CSCI3180 – Principles of Programming Languages – Spring 2013

Course Project Phase 1 Exercise — Tic-Tac-Toe

Phase 1 Deadline: Mar 10, 2013 (Sunday) 23:59

1 Introduction

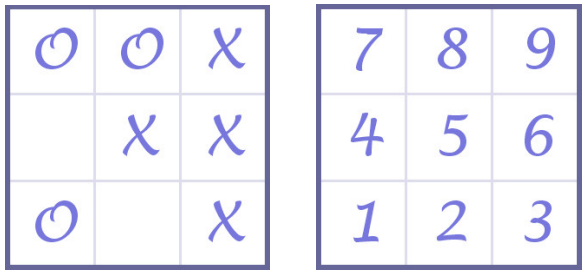
This is a simple programming exercise for Course Project Phase 1 in order to help you get familiar with Smalltalk and the corresponding development environment. In this exercise, you have to implement the Tic-Tac-Toe game using Smalltalk.

2 Exercise Details

In this programming exercise, you have to strictly follow the game rules for Tic-Tac-Toe stated in this section.

2.1 Tic-Tac-Toe Description

The game board consists of 3 rows and 3 columns and it is empty initially. Players can mark their symbols on the cells. There are two players in the game, Player X and Player O, which take turns to mark their symbol X and O respectively on an empty cell. Player X always plays first in the game. The first player who gets three symbols in a row (horizontally, vertically or diagonally) wins the game. If no one wins the game until all cells are filled, the game ends in a tie. Figure 1a shows a game won by Player X.



(a) Player X wins the game (b) Position index of each cell

Figure 1: Tic-Tac-Toe Game Board

2.2 Types of Players

The two players, Player X and Player O, can be controlled by either human or computer. Users can choose the types (human or computer) of Player X and Player O before the game starts.

Human-Controlled Player

A human-controlled player needs human user to provide instruction for each move. In each turn, the symbol of the human-controlled player will be put in the position input by the human. The input provided by the human is an integer from 1 to 9, indicating the position of the symbol that should be marked in the 3x3 grid (as shown in Figure 1b).

Computer-Controlled Player

A computer-controlled player makes each move based on the following strategies making decisions in exactly the specified order.

1. **Winning**
   If it is possible to mark a cell i and win the game, mark i.

2. **Blocking**
   If the opponent will win the game by marking a cell i in next turn, mark i.

3. **Mark Center**
   Mark the centering cell (position 5 in Figure 1b) if it is unoccupied.

4. **Mark Corner**
   Randomly mark one of the unoccupied corner cells (positions 1, 3, 7, 9 in Figure 1b).

5. **Mark Others**
   Randomly mark one of the unoccupied cells.

The strategies 1 to 5 should be checked/performed sequentially. For example, if strategy 1 does not work, perform strategy 2 and so on so forth.

3 Input/Output Specification

You have to strictly follow the input/output specification in this section.

3.1 Input Specification

In this exercise, you are required to use dialog box to get input information. Since this is a simple program, there are just two operations requiring user-input.

Types of Players

Users can choose the types of Player X and Player O to be either human- or computer-controlled. You have to use two dialog boxes to request for the types of the two players before starting the game (as shown in Figure 2).

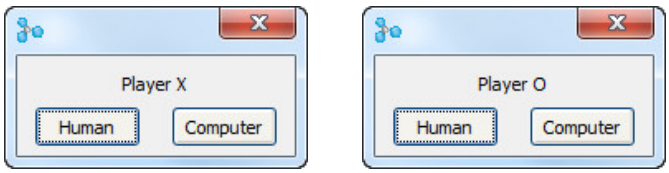


Figure 2: Dialog boxes for players' types

Human-controlled Player's Move

Human-controlled players need user-inputs to determine their next moves. When it is human-controlled player's turn during the game, your program should pop up a dialog box (see Figure 4) to request for the position of the next move. The input should be an integer from 1 to 9. Any invalid input or moves should be forbidden and the same dialog should be pop up again until a valid move is made.

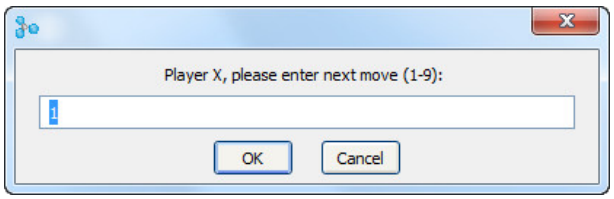


Figure 3: Dialog box for human-controlled player's move

3.2 Output Specification

All the output should be printed to the Transcript window. Whenever there are changes made in the game board, your program should clear the Transcript and print out the updated game board. When the game is finished, a message that indicates the result of the game should be printed.

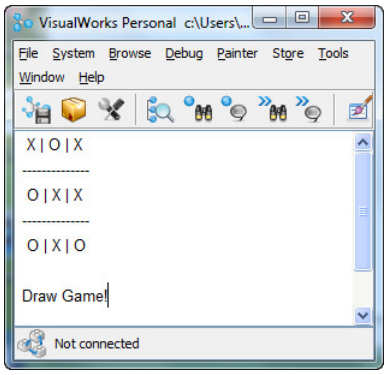


Figure 4: Output shown in the Transcript window

4 Smalltalk Classes

Please follow the classes `TicTacToe`, `Human` and `Computer` defined below in the Smalltalk implementation. You are free to add other variables, methods or classes. You should run your program using the statement: `TicTacToe new startGame.`

1. **Class** `TicTacToe`
   A game object which holds the game board, coordinate players' moves and control game logics. You have to implement it with the following components:

   - **Instance Variable(s)**

     `private gameBoard`
     - This is an array representing the game board.

     `private player1`
     - This is a variable representing Player X.

     `private player2`
     - This is a variable representing Player O.

     `private turn`
     - This is a variable indicating the player that should play in the current turn.

   - **Instance Method(s)**

     `public startGame`
     - Start a new game and play until winning/losing or draw.

     `public printGameBoard`
     - Print out the game board in the Transcript window.

2. **Class** `Player`
   A class representing player. You have to implement it with the following components:

   - **Instance Variable(s)**

     `private id`
     - This is a variable representing the identity of the player (Player X or Player O).

   - **Instance Method(s)**

     `public initialize: id`
     - Initialize the player with its identity.

`public abstract nextMove`
- An abstract method to be implemented in subclasses, which returns the position of next move.

3. **Classes** `Human` and `Computer`
   Classes extending the super class `Player` which represent the human- and computer-controlled player respectively.