ERROR in American Checkers: the game does not give next player the turn after current player has no more valid jump moves to make during a jump move series.

Status: **UNSOLVED** - 22/09/2019

Error in:

```
Scenario Outline: Valid Jump 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 "two" squares away from the source coordinate
Then the piece at the source coordinate is moved to the destination coordinate
And the opponent piece in between the source and destination coordinates are removed from the board
 >>>>>>>>>> And the next turn is given to the "<next_turn_player>" player

Examples:
| file_name      | next_turn_player |
| validJumpMove1 | other            |
```

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

```
[validJumpMove1]
boardSetUp=validJumpMoveBoard1
playerMove=5,3>7,5
turn=black

[validJumpMoveBoard1]
5,3=black-pawn
6,6=black-king
6,4=white-pawn
5,5=white-king
```

Test outputs:

Before the move is conducted:

```
Player turn: black
Player move: (5,3)->(7,5)
B: Pawn of the 'black' player.
A: King of the 'black' player.
W: Pawn of the 'white' player.
Z: King of the 'white' player.

    0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   |   | 7
   --- --- --- --- --- --- --- ---
6 |   |   |   |   |   |   | A |   | 6
   --- --- --- --- --- --- --- ---
5 |   |   |   |   |   | Z |   |   | 5
   --- --- --- --- --- --- --- ---
4 |   |   |   |   |   |   | W |   | 4
   --- --- --- --- --- --- --- ---
3 |   |   |   |   |   | B |   |   | 3
   --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
   --- --- --- --- --- --- --- ---
1 |   |   |   |   |   |   |   |   | 1
   --- --- --- --- --- --- --- ---
0 |   |   |   |   |   |   |   |   | 0
   --- --- --- --- --- --- --- ---
    0   1   2   3   4   5   6   7
```

After the move is conducted:

```
Test: validJumpMove1
Status: Test ended with a valid player move.
Informers: [Player will be asked for another destination coordinate (previous move was a jump
move) (there are still possibilities for a jump move).]

Board:
    0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
7 |   |   |   |   |   |   |   |   | 7
   --- --- --- --- --- --- --- ---
6 |   |   |   |   |   |   | A |   | 6
   --- --- --- --- --- --- --- ---
5 |   |   |   |   |   | Z |   | B | 5
   --- --- --- --- --- --- --- ---
4 |   |   |   |   |   |   |   |   | 4
   --- --- --- --- --- --- --- ---
3 |   |   |   |   |   |   |   |   | 3
   --- --- --- --- --- --- --- ---
2 |   |   |   |   |   |   |   |   | 2
   --- --- --- --- --- --- --- ---
1 |   |   |   |   |   |   |   |   | 1
   --- --- --- --- --- --- --- ---
0 |   |   |   |   |   |   |   |   | 0
   --- --- --- --- --- --- --- ---
    0   1   2   3   4   5   6   7
```

Expected behavior:

After the jump move (5,3 → 7,5) is conducted, the informer should say:

```
“Player will NOT be asked for another destination coordinate (previous move was a
jump move) because there are no more possibilities for a jump move.”
```

And the next turn should be given to the other player.

Actual behavior:

After the jump move (5,3 → 7,5) is conducted, the informer says:

```
"Player will be asked for another destination coordinate (previous move was a jump
move) (there are still possibilities for a jump move)."
```

And the next turn is NOT given to the other player.

Error:

The game decides that the “A” piece can be jumped over by “B” piece. However, they belong to the same player, and thus another jump move is not possible at that point.

HOW DOES CUCUMBER HELP US FIND THE PROBLEM?

The following method in AmericanCheckersScenarioTester class fails in the 4th line:

```
1  @Override
2  public void theNextTurnIsGivenToTheP1Player(String p1) {
3        if (p1.equals("other")) {
4              assertFalse(playerWasGoingToMakeAnotherMove);
5              assertFalse(referee.getCurrentPlayer().equals(player));
6        } else if (p1.equals("current")) {
7              assertTrue(playerWasGoingToMakeAnotherMove);
8              assertTrue(referee.getCurrentPlayer().equals(player));
9        }
10 }
```

And we know that the game thinks the player was going to make another move after the `playerMove` specified in ini file is conducted.

TesterReferee class gives us a more explicit feedback (see in Test Outputs page) about why the game thinks that player was going to make another move. Namely: the game thinks there are still possible jump moves.

We can debug the program from a breakpoint where the `playerWasGoingToMakeAnotherMove` variable was set to `true`, and find the code that causes this test failure.

WHERE IS THE PROBLEM?

In CoordinateBasedOperations class, `public List<ICoordinate> findAllowedContinousJumpList(AbstractPiece piece)` method, lines #200-203:

```
if(numberOfPiecesOnPath==1) {
      ICoordinate destination = path.get(path.size()-1);
      secondJumpList.add(destination);
}
```

This method is used in Referee class' `protected boolean conductCurrentPlayerTurnAgain(MoveOpResult moveOpResult, AbstractPiece piece)` method to find the list of possible jump move destination coordinates after a jump move is already conducted. However it adds the coordinate as a valid jump move destination without checking that the piece in path (the piece that will be jumped) is not the player's own piece. The move will not get conducted if chosen anyway, but the game can't proceed from there because it expects a jump move to happen.