Group 13

Author: 陈怡珺

Software VALIDATION

HuaRongDao System

Table of Contents

[System Architecture 2](#_Toc10449121)

[T1: Unit Test 2](#_Toc10449122)

[T1.1: Chessboard Unit Test 2](#_Toc10449123)

[T2: Integration Test 12](#_Toc10449124)

T2.1: ChessBoard+SelfMenu Integration…………………………………………………………………………..……………12

T2.2: ChessBoard+MainInterface Integration…………………………………………………………………….…………..13

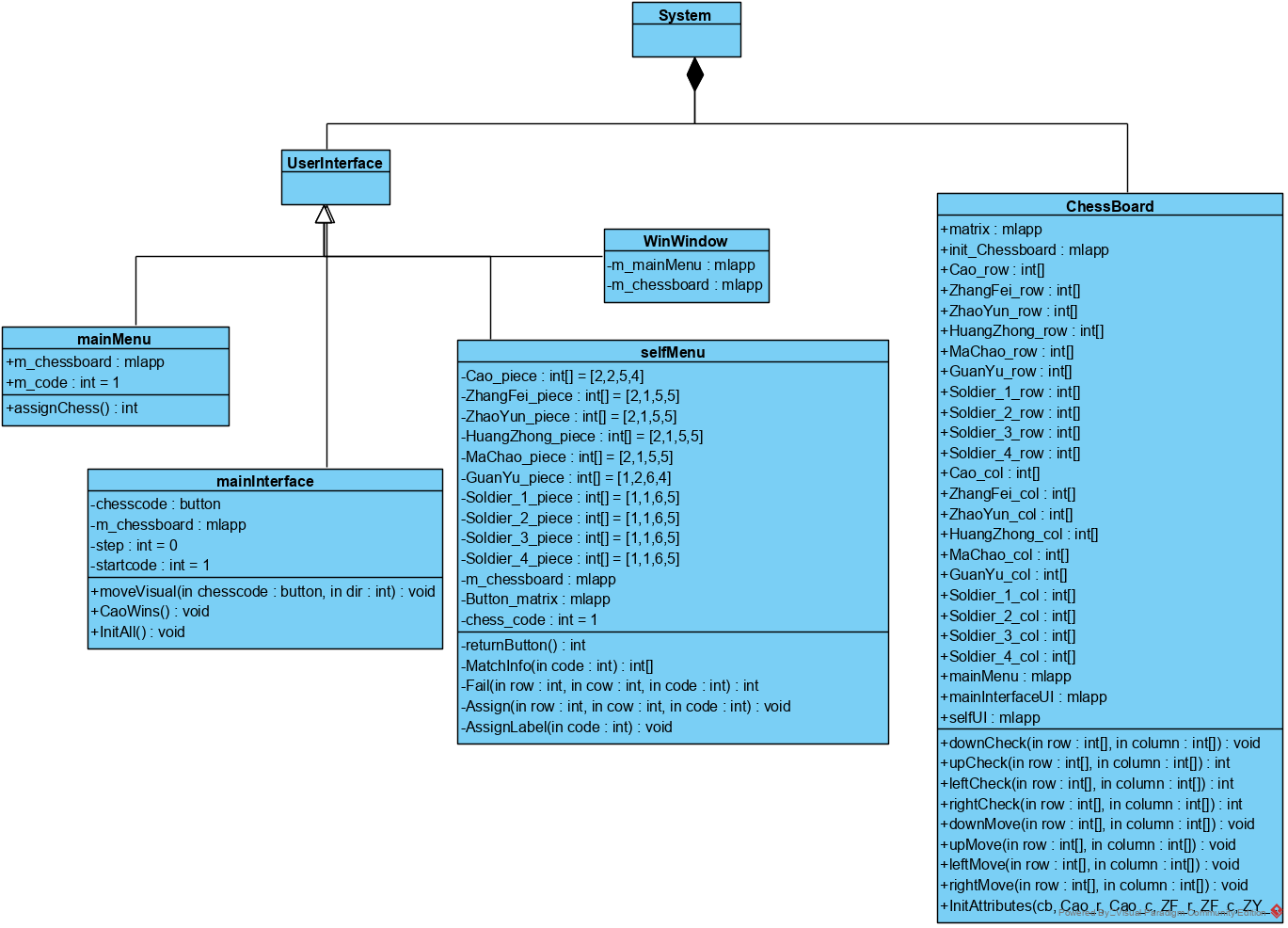
T2.3: ChessBoard+MainMenu Integration……………………………………………………………………..………………14

[T3: Functional Test……………………………………………………………………………………………………………..…………….14](#_Toc10449125)

T4:Uppaal test………………………………………………………………………………………………………………………………..25

## System Architecture

The system architecture is shown below:



## T1: Unit Test

### T1.1: Chessboard Unit Test

T1.1.1: Test downCheck()

function is\_dirD = downCheck(board,row,column) Tcover1.1.1

is\_dirD = false;

if row(end)+1 <= 5

if board.matrix(row(end)+1,column) == zeros(1,length(column))

is\_dirD = true;

end

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.1.1 |
| Coverage Item | Tcover1.1.1 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirD = CD.downCheck(CD.Cao\_row,CD.Cao\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirD ==false |

|  |  |
| --- | --- |
|  | Test Case T1.1.1.2 |
| Coverage Item | Tcover1.1.1.1 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirD = CD.downCheck(CD.Soldier\_1\_row,CD.Soldier\_1\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirD ==true |

* Test coverage: 2/2=100%
* Test result: 2 passed

T1.1.2: Test upCheck()

function is\_dirU = upCheck(board,row,column) Tcover1.1.2

is\_dirU = false;

if row(1)-1 >= 1

if board.matrix(row(1)-1,column) == zeros(1,length(column))

is\_dirU = true;

end

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.2.1 |
| Coverage Item | Tcover1.1.2 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,0,1,1;1,1,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=5;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirU = CD.upCheck(CD.Cao\_row,CD.Cao\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirU ==false |

|  |  |
| --- | --- |
|  | Test Case T1.1.2.2 |
| Coverage Item | Tcover1.1.2 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,0,1,1;1,1,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=5;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirU = CD.upCheck(CD.Soldier\_1\_row,CD.Soldier\_1\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirU ==true |

* Test coverage: 2/2=100%
* Test result: 2 passed

T1.1.3: Test rightCheck()

function is\_dirR = rightCheck(board,row,column) Tcover1.1.3

is\_dirR = false;

if column(end)+1 <= 4

if board.matrix(row,column(end)+1) == zeros(length(row),1)

is\_dirR = true;

end

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.3.1 |
| Coverage Item | Tcover1.1.3 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirR = CD.rightCheck(CD.Soldier\_3\_row,CD.Soldier\_3\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirR ==true |

|  |  |
| --- | --- |
|  | Test Case T1.1.3.2 |
| Coverage Item | Tcover1.1.3 |
| Input | CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirR = CD.rightCheck(CD.Soldier\_4\_row,CD.Soldier\_4\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirR ==false |

* Test coverage: 2/2=100%
* Test result: 2 passed

T1.1.4: Test leftCheck()

function is\_dirL = leftCheck(board,row,column) Tcover1.1.4

is\_dirL = false;

if column(1)-1 >= 1

if board.matrix(row,column(1)-1) == zeros(length(row),1)

is\_dirL = true;

end

end

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.4.1 |
| Coverage Item | Tcover1.1.4 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,0,1;1,0,1,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=5;  CD.Soldier\_3\_row=5;  Cd.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirL = CD.leftCheck(CD.Cao\_row,CD.Cao\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirL ==false |

|  |  |
| --- | --- |
|  | Test Case T1.1.4.2 |
| Coverage Item | Tcover1.1.4 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,0,1;1,0,1,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=5;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  is\_dirL = CD.leftCheck(CD.Soldier\_2\_row,CD.Soldier\_2\_col); |
| State | CD = ChessBoard; |
| Expected Output | is\_dirL ==false |

* Test coverage: 2/2=100%
* Test result: 2 passed

T1.1.5: Test downMove ()

function downMove(board,row,column) Tcover1.1.5

board.matrix(row(1),column) = zeros(1,length(column));

board.matrix(row + 1, column) = ones(length(row),length(column));

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.5 |
| Coverage Item | Tcover1.1.5 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  CD.downMove(CD.Soldier\_1\_row,CD.Soldier\_1\_col); |
| State | CD = ChessBoard; |
| Expected Output | CD.matrix=[1,1,1,1;1,1,1,1;1,1,1,1;1,0,1,1;1,1,0,1] |

* Test coverage: 1/1=100%
* Test result: 1 passed

T1.1.6: Test upMove ()

function upMove(board,row,column) Tcover1.1.6

board.matrix(row(end),column) = zeros(1,length(column));

board.matrix(row - 1, column) = ones(length(row),length(column));

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.6 |
| Coverage Item | Tcover1.1.6 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,0,1,1;1,1,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=5;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  CD.upMove(CD.Soldier\_1\_row,CD.Soldier\_1\_col); |
| State | CD = ChessBoard; |
| Expected Output | CD.matrix=[1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1] |

* Test coverage: 1/1=100%
* Test result: 1 passed

T1.1.7: Test leftMove ()

function leftMove(board,row,column) Tcover1.1.7

board.matrix(row,column(end)) = zeros(length(row),1);

board.matrix(row,column - 1) = ones(length(row),length(column));

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.7 |
| Coverage Item | Tcover1.1.7 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,0,1;1,0,1,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=5;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  CD.leftMove(CD.Soldier\_2\_row,CD.Soldier\_2\_col); |
| State | CD = ChessBoard; |
| Expected Output | CD.matrix=[1,1,1,1;1,1,1,1;1,1,1,1;1,1,0,1;1,1,0,1] |

* Test coverage: 1/1=100%
* Test result: 1 passed

T1.1.8: Test rightMove ()

function rightMove(board,row,column) Tcover1.1.8

board.matrix(row,column(1)) = zeros(length(row),1);

board.matrix(row,column + 1) = ones(length(row),length(column));

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.8 |
| Coverage Item | Tcover1.1.8 |
| Input | CD.matrix = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1];  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4;  CD.rightMove(CD.Soldier\_3\_row,CD.Soldier\_3\_col); |
| State | CD = ChessBoard; |
| Expected Output | CD.matrix=[1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;0,1,0,1] |

* Test coverage: 1/1=100%
* Test result: 1 passed

T1.1.9: Test InitAttribute()

function rightMove(board,row,column) Tcover1.1.8

board.matrix(row,column(1)) = zeros(length(row),1);

board.matrix(row,column + 1) = ones(length(row),length(column));

end

* Coverage Criteria: Statement coverage
* Test case

|  |  |
| --- | --- |
|  | Test Case T1.1.9 |
| Coverage Item | Tcover1.1.9 |
| Input | CD.InitAttributes([1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1]  ,[1,2],[2,3],[1,2],1,[1,2],4,[3,4],1,[3,4],4,3,[2,3],  4,2,4,3,5,1,5,4); |
| State | CD = ChessBoard; |
| Expected Output | CD.matrix=  [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1;1,0,0,1]  CD.Cao\_row=[1,2];  CD.ZhangFei\_row=[1,2];  CD.ZhaoYun\_row=[1,2];  CD.HuangZhong\_row=[3,4];  CD.MaChao\_row=[3,4];  CD.GuanYu\_row=3;  CD.Soldier\_1\_row=4;  CD.Soldier\_2\_row=4;  CD.Soldier\_3\_row=5;  CD.Soldier\_4\_row = 5;  CD.Cao\_col=[2,3];  CD.ZhangFei\_col=1;  CD.ZhaoYun\_col=4;  CD.HuangZhong\_col=1;  CD.MaChao\_col=4;  CD.GuanYu\_col=[2,3];  CD.Soldier\_1\_col=2;  CD.Soldier\_2\_col=3;  CD.Soldier\_3\_col=1;  CD.Soldier\_4\_col=4; |

* Test coverage: 1/1=100%
* Test result: 1 passed

## T2: Integration Test

T2.1: ChessBoard+SelfMenu Integration

T2.: Test no Overlap when placing

|  |  |
| --- | --- |
|  | Test Case T2.1.4 |
| Input | press(testCase.App.B2\_4);  press(testCase.App.B5\_2);  press(testCase.App.B5\_4); |
| State | testCase.chessboard = ChessBoard;  testCase.App = selfMenu(testCase.chessboard);  testCase.chessboard.selfUI = testCase.App; |
| Expected Output | testCase.App.CurrentRole.Text  ='当前角色：曹操'); |

* Test coverage: 1/1=100%
* Test result: 1 passed

T2.2: ChessBoard+MainInterface Integration

T2.2.1: Test chess will not move out of chessboard

|  |  |
| --- | --- |
|  | Test Case T2.2.1 |
| Input | testCase.press(testCase.maininterface.Soldier\_4);  testCase.press(testCase.maininterface.Down); |
| State | testCase.chessboard = ChessBoard;  testCase.App = selfMenu(testCase.chessboard);  testCase.chessboard.selfUI = testCase.App;  testCase.press(testCase.App.B3\_2);  testCase.press(testCase.App.B1\_1);  testCase.press(testCase.App.B1\_4);  testCase.press(testCase.App.B4\_1);  testCase.press(testCase.App.B3\_4);  testCase.press(testCase.App.B1\_2);  testCase.press(testCase.App.B2\_2);  testCase.press(testCase.App.B2\_3);  testCase.press(testCase.App.B3\_1);  testCase.press(testCase.App.B5\_4);  testCase.press(testCase.App.StartGame);  testCase.maininterface =  testCase.chessboard.mainInterfaceUI; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,5);  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4); |

* Test coverage: 1/1=100%
* Test result: 1 passed

T2.2.2: Test chess will not overlap in chessboard

|  |  |
| --- | --- |
|  | Test Case T2.2.3 |
| Input | press(testCase.maininterface.Soldier\_4);  press(testCase.maininterface.Up); |
| State | testCase.chessboard = ChessBoard;  testCase.App = selfMenu(testCase.chessboard);  testCase.chessboard.selfUI = testCase.App;  testCase.press(testCase.App.B3\_2);  testCase.press(testCase.App.B1\_1);  testCase.press(testCase.App.B1\_4);  testCase.press(testCase.App.B4\_1);  testCase.press(testCase.App.B3\_4);  testCase.press(testCase.App.B1\_2);  testCase.press(testCase.App.B2\_2);  testCase.press(testCase.App.B2\_3);  testCase.press(testCase.App.B3\_1);  testCase.press(testCase.App.B5\_4);  testCase.press(testCase.App.StartGame);  testCase.maininterface = testCase.chessboard.mainInterfaceUI; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,5);  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4); |

* Test coverage: 1/1=100%
* Test result: 1 passed

T2.3: ChessBoard+MainMenu Integration

T2.3.2: Test jumping to selfMenu

|  |  |
| --- | --- |
|  | Test Case T2.3.2 |
| Input | testCase.press(testCase.App1.User);  testCase.App = testCase.chessboard.selfUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | class(testCase.chessboard.selfUI)=  class(selfMenu(testCase.chessboard) |

* Test coverage: 1/1=100%
* Test result: 1 passed

## T3: Functional Test

T3.1: Use Case “Set Chess and play to win”

|  |  |
| --- | --- |
|  | Test Case T2.2.2 |
| Input | testCase.press(testCase.App.B3\_2);  testCase.press(testCase.App.B1\_1);  testCase.press(testCase.App.B1\_4);  testCase.press(testCase.App.B4\_1);  testCase.press(testCase.App.B3\_4);  testCase.press(testCase.App.B1\_2);  testCase.press(testCase.App.B2\_2);  testCase.press(testCase.App.B2\_3);  testCase.press(testCase.App.B3\_1);  testCase.press(testCase.App.B5\_4);  testCase.press(testCase.maininterface.Cao); testCase.press(testCase.maininterface.Down); |
| State | testCase.chessboard =  ChessBoard;  testCase.App =  selfMenu(testCase.chessboard);  testCase.chessboard.selfUI  = testCase.App;  testCase.maininterface =  testCase.chessboard.mainInterfaceUI; |
| Expected Output | WinWindow.mlapp |

T3.2: Use Case “Choose a basic form and start game”

|  |  |
| --- | --- |
|  | Test Case T2.3.1.1 |
| Input | testCase.choose(testCase.App1.DropDown,'横刀立马');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 0, 0, 1])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,5)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,5)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,1)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,1)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,4)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,4)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.2 |
| Input | testCase.choose(testCase.App1.DropDown,'左右步兵');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;0, 1, 1, 0;1, 1, 1, 1;1, 1, 1, 1])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,5)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,5)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,5)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,1)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,4)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,2)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,3)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.3 |
| Input | testCase.choose(testCase.App1.DropDown,'兵分三路');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 0, 0, 1])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[2,3])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[4,5])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[4,5])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,4)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,1)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,4)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,1)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,4)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,3) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.4 |
| Input | testCase.choose(testCase.App1.DropDown,'兵来将阻');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1; 0, 1, 1, 0])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,5)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,4)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[2,3])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[2,3])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,4)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,1)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,4)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.5 |
| Input | testCase.choose(testCase.App1.DropDown,'层层设防');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;0, 1, 1, 0])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,4)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,5)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,2)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,1)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,2)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.6 |
| Input | testCase.choose(testCase.App1.DropDown,'插翅难飞');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 0, 1;1, 1, 0, 1;1, 1, 1, 1])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,5)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,5)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,4)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,1)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,4)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,2)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.7 |
| Input | testCase.choose(testCase.App1.DropDown,'过五关');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;0, 1, 1, 0])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,3)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,3)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,4)  testCase.verifyEqual(testCase.chessboard.MaChao\_row,4)  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,5)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,2)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.MaChao\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.8 |
| Input | testCase.choose(testCase.App1.DropDown,'近在咫尺');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;0, 0, 1, 1])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[4,5])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,3)  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,3)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,2)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,3)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.MaChao\_col,4)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,4) |

|  |  |
| --- | --- |
|  | Test Case T2.3.1.9 |
| Input | testCase.choose(testCase.App1.DropDown,'前阻后挡');  testCase.press(testCase.App1.Start);  testCase.App = testCase.chessboard.mainInterfaceUI; |
| State | testCase.chessboard = ChessBoard;  testCase.App1 = mainMenu(testCase.chessboard);  testCase.chessboard.mainMenu = testCase.App1; |
| Expected Output | testCase.verifyEqual(testCase.chessboard.matrix,[1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;1, 1, 1, 1;0, 1, 1, 0])  testCase.verifyEqual(testCase.chessboard.Cao\_row,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_row,5)  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.HuangZhong\_row,[3,4])  testCase.verifyEqual(testCase.chessboard.MaChao\_row,[2,3])  testCase.verifyEqual(testCase.chessboard.GuanYu\_row,1)  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_row,2)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_row,3)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_row,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_row,4)  testCase.verifyEqual(testCase.chessboard.Cao\_col,[1,2])  testCase.verifyEqual(testCase.chessboard.ZhangFei\_col,[2,3])  testCase.verifyEqual(testCase.chessboard.ZhaoYun\_col,1)  testCase.verifyEqual(testCase.chessboard.HuangZhong\_col,2)  testCase.verifyEqual(testCase.chessboard.MaChao\_col,3)  testCase.verifyEqual(testCase.chessboard.GuanYu\_col,[3,4])  testCase.verifyEqual(testCase.chessboard.Soldier\_1\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_2\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_3\_col,4)  testCase.verifyEqual(testCase.chessboard.Soldier\_4\_col,3) |

* Test coverage: 9/9=100%
* Test result: 9 passed

T3.3: Use Case “Return to MainMenu after win”

|  |  |
| --- | --- |
|  | Test Case T2.4.1 |
| Input | testCase.press(testCase.App.Button); |
| State | testCase.chessboard = ChessBoard;  testCase.chessboard.matrix=[1,1,1,1;  1,1,1,1;  1,1,1,1;  1,0,0,1;  1,1,1,1;];  testCase.chessboard.Cao\_row=[4,5];  testCase.chessboard.Cao\_col=[2,3];  testCase.chessboard.ZhangFei\_row=[1,2];  testCase.chessboard.ZhangFei\_col=1;  testCase.chessboard.ZhaoYun\_row=[1,2];  testCase.chessboard.ZhaoYun\_col=4;  testCase.chessboard.HuangZhong\_row=[4,5];  testCase.chessboard.HuangZhong\_col=1;  testCase.chessboard.MaChao\_row=[3,4];  testCase.chessboard.MaChao\_col=4;  testCase.chessboard.GuanYu\_row=1;  testCase.chessboard.GuanYu\_col=[2,3];  testCase.chessboard.Soldier\_1\_row=2;  testCase.chessboard.Soldier\_1\_col=2;  testCase.chessboard.Soldier\_2\_row=2;  testCase.chessboard.Soldier\_2\_col=3;  testCase.chessboard.Soldier\_3\_row=3;  testCase.chessboard.Soldier\_3\_col=1;  testCase.chessboard.Soldier\_4\_row=5;  testCase.chessboard.Soldier\_4\_col=4;  testCase.App = WinWindow(mainInterface(testCase.chessboard,1),  testCase.chessboard);  testCase.chessboard.WinUI = testCase.App;  %testCase.press(testCase.maininterface.); |
| Expected Output | class(testCase.chessboard.mainMenu)  =class(mainMenu(testCase.chessboard) |

* Test coverage: 1/1=100%
* Test result: 1 passed

T3.4: Use Case “Leave after win”

|  |  |
| --- | --- |
|  | Test Case T2.4.2 |
| Input | testCase.press(testCase.App.Button\_2); |
| State | testCase.chessboard = ChessBoard;  testCase.chessboard.matrix=[1,1,1,1;  1,1,1,1;  1,1,1,1;  1,0,0,1;  1,1,1,1;];  testCase.chessboard.Cao\_row=[4,5];  testCase.chessboard.Cao\_col=[2,3];  testCase.chessboard.ZhangFei\_row=[1,2];  testCase.chessboard.ZhangFei\_col=1;  testCase.chessboard.ZhaoYun\_row=[1,2];  testCase.chessboard.ZhaoYun\_col=4;  testCase.chessboard.HuangZhong\_row=[4,5];  testCase.chessboard.HuangZhong\_col=1;  testCase.chessboard.MaChao\_row=[3,4];  testCase.chessboard.MaChao\_col=4;  testCase.chessboard.GuanYu\_row=1;  testCase.chessboard.GuanYu\_col=[2,3];  testCase.chessboard.Soldier\_1\_row=2;  testCase.chessboard.Soldier\_1\_col=2;  testCase.chessboard.Soldier\_2\_row=2;  testCase.chessboard.Soldier\_2\_col=3;  testCase.chessboard.Soldier\_3\_row=3;  testCase.chessboard.Soldier\_3\_col=1;  testCase.chessboard.Soldier\_4\_row=5;  testCase.chessboard.Soldier\_4\_col=4;  testCase.App = WinWindow(mainInterface(testCase.chessboard,1),  testCase.chessboard);  testCase.chessboard.WinUI = testCase.App;  %testCase.press(testCase.maininterface.); |
| Expected Output | No App left |

* Test coverage: 1/1=100%
* Test result: 1 passed

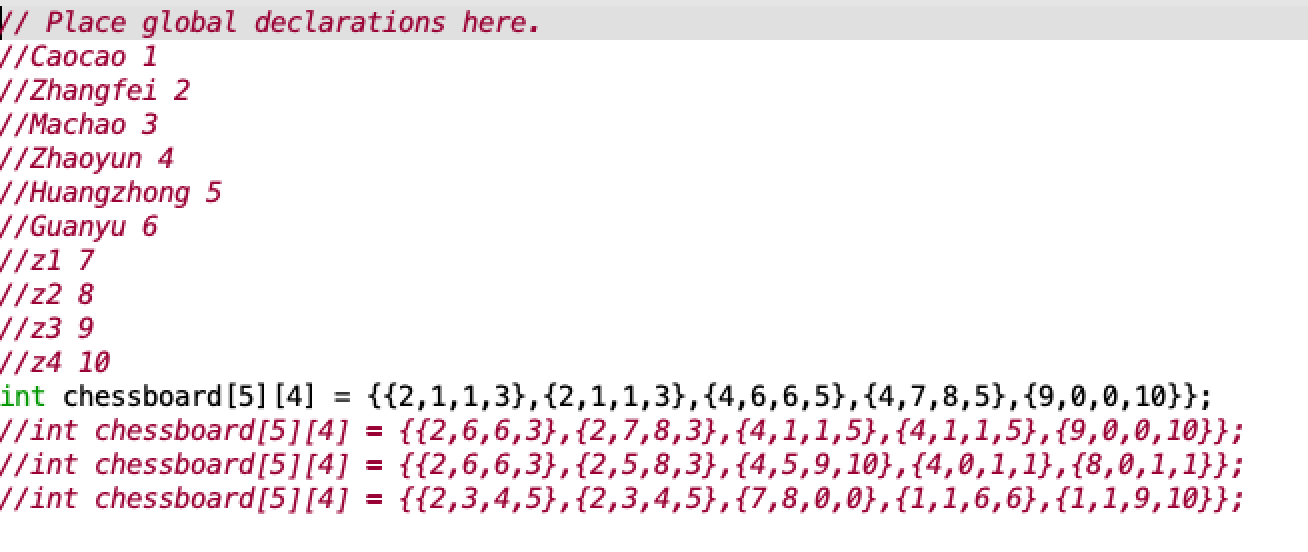
## T1: Uppaal test

T1.1: Check available of win

Statement：

1 stands for Caocao, 2 stands for Zhangfei, 3 stands for Machao, 4 stands for Zhaoyun, 5 stands for Huangzhong, 6 stands for Guanyu, 7 stands for zu\_1, 8 stands for zu\_2, 9 stands for zu\_3, 10 stands for zu\_4.

Chessboard is a matrix of 5\*4 to show the position of every chess.



Model: One node presents the state that Caocao not win.

Win node presents the state that Caocao runout.

Forty urgent node to check whether chess is available to move up/down/left/right(U/D/L/R) and update data of chessboard moved.

If the initial chessboard can have a solution to win, after searching, it will finally arrive ‘Win’ node.

