CS 3331 - Advanced Object-Oriented Programming

Group Work: Test-Driven Development with JUnit and Eclipse

Develop class Board by applying the "test-first principle": Start
software development by writing tests. The class Board is to keep
track of the state of a Connect Five board.

Step 1. Write the following JUnit test class. While typing the code,
  you will encounter several errors. Ignore them for now, as you will
  fix them later using Eclipse tools.

```
  public class BoardTest {

    private Board board;

    @Before
    public void setUp() {
        board = new Board();
    }

    @Test
    public void testBoard() {
        assertEquals(9, board.size()); // default size
    }
  }
```

  1.1 Select @Test and press Ctrl-1 (Quick-Fix). Select "Add JUnit 4
    library to the build path." Eclipse will add the JUnit jar files
    to the build path and also import org.junit.Test.

1.2 Press Ctrl-1 (Quick-Fix) on @Before and select "import 'Before' (org.junit)".

1.3 Let the Eclipse create the missing class Board including its methods.
(a) Press Ctrl-1 on "Board" from the board field declaration, and select "create class Board".
(b) Similarly create the size() methods; press Ctrl-1 on method names.

1.4 Press Ctrl-1 on "assertEquals" and select "add static import org.junit.Assert.*"

1.5 Run the test by selecting "Run As" > "JUnit Test" from the popup menu, and make sure the test passes.

1.6 Introduce a new constructor that takes a custom board size as a parameter and test the constructor, e.g.,

```
@Test
public void testBoard2() {
    board = new Board(4);
    assertEquals(4, board.size());
}
```

You may need to refactor (rewrite) the code of the default constructor.

Step 2. Incrementally write other methods of the Board class by repeating the following steps for each (or a few closely-related)

```
  method. You may need to introduce other classes such as
Disc; but
  don't worry about coding its implementation or testing
it.

  Write tests.
  Generate the stub (skeletal code) using Eclipse Quick-
Fix.
  Code the generated skeletal methods.
  Make sure all the tests pass.

  Q: What other methods do you need to define for the Board
class?
  A: Think from a client's perspective; i.e., what service
does
     a client (a view/control class) require from the Board
class?

     Know the board
     Know all the discs, rows, columns
     Find the disc at a specified indexes
     Find all discs for player 1/player 2
     Determine if a player has won
     Clear the values of all discs
     ...

     Operations for the Disc class?
```