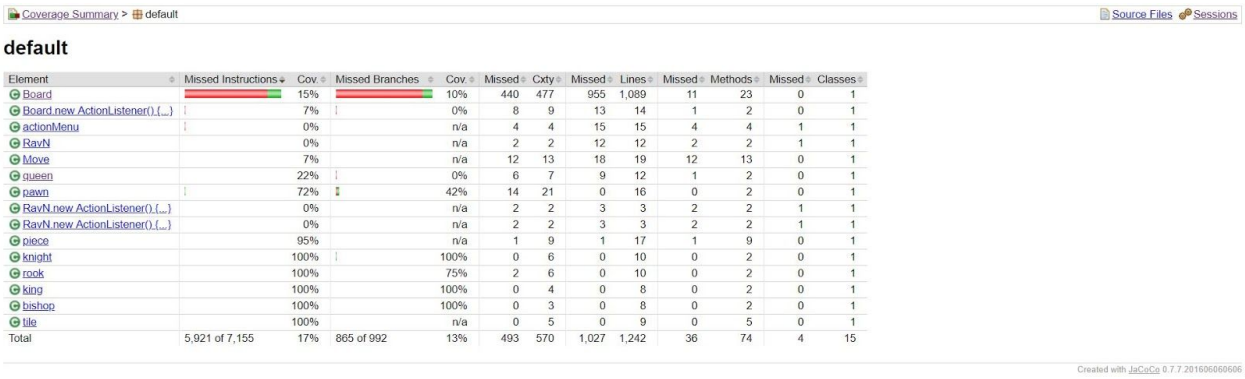
Alex Li, Mira Pratt
4/29/18
EECS 393

Functional Testing report

Unit Testing

We make use of JUnit automated testing to run unit tests where appropriate. We have enclosed code coverage measurement from those unit tests.

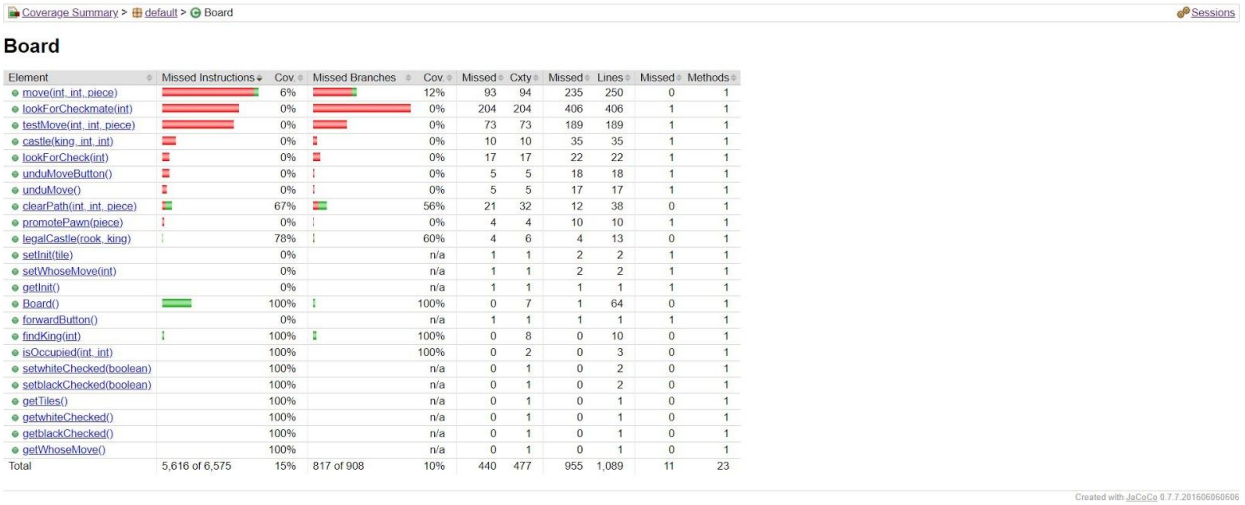


Coverage Summary > default — Source Files — Sessions

default

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Board		15%		10%	440	477	955	1,089	11	23	0	1
Board.new ActionListener() {...}		7%		0%	8	9	13	14	1	2	0	1
actionMenu		0%		n/a	4	4	15	15	4	4	1	1
RavN		0%		n/a	2	2	12	12	2	2	1	1
Move		7%		n/a	12	13	18	19	12	13	0	1
queen		22%		0%	6	7	9	12	1	2	0	1
pawn		72%		42%	14	21	0	16	0	2	0	1
RavN.new ActionListener() {...}		0%		n/a	2	2	3	3	2	2	1	1
RavN.new ActionListener() {...}		0%		n/a	2	2	3	3	2	2	1	1
piece		95%		n/a	1	9	1	17	1	9	0	1
knight		100%		100%	0	6	0	10	0	2	0	1
rook		100%		75%	2	6	0	10	0	2	0	1
king		100%		100%	0	4	0	8	0	2	0	1
bishop		100%		100%	0	3	0	8	0	2	0	1
tile		100%		n/a	0	5	0	9	0	5	0	1
Total	5,921 of 7,155	17%	865 of 992	13%	493	570	1,027	1,242	36	74	4	15

Created with JaCoCo 0.7.7.201606060606



Coverage Summary > default > Board — Sessions

Board

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
move(int, int, piece)		6%		12%	93	94	235	250	0	1
lookForCheckmate(int)		0%		0%	204	204	406	406	1	1
testMove(int, int, piece)		0%		0%	73	73	189	189	1	1
castle(king, int, int)		0%		0%	10	10	35	35	1	1
lookForCheck(int)		0%		0%	17	17	22	22	1	1
unduMoveButton()		0%		0%	5	5	18	18	1	1
unduMove()		0%		0%	5	5	17	17	1	1
clearPath(int, int, piece)		67%		56%	21	32	12	38	0	1
promotePawn(piece)		0%		0%	4	4	10	10	1	1
legalCastle(rook, king)		78%		60%	4	6	4	13	0	1
setInit(tile)		0%		n/a	1	1	2	2	1	1
setWhoseMove(int)		0%		n/a	1	1	2	2	1	1
getInit()		0%		n/a	1	1	1	1	1	1
Board()		100%		100%	0	7	1	64	0	1
forwardButton()		0%		n/a	1	1	1	1	1	1
findKing(int)		100%		100%	0	8	0	10	0	1
isOccupied(int, int)		100%		100%	0	2	0	3	0	1
setwhiteChecked(boolean)		100%		n/a	0	1	0	2	0	1
setblackChecked(boolean)		100%		n/a	0	1	0	2	0	1
getTiles()		100%		n/a	0	1	0	1	0	1
getwhiteChecked()		100%		n/a	0	1	0	1	0	1
getblackChecked()		100%		n/a	0	1	0	1	0	1
getWhoseMove()		100%		n/a	0	1	0	1	0	1
Total	5,616 of 6,575	15%	817 of 908	10%	440	477	955	1,089	11	23

Created with JaCoCo 0.7.7.201606060606

Functional Tests

For features that are too cumbersome to test using unit testing, we have performed the following functional tests. As the scenarios in chess are diverse, tests are performed multiple times in differing scenarios until the tester is satisfied the test has passed or failed.

Test Name	Steps Performed	Expected Results	Actual Result	Pass/Fail	Corrected?

Start Program	-Run the RavN.exe application file	Opens a GUI with a chessboard and action bar	Opens a GUI with a chess board and action bar	Pass	N/A
Out of turn move	-Move a black piece when it is white's turn. -Do the inverse	Pieces will not move in either case	Pieces did not move in either case	Pass	N/A
Off Board move	-Select a valid piece to move and attempt to move it off board by clicking off board	Piece does nothing	Piece does nothing	Pass	N/A
Take Friendly Piece	-Attempt to take a white piece with a white piece or a black piece with a black piece	Piece does nothing	Piece does nothing	Pass	N/A
Move Pawn	- Move a pawn forward by 2 at the start -Move a pawn forward by one on the second move -Move a pawn diagonally on top of an enemy piece	In all cases, the pawn moves correctly to the designated square	In all cases the pawn moved correctly to the designated space	Pass	N/A
Move Knight	-Move a knight over pieces two	In all cases, the knight moves	In all cases, the knight moves	Pass	N/A

	squares vertically and one square horizontally -Move a knight over pieces two squares horizontally and one square vertically -move a knight to each of the eight squares it can reach once in the middle of the board and back -Perform above with Black and White Pieces	correctly to the designated square	correctly to the designated square		
Move Rook	-Move a rook vertically forwards any number of spaces -Move a rook vertically backwards any number of spaces -Move a rook horizontally left any number of spaces then right	In all cases, the rook moves correctly to the designated square	In all cases, the rook moved correctly to the designated square	Pass	N/A
Move Bishop	-Move each bishop back and forth	In all cases, the bishop	In all cases, the bishop	Pass	N/A

	along each diagonal both one square and several squares	moves correctly to the designated square	moves correctly to the designated square		
Move King	-Move king one space forward, then one space back -Move king one space sideways left then right -Move king diagonally one space up and right, down and right, up and left, down and left	In all cases, the king moves correctly to the designated square	In all cases, the king moved correctly to the designated square	Pass	N/A
Move Queen	-Perform same tests as for the rooks and the bishops	In all cases, the queen moves correctly to the designated square	In all cases, the queen moves correctly to the designated square	Pass	N/A
Illegal Move Pawn	-Move pawn forward by 2 on the second move or later -Move pawn diagonally to an empty space -Move pawn forward to take a piece -Move pawn anywhere that is not	In all cases the pawn piece will remain in it's starting position. Player turn will not change.	In all cases the pawn piece remained in it's starting position. Player turn did not change.	Pass	N/A

	one space in front of it on the second move or later -Move a pawn that is currently blocking the king from an attacking piece out of the way				
Illegal Move Knight	-Move a knight that is currently blocking the king from an attacking piece out of the way	In all cases the knight piece will remain in it's starting position. Player turn will not change.	In all cases the knight piece will remain in it's starting position. Player turn will not change.	Pass	N/A
Illegal Move Rook	-Move the rook diagonally, or to a space not diagonal, horizontal, or vertical from the rook. -Move a rook blocking the king from an attacking piece	In all cases the rook will not move, and the player move will not change.	In all cases the rook did not move, and the player move will not change.	Pass	N/A
Illegal Move Bishop	-Move a bishop that is currently blocking the king from an attacking piece out of the way	In all cases the bishop will not move, and the player move will not change.	In all cases the bishop will not move, and the player move will not change.	Pass	N/A

Illegal Move King	-Move King into a square where it can be captured - Move king more than one space in any direction, or to any space not one space away in any direction	In all cases except a legal castle the king will not move, and the player turn will not change.	In most cases except a legal castle the king will not move, and the player turn will not change. When an unmoved king is selected and can legally castle then you can try to move it to the g-file or c-file anywhere and the castling move will occur.	fail	Corrected, Added in conditions that checked if a king move was 2 pieces away that the selected destination was one of the correct spaces for castling for that player. Each castle move is now tied to one specific space.
Illegal Move Queen	-Move a queen to a space not directly horizontal, vertical or diagonal to it. -Move a queen over a friendly piece	In all cases the queen will not move and the player turn will not change	In all cases the queen did not move and the player turn did not change	Pass	N/A
Legal Castle	If a rook and king of one color have both not moved, and the path between them is clear. Move the king to the correspondin g legal castle	A castle move will be performed	A castle move was performed	Pass	N/A

	position on the rook's side.				
Check Test	-Threaten a king in such a way that check can be blocked the next turn -Threaten a king in such a way the king can make a move the next turn that negates the check (traversal or aggressive). -Threaten a king in such a way the either of the two scenarios above can occur	In all cases the program should display a dialog box that says the appropriate player is in check.	In all cases the program displays a dialog box saying the appropriate player is in check.	Pass	N/A
Checkmate Test	-Place a each king piece in checkmate	In all cases the program should display a dialog box that says the appropriate player is in checkmate.	In all cases the program displayed a dialog box that says the appropriate player is in checkmate.	Pass	N/A
Move after checkmate	-Try to make a move after checkmate -Undo a move after chekmate	In all cases nothing should happen	Nothing happens in all cases	Pass	N/A

Move to check	-Move the king of either color into a space where it would be in check -Move a friendly piece other than the king in such a way that places the king in check -When a player is in check, move a piece so the player does not escape check	No move will be performed, player move will not change.	No move was performed, and player move did not change.	Pass	N/A
Undo Move button test	-Make any move then undo that move -Press the undo move button a second time after pressing it once	In the first case, the previous move should be undone and the player turn should change. In the second case nothing should happen.	In the first case the previous move is undone. The player turn changes back. In the second case nothing further occurs.	Pass	N/A
Pawn Promotion	Move a pawn from either side to the enemy player's back row through traversal Move a pawn from either	In all cases the pawn should be replaced by a queen piece of the same color	In all cases the pawn was replaced by a queen of the same color	Pass	N/A

	side to the enemy player's back row by taking a piece Perform both of these with pawns from both sides as many times as necessary				