```
package com.darringer.games.ttt.control;

import static com.darringer.games.ttt.model.TTTGameStatus.IN_PROGRESS;
import static com.darringer.games.ttt.model.TTTGameStatus.NOT_POSSIBLE;
import static com.darringer.games.ttt.model.TTTGameStatus.O_WIN;
import static com.darringer.games.ttt.model.TTTGameStatus.TIE;
import static com.darringer.games.ttt.model.TTTGameStatus.UNKNOWN;
import static com.darringer.games.ttt.model.TTTGameStatus.X_WIN;
import static com.darringer.games.ttt.model.TTTModel.O_SQUARE;
import static com.darringer.games.ttt.model.TTTModel.X_SQUARE;
import static org.junit.Assert.assertTrue;

import org.junit.Test;

import com.darringer.games.ttt.model.TTTModel;

/**
 *
 * @author cdarringer
 *
 */
public class TTTGameControllerTest {

	private TTTGameController controller = new TTTGameController();

	@Test
	public void testNewGame() {
		TTTModel model;
```

```
        model = new TTTModel();
        assertTrue("New games are in progress", IN_PROGRESS == model.getStatus());

        model = new TTTModel("xxxxxxxxx");
        assertTrue("Loaded games are unknown", UNKNOWN == model.getStatus());

        model = new TTTModel("xabxcdxef");
        assertTrue("Invalid values are not possible", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel("xxx");
        assertTrue("Invalid games are not possible", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel("xxxxxxxxxxx");
        assertTrue("Invalid games are not possible", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel("");
        assertTrue("Invalid games are not possible", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel(null);
        assertTrue("Invalid games are not possible", NOT_POSSIBLE == model.getStatus());


    }

    @Test
    public void testInvalidMoves() {
        TTTModel model;

        model = new TTTModel();
```

```
        model = controller.makeMove(model, -1, X_SQUARE);
        assertTrue("Index should be valid", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel();
        model = controller.makeMove(model, 10, X_SQUARE);
        assertTrue("Index should be valid", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel();
        model = controller.makeMove(model, 5, 'z');
        assertTrue("Index should be valid", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel("--------x");
        model = controller.makeMove(model, 8, X_SQUARE);
        assertTrue("Square should not already be occupied", NOT_POSSIBLE ==
model.getStatus());

        model = new TTTModel("x--------");
        model = controller.makeMove(model, 0, X_SQUARE);
        assertTrue("Square should not already be occupied", NOT_POSSIBLE ==
model.getStatus());
        model = new TTTModel();
        model = controller.makeMove(model, -1, 'Z');
        assertTrue("Square value should be valid", NOT_POSSIBLE == model.getStatus());

        model = new TTTModel("xxx------");
        model = controller.makeMove(model, 5, X_SQUARE);
        assertTrue("Square value should be valid", X_WIN == model.getStatus());

    }
```

```
@Test
public void testWinningMoves() {
        TTTModel model;

        model = new TTTModel("-xx------");
        model = controller.makeMove(model, 0, X_SQUARE);
        assertTrue("X should win horizontal", X_WIN == model.getStatus());

        model = new TTTModel("---x--x--");
        model = controller.makeMove(model, 0, X_SQUARE);
        assertTrue("X should win vertical", X_WIN == model.getStatus());

        model = new TTTModel("----x---x");
        model = controller.makeMove(model, 0, X_SQUARE);
        assertTrue("X should win diagonal", X_WIN == model.getStatus());

        model = new TTTModel("---xx----");
        model = controller.makeMove(model, 5, X_SQUARE);
        assertTrue("X should win diagonal", X_WIN == model.getStatus());

        model = new TTTModel("-------xx");
        model = controller.makeMove(model, 6, X_SQUARE);
        assertTrue("X should win diagonal", X_WIN == model.getStatus());

        model = new TTTModel("-------xx");
        model = controller.makeMove(model, 6, X_SQUARE);
        assertTrue("X should win diagonal", X_WIN == model.getStatus());
```

```
                model = new TTTModel("----x--x-");
                model = controller.makeMove(model, 1, X_SQUARE);
                assertTrue("X should win diagonal", X_WIN == model.getStatus());

                model = new TTTModel("-----x--x");
                model = controller.makeMove(model, 2, X_SQUARE);
                assertTrue("X should win diagonal", X_WIN == model.getStatus());

                model = new TTTModel("----x-x-x");
                model = controller.makeMove(model, 2, X_SQUARE);
                assertTrue("X should win diagonal", X_WIN == model.getStatus());

                model = new TTTModel("----x---x");
                model = controller.makeMove(model, 0, X_SQUARE);
                assertTrue("X should win diagonal", X_WIN == model.getStatus());

                model = new TTTModel("----x-x--");
                model = controller.makeMove(model, 2, X_SQUARE);
                assertTrue("X should win diagonal", X_WIN == model.getStatus());

                model = new TTTModel("-oo------");
                model = controller.makeMove(model, 0, O_SQUARE);
                assertTrue("0 should win ", O_WIN == model.getStatus());

                model = new TTTModel("----oo---");
                model = controller.makeMove(model, 3, O_SQUARE);
                assertTrue("0 should win ", O_WIN == model.getStatus());

                model = new TTTModel("-------oo");
```

```
                model = controller.makeMove(model, 6, O_SQUARE);
                assertTrue("0 should win ", O_WIN == model.getStatus());

                model = new TTTModel("---o--o--");
                model = controller.makeMove(model, 0, O_SQUARE);
                assertTrue("0 should win ", O_WIN == model.getStatus());

        }

        @Test
        public void testLosingMoves() {
                TTTModel model;

                model = new TTTModel("-xoxx-oo-");
                model = controller.makeMove(model, 0, X_SQUARE);
                assertTrue("O should win on next move", O_WIN == model.getStatus());
                assertTrue("O took the win horizontally", O_SQUARE == model.getSquare(8));

                model = new TTTModel("-xo-xo-o-");
                model = controller.makeMove(model, 0, X_SQUARE);
                assertTrue("O should win on next move", O_WIN == model.getStatus());
                assertTrue("O took the win vertcally", O_SQUARE == model.getSquare(8));

                model = new TTTModel("oxo-o---x");
                model = controller.makeMove(model, 3, X_SQUARE);
                assertTrue("O should win on next move", O_WIN == model.getStatus());
                assertTrue("O took the win diagonally", O_SQUARE == model.getSquare(6));
        }
```

```
	@Test
	public void testTieMoves() {
		TTTModel model;

		model = new TTTModel("oxoxox--x");
		model = controller.makeMove(model, 6, X_SQUARE);
		assertTrue("O should tie the game", TIE == model.getStatus());
		assertTrue("O takes the last square", O_SQUARE == model.getSquare(7));
	}

	@Test
	public void testInProgressMoves() {
		TTTModel model;

		model = new TTTModel("x--------");
		model = controller.makeMove(model, 2, O_SQUARE);
		assertTrue("In progress game", IN_PROGRESS == model.getStatus());

		model = new TTTModel("o--------");
		model = controller.makeMove(model, 2, O_SQUARE);
		assertTrue("In progress game", IN_PROGRESS == model.getStatus());

	}


}
```