Tic-Tac-Toe

Team Name:

Manager:

Recorder:

Presenter:

Analyst:

This is a Process Oriented Guided Inquiry Learning (POGIL) activity. You and your team will examine a working program. A series of questions will guide you through a cycle of exploration, concept invention, and application. There is strong evidence that this is more effective (and less boring) than a traditional lecture.

By the time you are done with this activity, you and your team should be able to:

- decompose a program into static methods with Javadoc comments.
- use multidimensional arrays.
- build simple graphic user interfaces (GUIs).
- manage your team more effectively.

Your team's recorder is responsible for writing your team's answers to the numbered questions on this form.

After you complete this activity, please fill out the short survey at

http://goo.gl/forms/HXjyuUb2ou

to improve this activity for future users.

Playing the game

Open the Tic-Tac-Toe project in Eclipse. Run TicTacToe.java to play the game. This is a two-player game, so you'll have to divide your team into pairs to play.

1. Which members of your team were paired to play together?

2. After these games are done, form different pairs and play a second time. Which members of your team were paired?

3. Is everyone done playing and ready to pay attention to the team?

You may need to go back to the game to answer some of the questions to come, but you should do so *deliberately*, because your team's manager assigned one or more people to find something out, not merely because you got bored with the conversation or thought you could answer a question better on your own.

4. What happens if you try to make an illegal move, e.g., click on an occupied square?

5. Does the program detect the end of the game?

6. Does the program tell you who, if anyone, has won the game?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have.

Static methods and Javadoc comments

Examine TicTacToe.java. This program is broken down into multiple *methods* (functions), each of which starts with the keywords `public static`. Find the `main` method.

7. How did you find it? Would this technique work well in a program with dozens of methods?

Lines 67-72 set up the board. Lines 73-74 do some scaling to simplify the graphics code. We will examine these lines later.

8. What does line 75 do?

9. What do you think line 76 does?

10. If you hold your mouse over the word `draw` on line 76, you get a pop-up tooltip. From where (in this file) did the descriptive text come?

11. Does editing that text change what appears in the tooltip?

12. Does the tooltip still gather the text if the comment starts with `/*` instead of `/**`?

13. When do you think the loop on lines 77-81 ends?

14. What do you think line 78 does?

15. What do you think line 79 does?

16. How does the behavior of the game change if you remove line 79? (Hint: you can use control-/ on a Windows machine or command-/ on a Mac to comment or uncomment one or more selected lines.)

17. How did your team experiment to answer the previous question? Did just one person modify the program or did everyone do it? Why did you do it that way?

18. Do you feel that you understand how this program works at a high level?

19. Do you feel that you understand the details of how things like `handleMouseClick` and `opposite` work?

Let's examine the behavior of one of the other methods. Add the following line to the beginning of `main` (before line 67):

```
StdOut.println(opposite('X'));
```

20. What is printed when you run the program with this modification? (Note that `StdOut.println` prints to the console, not in the graphic window.)

21. What is the value of `opposite('O')`?

Now examine the code for the `opposite` method.

22. Within the method, `opposite` uses an if statement to determine its answer. What does it do with the answer? Does it store it in a variable? Print it? Something else?

23. If you temporarily change the first occurrence of `char` on line 85 to `void`, several red underlines appear in the code, indicating that it no longer compiles. If you hold your mouse over line 87, a pop-up window appears. What does the first line in that window say?

24. In general, how does a method indicate that it is giving back an answer?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have.

Multidimensional arrays

25. Has your team been staying on task?

26. Examine line 67. How does this differ from line 18 in ShutTheBox.java (in the Shut the Box project)?

27. After line 67, what is the value of `board.length`?

28. What is the value of `board[0].length`? Does this change if the two numbers on line 67 are different?

29. What do the nested loops on lines 68-72 accomplish?

30. What do lines 48-50 accomplish?

31. How does the program's behavior change if lines 48 and 50 are removed?

32. After lines 98-105, what is the value of `lines.length`?

33. What is `lines[0].length`?

34. Which part of lines 98-105 represents `lines[0]`?

35. What is `lines[0][0].length`?

36. Which part of lines 98-105 represents `lines[0][0]`?

37. On lines 98-105, what does each pair of numbers represent?

38. What does each group of three pairs represent?

39. How many times (at most) does the loop on lines 106-114 run?

40. What does each pass through the loop accomplish?

41. Under what circumstances does the loop exit early?

42. Under what circumstances does the loop complete all of its passes, allowing the program to proceed to line 115?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have.

Graphic user interfaces

43. Has your team been managing time effectively? Have you been rushing through things, lingering too long on some questions, or getting distracted?

Except for the initial scaling on lines 73-74, all of the code for the graphic user interface occurs in the `draw` and `handleMouseClick` methods.

44. What is the first line of `draw`?

45. What do lines 9-12 accomplish?

46. What are the coordinates of the lower left corner of the graphic window?

47. What are the coordinates of the lower right corner?

48. What are the coordinates of the upper right corner?

49. If lines 73-74 are removed, what are the coordinates of the upper right corner?

50. `StdDraw.text` takes three arguments: two numbers and a String. What do they mean?

51. How does the program's behavior change if lines 38-40 are removed? How do you explain this?

52. How does the program's behavior change if lines 43-45 are removed? (Hint: try holding the mouse down while you play.) How do you explain this?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have.

Please fill out the survey at http://goo.gl/forms/HXjyuUb2ou.