MIS 6030 – Application Development

Case Study I – Non OO Console Applications

A 2-Player Console Non-OO Tic-Tac-Toe

1. Run the following application
2. Draw a flow chart for the application
3. With examples from the program, describe the rational of selecting the following features:

* *Arrays vs. primitive Variables*
* *Constants vs. Variables*
* *Do loop vs. the for loop*
* *If statement vs. the switch*

import java.util.Scanner;

/\*\*

\* Tic-Tac-Toe: Two-player console, non-graphics, non-OO version.

\* All variables/methods are declared as static (belong to the class)

\* in the non-OO version.

\*/

public class ClassWork {

// Name-constants to represent the seeds and cell contents

public static final int EMPTY = 0;

public static final int CROSS = 1;

public static final int NOUGHT = 2;

// Name-constants to represent the various states of the game

public static final int PLAYING = 0;

public static final int DRAW = 1;

public static final int CROSS\_WON = 2;

public static final int NOUGHT\_WON = 3;

// The game board and the game status

public static final int ROWS = 3, COLS = 3; // number of rows and columns

public static int[][] board = new int[ROWS][COLS]; // game board in 2D array

// containing (EMPTY, CROSS, NOUGHT)

public static int currentState; // the current state of the game

// (PLAYING, DRAW, CROSS\_WON, NOUGHT\_WON)

public static int currentPlayer; // the current player (CROSS or NOUGHT)

public static int currntRow, currentCol; // current seed's row and column

public static Scanner in = new Scanner(System.in); // the input Scanner

/\*\* The entry main method (the program starts here) \*/

public static void main(String[] args) {

// Initialize the game-board and current status

initGame();

// Play the game once

do {

playerMove(currentPlayer); // update currentRow and currentCol

updateGame(currentPlayer, currntRow, currentCol); // update currentState

printBoard();

// Print message if game-over

if (currentState == CROSS\_WON) {

System.out.println("'X' won! Bye!");

} else if (currentState == NOUGHT\_WON) {

System.out.println("'O' won! Bye!");

} else if (currentState == DRAW) {

System.out.println("It's a Draw! Bye!");

}

// Switch player

currentPlayer = (currentPlayer == CROSS) ? NOUGHT : CROSS;

} while (currentState == PLAYING); // repeat if not game-over

}

/\*\* Initialize the game-board contents and the current states \*/

public static void initGame() {

for (int row = 0; row < ROWS; ++row) {

for (int col = 0; col < COLS; ++col) {

board[row][col] = EMPTY; // all cells empty

}

}

currentState = PLAYING; // ready to play

currentPlayer = CROSS; // cross plays first

}

/\*\* Player with the "theSeed" makes one move, with input validation.

Update global variables "currentRow" and "currentCol". \*/

public static void playerMove(int theSeed) {

boolean validInput = false; // for input validation

do {

if (theSeed == CROSS) {

System.out.print("Player 'X', enter your move (row[1-3] column[1-3]): ");

} else {

System.out.print("Player 'O', enter your move (row[1-3] column[1-3]): ");

}

int row = in.nextInt() - 1; // array index starts at 0 instead of 1

int col = in.nextInt() - 1;

if (row >= 0 && row < ROWS && col >= 0 && col < COLS && board[row][col] == EMPTY) {

currntRow = row;

currentCol = col;

board[currntRow][currentCol] = theSeed; // update game-board content

validInput = true; // input okay, exit loop

} else {

System.out.println("This move at (" + (row + 1) + "," + (col + 1)

+ ") is not valid. Try again...");

}

} while (!validInput); // repeat until input is valid

}

/\*\* Update the "currentState" after the player with "theSeed" has placed on

(currentRow, currentCol). \*/

public static void updateGame(int theSeed, int currentRow, int currentCol) {

if (hasWon(theSeed, currentRow, currentCol)) { // check if winning move

currentState = (theSeed == CROSS) ? CROSS\_WON : NOUGHT\_WON;

} else if (isDraw()) { // check for draw

currentState = DRAW;

}

// Otherwise, no change to currentState (still PLAYING).

}

/\*\* Return true if it is a draw (no more empty cell) \*/

// TODO: Shall declare draw if no player can "possibly" win

public static boolean isDraw() {

for (int row = 0; row < ROWS; ++row) {

for (int col = 0; col < COLS; ++col) {

if (board[row][col] == EMPTY) {

return false; // an empty cell found, not draw, exit

}

}

}

return true; // no empty cell, it's a draw

}

/\*\* Return true if the player with "theSeed" has won after placing at

(currentRow, currentCol) \*/

public static boolean hasWon(int theSeed, int currentRow, int currentCol) {

return (board[currentRow][0] == theSeed // 3-in-the-row

&& board[currentRow][1] == theSeed

&& board[currentRow][2] == theSeed

|| board[0][currentCol] == theSeed // 3-in-the-column

&& board[1][currentCol] == theSeed

&& board[2][currentCol] == theSeed

|| currentRow == currentCol // 3-in-the-diagonal

&& board[0][0] == theSeed

&& board[1][1] == theSeed

&& board[2][2] == theSeed

|| currentRow + currentCol == 2 // 3-in-the-opposite-diagonal

&& board[0][2] == theSeed

&& board[1][1] == theSeed

&& board[2][0] == theSeed);

}

/\*\* Print the game board \*/

public static void printBoard() {

for (int row = 0; row < ROWS; ++row) {

for (int col = 0; col < COLS; ++col) {

printCell(board[row][col]); // print each of the cells

if (col != COLS - 1) {

System.out.print("|"); // print vertical partition

}

}

System.out.println();

if (row != ROWS - 1) {

System.out.println("-----------"); // print horizontal partition

}

}

System.out.println();

}

/\*\* Print a cell with the specified "content" \*/

public static void printCell(int content) {

switch (content) {

case EMPTY: System.out.print(" "); break;

case NOUGHT: System.out.print(" O "); break;

case CROSS: System.out.print(" X "); break;

}

}

}