ERROR in Spanish Checkers: Queen's multiple square march move is not allowed.

Status: **UNSOLVED** - 20/10/2019

Error in:

```
  Scenario Outline: Valid Regular Move
    Given the game is played up to a certain point from file "<file_name>"
    When the player picks a valid source coordinate that has a "<piece_type>"
            piece in it
    And the player picks a valid destination coordinate that is "one" squares
            away from the source coordinate
    >>>>>>>>>> Then the piece at the source coordinate is moved to the
            destination coordinate
    And the next turn is given to the "other" player

    Examples:
      | file_name          | piece_type |
      | validRegularMove1 | queen      |
```

The game set-up in SpanishCheckers.ini file is specified as follows:

```
[validRegularMove7]
boardSetUp=validRegularMoveBoard7
playerMove=4,4>7,1
turn=black

[validRegularMoveBoard7]
6,4=white-pawn
5,1=white-pawn
4,4=black-queen
1,3=black-pawn
4,6=white-pawn
```

Expected Behavior:

The queen should be able to move to the desired destination coordinate. Queens can travel multiple squares in one move without jumping any piece.

Actual Behavior:

The queen is NOT moved to the desired destination coordinate. The player is asked for another move.

Error:

The move is not evaluated as a valid move. The game thinks that the queen can't travel multiple squares.

Test Outputs:

Before the move is conducted:

```
Testing: validRegularMove7
Player turn: black
Player move: (4,4)->(7,1)
No promote count: 0
No capture count: 0
Best sequence: []
Prior move sequence: []
Game begins ...
    0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   |   | 7
   --- --- --- --- --- --- --- ---
6 |   |   |   |   | W |   |   |   | 6
   --- --- --- --- --- --- --- ---
5 |   |   |   |   |   |   |   |   | 5
   --- --- --- --- --- --- --- ---
4 |   |   |   |   | A |   | W |   | 4
   --- --- --- --- --- --- --- ---
3 |   | B |   |   |   |   |   |   | 3
   --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
   --- --- --- --- --- --- --- ---
1 |   |   |   |   |   | W |   |   | 1
   --- --- --- --- --- --- --- ---
0 |   |   |   |   |   |   |   |   | 0
   --- --- --- --- --- --- --- ---
    0   1   2   3   4   5   6   7
```

Test Outputs:

After the move is conducted:

```
Test: validRegularMove7
Informers: [There must be one piece on jump path 0, Player will be
asked for another source coordinate because the previous move was in-
valid.]
Final Informers: [There must be one piece on jump path 0, Player will
be asked for another source coordinate because the previous move was
invalid.]
Was player going to make another move?: true
End of the game?: false
Board:
    0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   |   | 7
   --- --- --- --- --- --- --- ---
6 |   |   |   |   | W |   |   |   | 6
   --- --- --- --- --- --- --- ---
5 |   |   |   |   |   |   |   |   | 5
   --- --- --- --- --- --- --- ---
4 |   |   |   |   | A |   | W |   | 4
   --- --- --- --- --- --- --- ---
3 |   | B |   |   |   |   |   |   | 3
   --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
   --- --- --- --- --- --- --- ---
1 |   |   |   |   |   | W |   |   | 1
   --- --- --- --- --- --- --- ---
0 |   |   |   |   |   |   |   |   | 0
   --- --- --- --- --- --- --- ---
    0   1   2   3   4   5   6   7
```

HOW DOES CUCUMBER HELP US FIND THE PROBLEM?

The following method in `AmericanCheckersScenarioTester` (superclass of SpanishCheckersScenarioTester) fails (lines are renumbered):

```
1 @Override
2 public void thePieceAtTheSourceCoordinateIsMovedToTheDestinationCoordinate() {
3   //Check if destination coordinate now holds the moved piece.
4   assertEquals(pieceOfPlayerMove,
5     getPieceAtCoordinate(destinationCoordinateOfPlayerMove));
6   //Check if the source coordinate is now empty.
7   assertTrue(getPieceAtCoordinate(sourceCoordinateOfPlayerMove) == null);
8   //Check if the piece's current coordinate is the same as player move's
9   //destination coordinate.
10  assertEquals(destinationCoordinateOfPlayerMove,
11  pieceOfPlayerMove.getCurrentCoordinate());
12 }
```

And we know that the queen is not moved to the destination coordinate. We can debug the code to find which decision leads to that.

WHERE IS THE PROBLEM?

In the `checkMove()` method of `SpanishReferee` class, the `RuleIfJumpMoveThenJumpedPieceMustBeOpponentPiece` rule class is evaluated. In its `evaluate(...)` method, the following check happens:

```
33 if(howManyPieceAreOnPath==0) {
34   if(moveConstraints.isMultipleSquareMarchMoveAllowed(moveDirection))
35     return true;
36 ...
```

It has to return true on the 35th line for this move to be valid, but `QueenMoveConstrains` class returns false from `isMultipleSquareMarchMoveAllowed(moveDirection)` for every possible direction.

Note: The same error also causes validRegularMove8 test to fail.