**Built**

Project is built using maven pom.xml

mvn clean compile install

**Run**

Project is run from src/test/java/GameTest

mvn test

**Thought process**

X’s and Os can be represented as a multidimentional array:

|  |  |  |
| --- | --- | --- |
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

Rather than pass in 9 parameters, make each a field in a Game object. Instead of using X and O in the program, use integers as this is easier to handle. Account for empty spaces with another integer. Use statics for these values to avoid Magic Numbers. Game needs to be passed in a Required board of size 9. This Required board can include X/O/Empty.

***Opening Conditions***

Assume we are passed an int[][]. Assume the convertion from X to 1, O to 2, Empty to 0 has been completed in another part of the code. Assume we are passed an int[][] of size 3 x 3.

***States***

Winner with no Empty Spaces/Winner with Empty Spaces/No Winner with no Empty Spaces/No Winner with Empty Spaces - to determine if no one can win when there are still spaces left would mean identifying how many moves left per player and so forth in added complexity. Therefore assume play will continue until there is a winner or until all spaces used – therefore this state will be No Winner Yet.

As the game can have a Winner and be Incomplete (with Empty Spaces) therefore check for Winner first, thereafter if No Winner check if game Complete.

***Ways to win***

8 ways to win (diagrams below – X placed in diagram but can be X or O).

4 ways to win have a marker in the centre, 4 ways have no marker in the centre. Therefore there is no advantage checking the centre first.

As the winning horizontal/vertical/diagonal line must have 3 X or O markers to win, to fail fast check the markers against X or O first (variable XorO used for this), as if the marker is Empty that line will not be a winning line anyway.

***Tests***

I tested the ‘happy path’ - Four states mentioned above tested:

completeGameWithWinner\_shouldReturnWinner

completeGameWithNoWinner\_shouldReturnNoWinner

incompleteGameWithWinner\_shouldReturnWinner

incompleteGameWithNoWinnerYet\_shouldReturnNoWinnerYet

More exception handling and tests could be added to check for non-happy path scenarios like if gameState passed is larger or smaller than 3X3.

***Ways to win - diagrams***

Horizontal:

|  |  |  |
| --- | --- | --- |
| X | X | X |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| X | X | X |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| X | X | X |

Vertical:

|  |  |  |
| --- | --- | --- |
| X |  |  |
| X |  |  |
| X |  |  |

|  |  |  |
| --- | --- | --- |
|  | X |  |
|  | X |  |
|  | X |  |

|  |  |  |
| --- | --- | --- |
|  |  | X |
|  |  | X |
|  |  | X |

Diagonal:

|  |  |  |
| --- | --- | --- |
| X |  |  |
|  | X |  |
|  |  | X |

|  |  |  |
| --- | --- | --- |
|  |  | X |
|  | X |  |
| X |  |  |