing each other, the "flying general" move may be executed, in which the General to move may cross the board to capture the enemy general. In practice, this rule is only used to enforce checkmate. This rule makes sense when, in tournaments, one player may cause the Generals to be facing each other. The opposing player can execute the flying general move to win. However, if the player makes a different move, the first player can then execute the move to assassinate the opponent. It is a desperation move. In our implementation we do not worry about this. If a player makes a move that has the opposing Generals facing each other, the player who made the move loses.

It is not allowed to give perpetual check, or a perpetual attack. If the game is repeating its position, the player forcing the repetition must do something else. For our version of Xiàngqí we use the following rules of perpetual check and move repetition:

> *If there is a sequence of moves where one side continually puts the other side's General in check, and if the exact same board configuration occurs within that sequence, then the side making the checks loses.*

When Xiàngqí is played in tournaments, there are several ways that the game can end up as a Draw. [4] In our implementation, we do not consider a Draw as a possible outcome, unless a version is described where some special rule identifies Draw possibilities.

2.6. Move notation

We use the notation for board locations described in Section 2.1.1. A move is represented in this document as an ordered pair, (s, d), where s is the source location (where the moving piece is at the beginning of the move) and d is the destination location (where the moving piece is at the end of the move).

An example opening move for Red would be (32, 35) that moves the left Cannon to the middle file.

Even though we use ordered pairs in this document, the coordinate ideitification implemented in the software may differ as defined in the appropriate interfaces.

2.7. Check and legal moves

There has been quite a bit of discussion about whether it is possible to move into check and what is legal when one's king is in check. There are several interpretations of this in the Xiàngqí literature. Here are some guidelines that we will adhere to if it is appropriate for the version of the game; that is, these are considered standard behavior. (For CS4233-C17, these will be in effect for Gamma Xiàngqí and beyond.)

- A player may not move the General to a position where the General is in check. If this is done, the implementation will return an INVALD move result.
- A player may not expose the General to check. That is if a player tries to make a move that would leave the General in check, the implementation will return an INVALID result.
- If a move puts a player in checkmate, the game should return a WIN for the player who put the opponent in checkmate and the game is over.
- If a General is in check (not checkmate), the player must make a move that removes the check either by moving the General to safety or blocking the check through capture of interposing a piece between the attacker and the General. Any other attemped move will result in the implementation returning INVALID.

The above cases indicate that there is no need for the *Flying General* move since a player may not place his or her own General into check, which is the necessary to employ the Flying General.

3. The basic implementation

Students will implement a Xiàngqí game manager that is capable of supporting several versions of Xiàngqí. The development of Xiàngqí evolves through several states, starting with Alpha Xiàngqí, which has minimal features but implements the main interfaces. Using Alpha Xiàngqí as the first building block, additional versions add more features until a full Xiàngqí game is implemented. Once this milestone is reached, it is possible to produce variations on standard Xiàngqí that exhibit interesting game features and offer opportunities for applying design patterns and principles taught in the course. Studnents are only responsible for implementing the versions assigned for the current term.

The implementation of Xiàngqí that students develop during the term will support playing a complete game, without any artificial intelligence (AI) player. For the final project deliverable however, the students may need develop some sort of AI player that can play a legal game of one or more versions of Xiàngqí. The skill of the AI player is not a gradable item, but it must play a legal game. I will provide a Xiàngqí tournament director application that will use student players and have them play against each other in a fun tournament at the end of the term. This is explained further in this document.

For this class I have provided students with a starting Eclipse project that should be used as the basis for code. The starter project has mostly interfaces and enumerations. Figure 3 shows the relationship of those elements in the starter project.

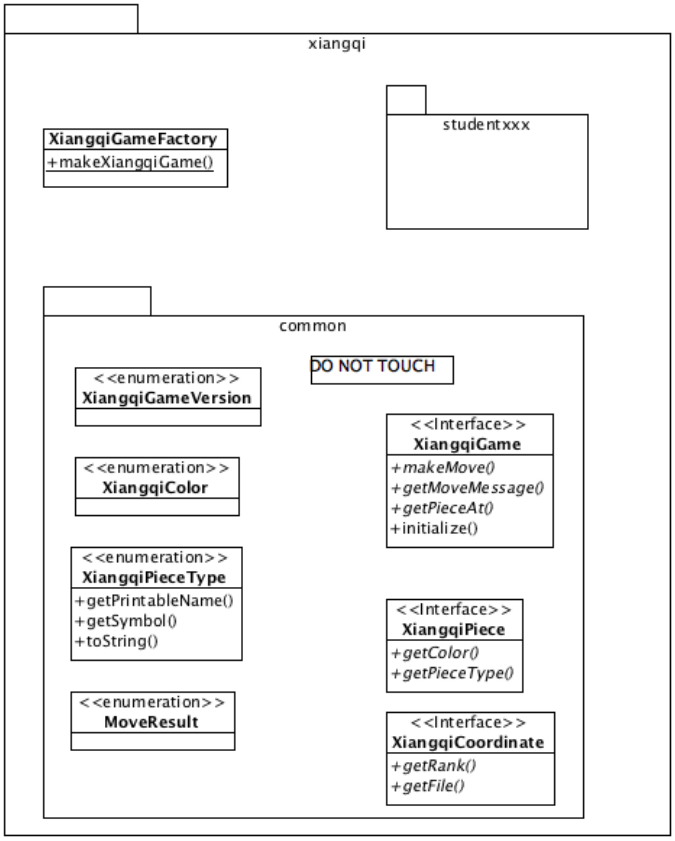

Figure 3. Xiàngqí starter code

Notice that there is a DO NOT TOUCH sign on the xiangqi.common package. You may not make any modifications, whatsoever to the Java files in this package.

3.1. Package structure

All student code will be placed in packages that are rooted at a package that is unique to the student, or students. In **Error! Reference source not found.** this is shown as xiangqi.studentxxx. If you are working on the project alone, you should rename this, replacing xxx with your WPI user name. So, if your user name is jj3927, you would rename the package to xiangqi.studentjj3927. If the project can be worked on with a partner, you should replace xxxx with some unique string that identifies both of the team members (perhaps three initials for each student). For purpose of this documentation, we will just refer to this package as xiangqi.studentx. All student code will be in subdirectories that are either children of the xianqi.studentx package or a package that is a descendent of that package.

Instructor supplied code and tests will always be in the other packages and never include code that is in the xiangqi.studentx package tree.

Figure 4 shows the package structure in Eclipse for the starting project. Notice that there is a README file that tells you what you need to do after importing the project into Eclipse.

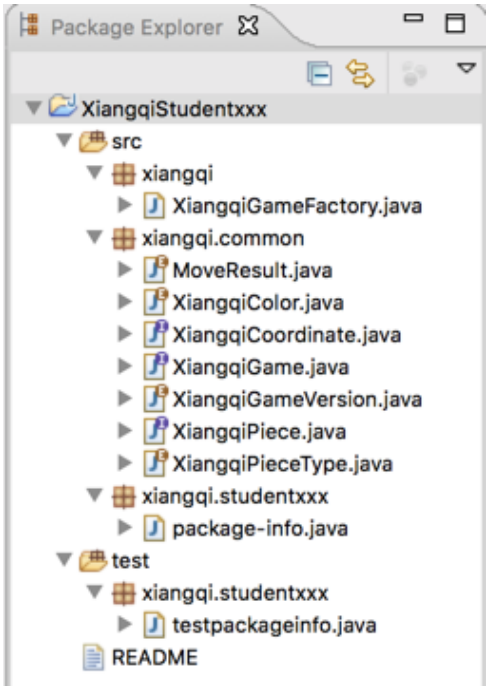

Figure 4. starter code package structure

4. Xiàngqí Versions and Variations

Students are expected to evolve the design and implementation of Xiàngqí. In order to do this they will implement different variations, or releases, as the course progresses. Variations gain complexity, from very simple—even silly and trivial—versions to complete support for different Xiàngqí versions and rule sets. This section describes the specifics of each version. The final form of this document will contain many more variations of Xiàngqí than students will be able to implement in a single term. The instructor will select the appropriate versions for each term that Hanto is used as the subject of the project.

Each version has its own set of rules. If not specified explicitly, the rules in [1] and [5] apply.

4.1. Alpha Xiàngqí

Alpha Xiàngqí is a version of the game that gets the student started on developing the game with the supplied code base. Using TDD, the implementation should be quite simple.

4.1.1. Pieces and capabilities

Xiàngqí Only one type of piece is used for Alpha Xiàngqí —the General. The board consists of 2 x 2 squares (making a 3 x 3 space). The fortress for each player consists of rank 0. The configurations at the beginning of the game is shown in Figure 5.

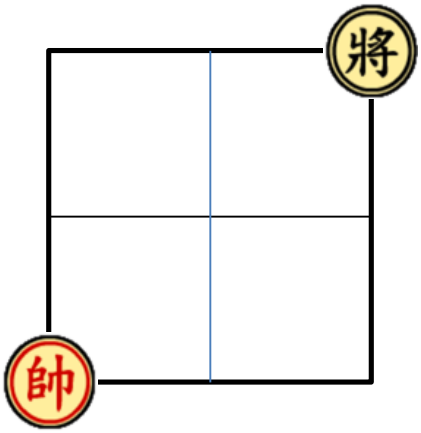

Figure 5. Alpha Xiàngqí setup

4.1.2. The rule set

There is just one complete turn in Alpha Xiàngqí. Red moves first and moves the General from (1, 1) to (1, 2). Now, the only thing Black can do is make the same move. This causes the Generals to face each other, and Red wins.

Your implementation should do the following:

- Make sure that the correct and/or valid coordinates are passed in as arguments to makeMove().
- Return the correct results from invoking makeMove(). The first time should return OK and the second should return RED_WINS.
- If an illegal move is attempted makeMove() should return ILLEGAL and have an appropriate message describing the problems when getMoveMessage() is called. The game ends as a draw.
- Any call to the getPieceAt() method should return a piece with NONE for the piece type and color.

4.1.3. Required deliverables

All of your code will be rooted in the **xiangi.studentx** package. The only code that you will change that is not in your **studentx** package is adding code to the XiangiGameFactory to produce the Alpha Xiàngqí game instance.

You will write the code using Test-Driven Development and all of your test code will be placed under the **test** source folder. You should have at least 90% code coverage in the implementation code (the **xiangqi.studentx** package, excepting interfaces and enumerations).

You are not required to deliver this code as an assignment on its own. It will be delivered with the first project assignment for Beta Xiàngqí.

4.2. Beta Xiàngqí

Beta Xiàngqí adds a little bit more capability to the game. The board consists of 4 x 4 squares, yielding a 5 x 5 playing space.

4.2.1. Pieces and capabilities

For this game, each player has a General, a Soldier, two Advisors, and two Chariots as shown in Figure 6. The capabilities of the Chariots are as specified in the standard rules. The Soldier can only move forward one space. The Advisors can move one space diagonally anywhere on the board. The King can only move horizontally between locations 12 and 14, one space at a time.

Your implementation should do the following:

- Make sure that the correct and/or valid coordinates are passed in as arguments to makeMove(). If not, return a result of ILLEGAL.
- Return the correct results from invoking makeMove().
- If an illegal move is attempted makeMove() should return ILLEGAL and have an appropriate message describing the problems when getMoveMessage() is called. The player is allowed to submit a new move.
- Any call to the getPieceAt() method should return the appropriate piece and color of the piece at that location, or the piece with color = NONE, type = NONE if the location is empty.

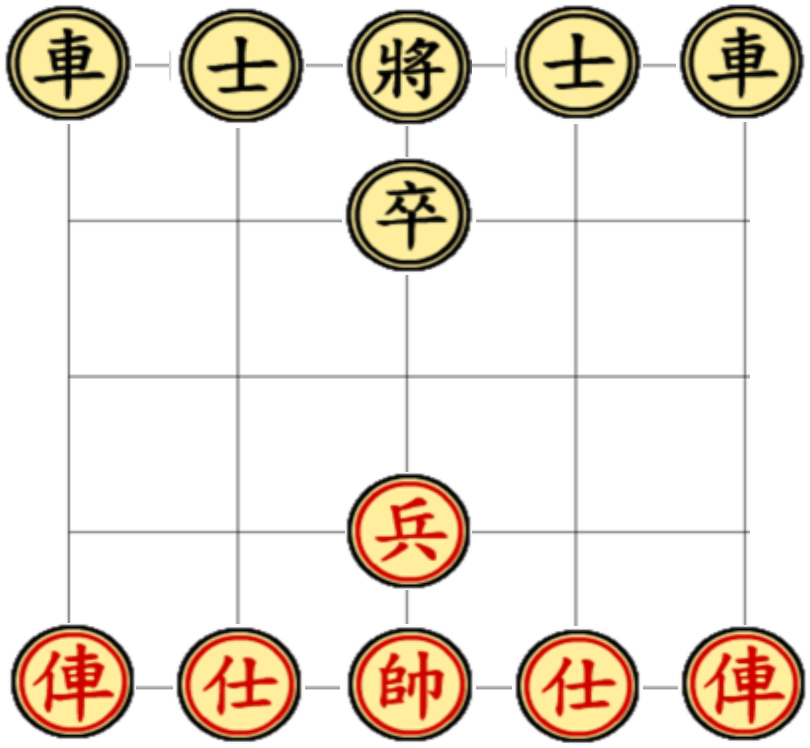

Figure 6. Beta Xiàngqí setup

4.2.2. The rule set

There are a total of ten (10) complete moves in this game. This means that Red moves ten or less times and Black move 10 or less times. If, at the end of Black's 10th move there is no winner, the game results in a draw.

4.2.3. Required deliverables

You will modify the XiangqiGameFactory to return an instance of a Beta Xiàngqí. The implementation for your game will be rooted in your **xiangqi.studentx**. You should organize your code such that the code that is unique to versions reside in packages that identify them as such, for instance all Alpha Xiàngqí code might reside in a **xiangqi.studentx.alpha** package. You might choose any scheme, but it should be consistent and obvious to a reviewer how you have organized your code.

You will write the code using TDD and all of your test code will be placed under the **test** source folder. You should have at least 92% code coverage in the implementation code, as in Alpha Xiàngqí.

This version of your project will be turned in for grading. It will contain your Alpha Xiàngqí and Beta Xiàngqí code.

4.3. Gamma Xiàngqí

Gamma Xiàngqí is the first version that uses the standard board, including the river and the Generals' palaces.

4.3.1. Pieces and capabilities

This version utilizes all of the pieces in the standard game, except for the Horse and the Cannon. Each piece in this game moves as in the standard game.

Your implementation should do the following:

- Make sure that the correct and/or valid coordinates are passed in as arguments to makeMove(). If not, return a result of ILLEGAL.
- Return the correct results from invoking makeMove().
- If an illegal move is attempted makeMove() should return ILLEGAL and have an appropriate message describing the problems when getMoveMessage() is called. The player is allowed to submit a new move.

Any call to the getPieceAt() method should return the appropriate piece and color of the piece at that location, or the piece with color = NONE, type = NONE if the location is empty. If the coordinates given to getPieceAt() are invalid, throw a CompletionException. This is a RuntimeException and does not need to be declared as being thrown.

4.3.2. The rule set

There is a move limit in this game of 25 complete moves. If, at the end of the 25th move, and there is no winner, the game results in a draw.

No checks are made for repeated moves. This will be added in a future version.

4.3.3. Required deliverables

You will modify the XiangqiGameFactory to return an instance of a Gamma Xiàngqí. The implementation for your game will be rooted in your **xiangqi.studentx**. You should organize your code such that the code that is unique to versions, if any, reside in packages that identify them as such. You might choose any scheme, but it should be consistent and obvious to a reviewer how you have organized your code.

You will write the code using TDD and all of your test code will be placed under the **test** source folder. You should have at least 95% code coverage in the implementation code, as in previous versions.

This version of your project will be turned in for grading. It will contain your code for all versions implemented thus far along with your tests.

Bibliography

[1] Wikipedia, "Xiangxi," 15 November 2016. [Online]. Available: https://en.wikipedia.org/wiki/Xiangqi. [Accessed 29 November 2016].

[2] Ancient Chess, "How to Play Chinese Chess Xiangqi," [Online]. Available: http://ancientchess.com/page/play-xiangqi.htm. [Accessed 5 December 2016].

[3] chesslight.com, "Chinese Chess," 2016. [Online]. Available: http://www.chesslight.com/cchess.php. [Accessed 5 December 2016].

[4] Hong Kong Chinese Chess Association, "Asia XiangQi Ruels," [Online]. Available: http://www.clubxiangqi.com/rules/asiarule.htm. [Accessed 25 December 2016].

[5] Xiangqi in English.com, "XiangQi in English," 2016. [Online]. Available: http://www.xqinenglish.com/introduction_to_xiangqi_chessboard.html. [Accessed 29 November 2016].

[6] H. B. Christensen, Flexible, Reliable Software Using Patterns and Agile Development, Boca Raton, FL: Chapman & Hall/CRC, 2010.

[7] E. Falkener, Games Ancient and Oriental and How to Play Them, New York, NY: Dover Publications, Inc., 1961, pp. 144-155.

[8] Springfrog, "Play Chinese Chess On Line Against the Computer with Western Pieces," [Online]. Available: http://www.springfrog.com/games/chess/chinese/. [Accessed 25 December 2016].