CS112 FINAL PROJECT

SOFTWARE DESIGN SPECIFICATION FOR TIC-TAC-TOE GAME

A JAVA PROJECT

Son Ngo

12-5-2015

Contents

1 Introduction 2
2 System Overview 2
3 Design Considerations 2
3.1 User Interface 2
3.2 Game Play 3
3.3 Game Engine 3
3.4 Object-Oriented Programming 3
4 Architectural Strategies 3
5 System Architecture 4
6 Detailed System Design 4

Software Design Specification for Tic-Tac-Toe Game

by Son Ngo

1 Introduction

The objective of this project is to implement a game of Tic-Tac-Toe in Java utilizing Object-Oriented programming techniques.

For this project, I have limited the scope to a game between a user and the computer, where the computer will smartly choose the best moves to beat the human player.

The intended audience is anyone with access to a computer with the JVM installed.

2 System Overview

The user will be presented with a rich GUI of the game. The user will be able to customize how the board, cross, and naught are displayed by changing colors and thickness.

The user will play as O and the computer will play as X. The user and the computer will take turns playing first in each game, with the user playing the first move when the game first starts.

A game ends when either the user or the computer wins (3 X's or O's in a row) or there is no more possible move (a draw).

The user will then be prompted asking whether they want to continue a new game.

3 Design Considerations

3.1 User Interface

I have chosen to implement this program using a Java applet that can the run in a browser, so that anyone with access to a computer and a copy of the JVM installed can play.

3.2 Game Play

For this project, a human play will play against the computer.

Human vs human and computer vs computer may be considered for future development.

3.3 Game Engine

The computer will smartly choose its next move in order to beat the human player.

3.4 Object-Oriented Programming

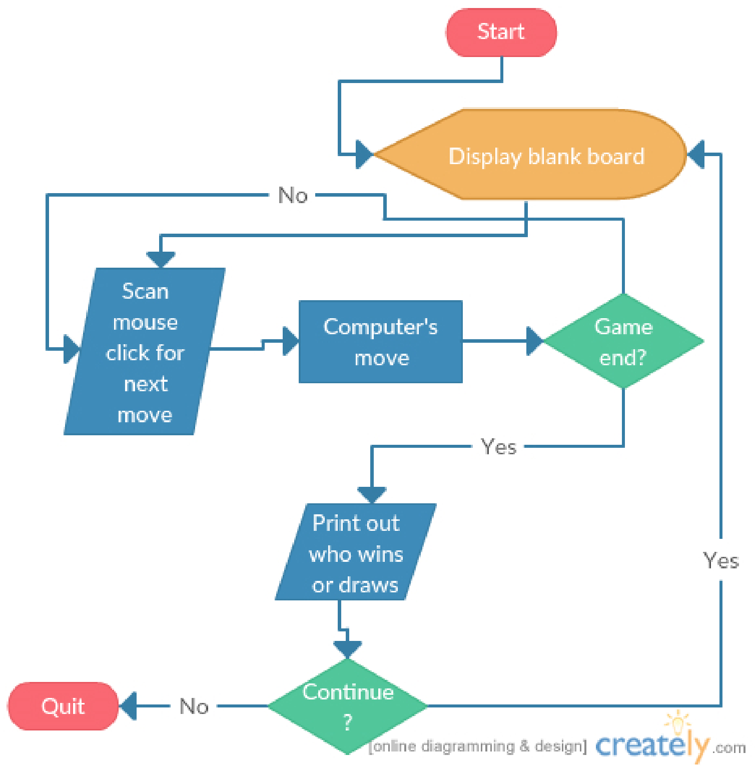
The program will utilize the applet libraries provided by Java, extending from JAplet and JPanel and implementing interfaces to scan mouse clicks for GUI customization and user's moves.

4 Architectural Strategies

Java applet is chosen because it provides me with many tools to easily set up and customize the game board as well as scanning the user's mouse input. In addition, playing with a friendly GUI is much more immersing than having to key in the positions of the moves.

Right now, only user vs computer mode is implemented due to time constraint. User vs user and computer vs computer are subject to future development.

Below is a simply flowchart of how the program would work:


Start
Display blank board
No
Scan mouse click for next move
Computer's move
Game end?
Yes
Print out who wins or draws
Yes
Continue ?
No
Quit
[online diagramming & design] creately.com

5 System Architecture

The program is broken down into multiple methods and classes under the class Tic_Tac_Toe.

As per Java applet implementation, there is an init method to initialize the game.

There are a few methods to customize the interface as per user's request, such as slider to change line thickness and palettes to change colors of X's and O's.

Board is a subclass of Tic_Tac_Toe to actually invoke control of the interface and the gameplay.

Inside Board is a few methods to read mouse clicks, check move's validity, paint the board, issue computer's move, and check winning condition.

6 Detailed System Design

Class	**Method**	**Description**
public class Tic_Tac_Toe **extends** JApplet **implements** ChangeListener, ActionListener	**public void** init()	Initialize the layout
	public void stateChanged(**ChangeEvent** e)	Change line thickness. Take in the value from the slider; set it equal to lineThickness; repaint board
private class Board **extends** JPanel **implements** MouseListener	**public void** paintComponent(**Graphics** g)	Redraw the board
	public void mouseClicked(**MouseEvent** e)	Draw an O where the mouse is clicked
	public void mousePressed(**MouseEvent** e)	Ignore other mouse events
	public void mouseReleased(**MouseEvent** e)	
	public void mouseEntered(**MouseEvent** e)	
	public void mouseExited(**MouseEvent** e)	
	void putX()	Computer plays X
	boolean won(**char** player)	**@param** player: X or O **@return** true if player has won

	boolean testRow(**char** player, **int** a, **int** b)	**@param** player: X or O **@param** a: first position[a] **@param** b: second position[b] **@return** true if player has won in the row from position[a] to position[b]
	void nextMove()	Play X in the best spot
	int findRow(**char** player)	**@param** player: X or O **@return** 0-8 for the position of a blank spot in a row if the other 2 spots are occupied by player, or -1 if no spot exists
	int find1Row(**char** player, **int** a, **int** b)	**@param** player: X or O **@param** a: first position[a] **@param** b: second position[b] **@return** the index of the blank spot if 2 of 3 spots in the row from position[a] to position[b] are occupied by player and the third is blank, else return -1
	boolean isDraw()	**@return** true if all 9 spots are filled
	void newGame(**char** winner)	Start a new game **@param** winner: X or O