ERROR in American Checkers: The player is accepted as the winner and the game is ended even though the opponent can still make a valid move.

Status: **UNSOLVED** – 26/09/2019

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
And the player picks a valid destination coordinate that is "one" squares away from the source coord
inate
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          |
| validRegularMove6 |
```

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

```
[validRegularMove6]
boardSetUp=validRegularMoveBoard6
turn=black
playerMove=2,0>3,1

[validRegularMoveBoard6]
2,0=black-pawn
7,7=white-pawn
6,6=black-pawn
```

Test Outputs:

Before the move is conducted:

```
Player turn: black
Player move: (2,0)->(3,1)
B: Pawn of 'black' player
A: King of 'black' player
W: Pawn of 'white' player
Z: King of 'white' player
     0   1   2   3   4   5   6   7
    --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   | W | 7
    --- --- --- --- --- --- --- ---
6 |   |   |   |   |   |   | B |   | 6
    --- --- --- --- --- --- --- ---
5 |   |   |   |   |   |   |   |   | 5
    --- --- --- --- --- --- --- ---
4 |   |   |   |   |   |   |   |   | 4
    --- --- --- --- --- --- --- ---
3 |   |   |   |   |   |   |   |   | 3
    --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
    --- --- --- --- --- --- --- ---
1 |   |   |   |   |   |   |   |   | 1
    --- --- --- --- --- --- --- ---
0 |   |   | B |   |   |   |   |   | 0
    --- --- --- --- --- --- --- ---
     0   1   2   3   4   5   6   7
```

After the move is conducted:

```
Status: Test ended with a valid player move. The game ended.
Informers: [OPPONENT'S MOVE IS BLOCKED!!!!!!, WINNER Player [id=0,
color=java.awt.Color[r=0,g=0,b=0]]]
Was player going to make another move?: false
End of the game? true
isDraw?: false, Winner: Player [id=0, color=java.awt.Color[r=0,g=0,b=0]], Loser: Player
[id=1, color=java.awt.Color[r=255,g=255,b=255]]
Board:
     0   1   2   3   4   5   6   7
    --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   | W | 7
    --- --- --- --- --- --- --- ---
6 |   |   |   |   |   |   | B |   | 6
    --- --- --- --- --- --- --- ---
5 |   |   |   |   |   |   |   |   | 5
    --- --- --- --- --- --- --- ---
4 |   |   |   |   |   |   |   |   | 4
    --- --- --- --- --- --- --- ---
3 |   |   |   |   |   |   |   |   | 3
    --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
    --- --- --- --- --- --- --- ---
1 |   |   |   | B |   |   |   |   | 1
    --- --- --- --- --- --- --- ---
0 |   |   |   |   |   |   |   |   | 0
    --- --- --- --- --- --- --- ---
     0   1   2   3   4   5   6   7
```

Expected behavior:

After the move is conducted, the white player should be given the turn as he has one possible jump move to make: `(7,7)->(5,5)`

Actual behavior:

After the move is conducted, the white player is NOT given the turn. The game is ended in black player's victory.

Error:

The game doesn't recognize the white player's jump move as valid, thus leaving him unable to perform a move, which ends the game with black player's victory.

HOW DOES CUCUMBER HELP US FIND THE PROBLEM?

The following method in AmericanCheckersScenarioTester class fails in the 6th line:

```
 1 @Override
 2 public void theNextTurnIsGivenToTheP1Player(String p1) {
 3     if (referee.gameEnded) {
 4          output("Informers: " + referee.informers.toString());
 5     }
 6     assertFalse(referee.gameEnded);
 7     if (p1.equals("other")) {
 8          assertFalse(referee.playerWasGoingToMakeAnotherMove);
 9          assertFalse(referee.getCurrentPlayer().equals(playerOfPlayerMove));
10     } else if (p1.equals("current")) {
11          assertTrue(referee.playerWasGoingToMakeAnotherMove);
12          assertTrue(referee.getCurrentPlayer().equals(playerOfPlayerMove));
13     }
14 }
```

We know the `gameEnded` variable is set to `true` even though it shouldn't be. The preceding 3 lines output the informers (see in Test Outputs page) to test output file which lets us know that the game thinks opponent's move is blocked. We can then debug the code from there and find where the error occurred.

WHERE IS THE PROBLEM?

In `RuleEndOfGameWhenOpponentBlocked` class' `private boolean isOpponentMoveBlocked (AbstractReferee referee)` method, each piece of the opponent gets checked for possible valid moves. In line 47, `private boolean isJumpedPieceExistsAndBelongsToOpponent(...)` method gets called. That method is like this (omitting the declaration):

```
0 CoordinatePieceMap coordinatePieceMap = referee.getCoordinatePieceMap();
1 List<ICoordinate> path = cbo.findPath(currentPiece, possibleMoveCoordinate);
2 ICoordinate pathCoordinate = path.get(1);
3 AbstractPiece pieceAtPath = coordinatePieceMap.getPieceAtCoordinate(pathCoordinate);
4 if (pieceAtPath==null || pieceAtPath.getPlayer().equals(currentPlayer)) {
5     return false;
6 }
7 return true;
```

The problem is at line 4, while checking `pieceAtPath.getPlayer().equals(currentPlayer)`: In the test run, the currentPlayer is the black player. However, this method gets called from the perspective of the other player.

- In the class name, “opponent” means the white player.
- In line 47 where this method is called from the white player's perspective, the intention is: “opponent” means the black player.
- If this method is considered alone, “opponent” means the white player, and the currentPlayer is the black player.

Thus, when the game is trying to find a valid jump move for white player, it thinks that if the jumped piece is a black piece, move can't happen.

The jump move that was supposed to be evaluated as valid passes in validJumpMove10 test.

Note: The same error also causes validRegularMove4 and validRegularMove5 tests to fail.