CIT/CSE337_Summer_II_2019 “Suggestions & Hints for the Class Project”_v3.0_190710

Overview

I am providing the following Use Case, UML Model and Java class code fragments as suggestions to help your Team get started with the Class Project. None of the following Java class code fragments are NOT necessarily the only way to implement the “TicTacToeGame”; it is simply one way. I have posted the “TicTacToeGame” Astah UML Model on Moodle separately. I have copied the AL_00 White Box sequence diagram (AL_00_WB_SD) from that UML Model to this document as shown in Figure 1. below. The Java code fragments below are consistent with the “Operational Concept” document for the Class Project already posted on Moodle.

NOTE: there may be some inconsistencies in the Java code fragments – so BEWARE!

Here is the user story:

Preconditions:

(1) The game has been launched & initialized. Player1 will be ‘X’ and Player2 will be ‘O’.

(2) Assume Player1 using ‘X’ moves first.

(3) All Board positions have been cleared.

User Story

1. the Use Case begins when Player1 selects a Board position and enters an ‘X’.

2. The System responds by storing Player1's move on the Board.

3. The System further responds by printing the current Board position to the command line.

4. The System further responds by checking if there is a win condition or a draw condition. If neither occurs, the System changes players to Player2.

5. Player2 selects a Board position and enters an ‘O’.

6. The System responds by entering Player2's move on the Board.

8. If there is no win or draw condition, the System further responds changing players back to Player1.

9. The System further responds by alternating players, recording each player's move if there is no win or draw condition.

10. The Use Case ends when the System determines there is a win or draw condition.

JUnit4.12 TDD Java Test Class Fragment, “TicTacToeTest.java”

You should use the JUnit4.12 test class to build the TicTacToeGame App production and helper classes. I have provided the Java code fragment in Figure 2. To get the Teams started on “filling in” the Manager, Display, Middleware and Database classes to satisfy the JUnit4.12 TDD tests.

Note I have not provided a Feature File or Step Definition File. You should produce these as part of your Java App.

The AL_00_WB_SD appears in Figure 1. below along with the JUnit4.12 TDD test class fragment in Figure 2.

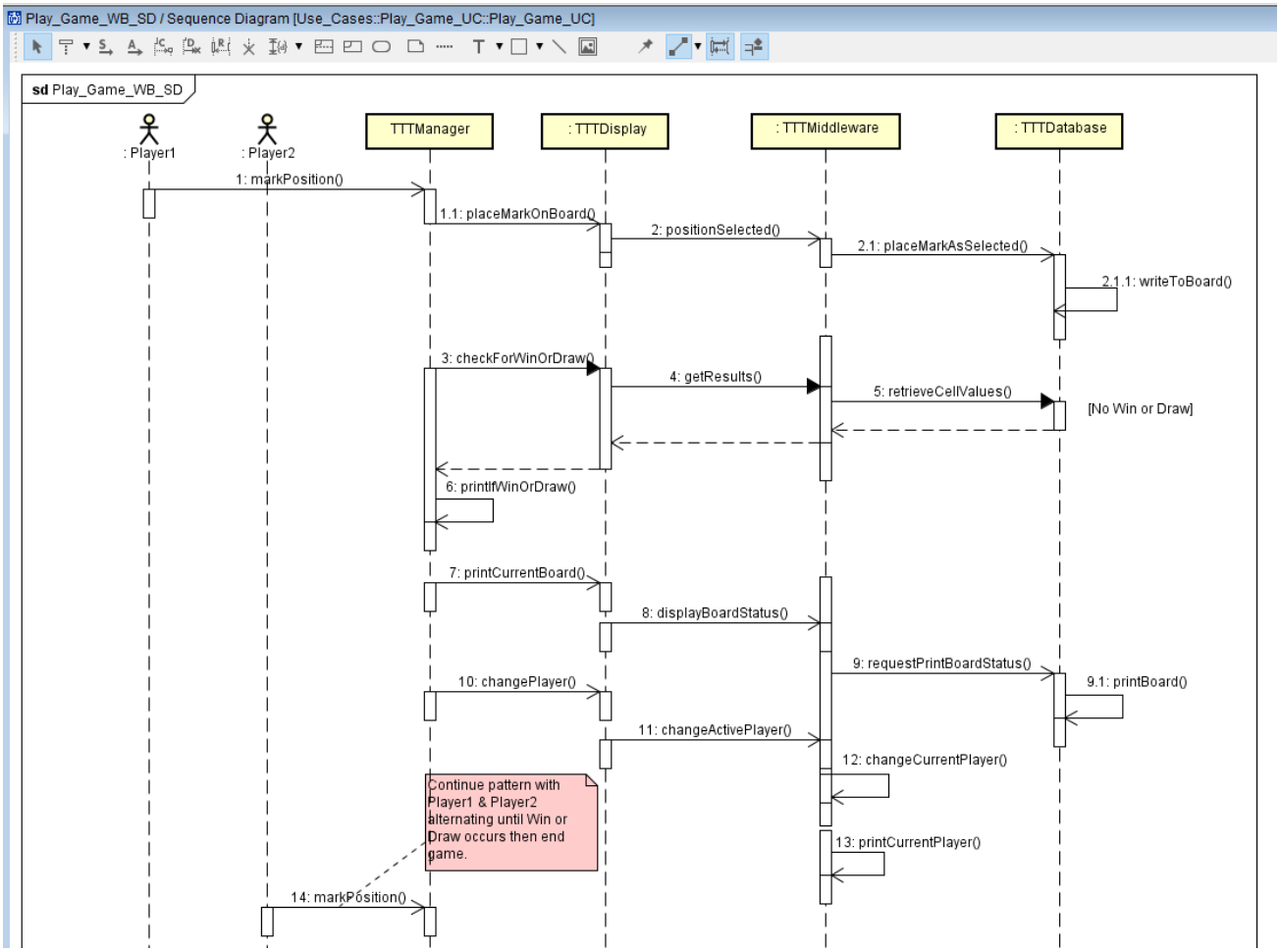

Figure 1. the AL_00_WB_SD for the use case, “Play_Game_UC”;

TicTacToeTest.java (D:\Academic_Year_2017\cit337\source\edu\oakland\projectTest\)

```java
package edu.oakland.projectTest;

import edu.oakland.production.*;
import edu.oakland.helper.*;
import org.junit.*;
import org.junit.Assert.*;

public class TicTacToeTest {
    // JUnit4.12 is conducting a TDD test;
    // instance variables go here;

    @Before
    // 1.  launch and initialize all classes of the App;
    // 2.  ensure Board is reset;

    @Test
    // 1.  fill the elements of the Board along a diagonal;
    // 2.  can the App correctly determine a win?

    @Test
    // 1.  fill the elements of the Board within a column;
    // 2.  can the App correctly determine a win?

    @Test
    // 1.  fill the elements of the Board within a row;
    // 2.  can the App correctly determine a win?

    @Test
    // 1.  fill all elements of the Board such that there is no win;
    // 2.  can the App correctly determine a draw?

}
```

Figure 2. the JUnit4.12 TDD test class Java code fragment “TicTacToeTest.java”;

The following Java class code fragments are based on the AL_00_WB_SD in Figure 1.

TicTacToeManager.java (D:\Academic_Year_2017\cit337\source\edu\oakland\manager\)

```java
package edu.oakland.helper;

import edu.oakland.production.*;
import java.util.*;

public class TicTacToeManager {
    // reference variables for subsystems;
    private TTTDisplay display;
    private TTTMiddleware middleware;
    private TTTDatabase database;
    private Scanner scanner;
    private String currentPlayer;
    private char currentPlayerMark;
    private boolean isWinOrDrawResult;

    public static void main(String[] args) {

        // create instance of TTTManager;
        TicTacToeManager manager = new TicTacToeManager();

        // launch and initialize all TTTgame classes;
        manager.launchApp();

        manager.chooseSquare();
        }
        // provide method to launch & intialize; must do TTTDatabase first;
        public void launchApp() {
            // launch & initialize database; make sure all cells are empty;

            // launch & intialize middleware; pass in database ref;

            // launch & initialize display; pass in middleware ref;
        }
        // provide method to select Board square;
```

```java
        public void chooseSquare() {
            /* use the Scanner class to request first player's input
            from the command line, including "Player1" or "Player2",
            & currentPlayerMark either 'X' or 'O'; */
            printCurrentBoard();

            checkForWinOrDraw();
        }
        // provide method to check for win or draw;  if none change players;
        public void checkForWinOrDraw() {
            // get results of the win or draw check;
            isWinOrDraw = display.checkForWinOrDraw();
            if(!isWinOrDrawResult) {
                changePlayers();
            }
            else {
                /* provide Java code to print either a win or draw to
                the command line and game over - restart; */
            }
        }
        // provide method to print Board status;
        public void printCurrentBoard() {
        // add Java code here;
        }
        // provide method to change players if no win or draw;
        public void changePlayers() {
            display.changePlayer(currentPlayerMark);
            // now loop back to the chooseSquare() method;
            chooseSquare();
        }
}
```

Figure 3. the Java code fragment “TicTacToeManager.java”;

TTTDisplay.java (D:\Academic_Year_2017\cit337\source\edu\oakland\production\)

```java
package edu.oakland.production;

public class TTTDisplay {
    private TTTMiddleware middleware;
    private int rowNumber;
    private int columnNumber;
    private char playerMark;
    private char[][]boardStatus;
    private boolean isWinOrDraw;

    // constructor to receive TTTMiddleware ref;
    TTTDisplay(TTTMiddleware ref) {
        middleware = ref;
    }

    // provide method to place mark in cell;
    public void placeMarkOnBoard(int row, int column, char mark) {
        rowNumber = row;
        columnNumber = column;
        playerMark = mark;
        middleware.positionSelected(rowNumber, columnNumber, playerMark);

    }
    //provide method to get results of a check for win or draw;
    public boolean checkforWinOrDraw() {
        isWinOrDraw = middleware.getResults();
        return isWinOrDraw;
    }
    // provide method to print Board status;
    public void displayBoardStatus {

        middleware.requestPrintBoardStatus();
    }

    // provide a method to change active player;
    public void changePlayer(char mark) {
        playerMark = mark;
        middleware.changeActivePlayer(playerMark);
    }
}

```

Figure 4. the Java code fragment “TTTDisplay.java”;

TTTMiddleware.java (D:\Academic_Year_2017\cit337\source\edu\oakland\production\)

```java
package edu.oakland.production;

public class TTTMiddleware {

    private TTTDatabase dataBase;
    private int rowNumber;
    private int colNumber;
    private  markChar;
    private boolean isBoardFull;
    private boolean isThereAWinOrDraw;

    public TTTMiddleware(TTTDatabase ref) {
        dataBase = ref;
    }
    // method to select position;
    public void positionSelected(int rowValue, int colValue, char markValue) {
        rowNumber = rowValue;
        colNumber = colValue;
        markChar = markValue;
        ref.placeMarkAsSelected(rowNumber, colNumber, markChar);
    }
    /* provide method to check for win or draw; this involves
    (1) retrieving sets of all rows, columns and diagonal cells;
    (2) check each set for all cells containing the same player "mark";
    (3) if (1) & (2) above are false, check to see if Board is full
    which means a tie; all these conditions lead to "game over"; */
    public boolean checkForWinOrDraw() {
    // provide Java code to check for win or draw for all possibilities;
        isThereAWinOrDraw = (result of analysis of all rows, columns and diags);
        return isThereAWinOrDraw;
    // provide Java code to print out win or draw; if neither, print nothing;

    // if win or draw, provide Java code to end game;
    }

    // provide a method to print the Board status;
    public void requestPrintBoardStatus() {

    }
    /*provide a method to change players; pass the mark of the
    current player; print the request for the next player with
    the opposite mark; */
    public void changeActivePlayer(char mark) {
        /* provide Java code that will change the value of markChar based on
        move just made; */
        markChar = mark;
        // provide Java code to print out who the next player will be;
        System.out.println("The next player will play " + markChar);
    }
}

```

Figure 5. the Java code fragment, “TTTMiddleware.java”;

TTTDatabase.java (D:\Academic_Year_2017\cit337\source\edu\oakland\production\)

```java
package edu.oakland.production;


public class Database {

    private GameBoard gameboard;
    private int numberOfRow;
    private int numberOfColumn;
    private char typeOfMark;

    /* positions on the GameBoard are indicated by "cell[i][j]" where
    "i" represents the row and "j" represents the column; rows and
    columns are indexed from [0] to [2]; therefore the center cell
    would be cell[1][1]; */

    // provide method to enter character in a cell;
    public void writeToBoard(int row, int col, char mark) {
    // use the helper class "GameBoard.java" for all Board instructions;
    }
    // provide method to print board after each player move;
    public void printBoardStatus() {

    }
    /* provide a method to return the value of cell[i][j] in order to
    check for a win or a draw; */
    public char getCellValues() {

    }
}

```

Figure 6. the Java class code fragment, “TTTDatabase.java”;

GameBoard.java (D:\Academic_Year_2017\cit337\source\edu\oakland\helper\)

```java
package edu.oakland.helper;

public class GameBoard {

    private char[][] board;
    private char currentPlayerMark;

    public static void main(String[] args) {
        GameBoard gameBoard = new GameBoard();

        gameBoard.initializeBoard();

        gameBoard.printBoard();
    }
        public void initializeBoard() {
            board = new char[3][3];
            // clear all values;
            // loop through rows;
            for(int i = 0; i < 3; i++) {
                // loop through columns;
                for(int j = 0; j < 3; j++) {
                        board[i][j] = '-';
                }
            }
        }
        public void printBoard() {
            System.out.println("--------------");
            for(int i = 0; i < 3; i++) {
                for(int j = 0; j < 3; j++) {
                    System.out.print(board[i][j] + " | ");
                }
                System.out.println();
                System.out.println("--------------");
            }
        }
}
```

Figure 7. the “helper” Java code fragment, “GameBoard.java”;

Figure 8. the results of compiling and executing “GameBoard.java”;