Shut the Box

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:

- declare and create arrays and access their elements.
- use and explain the parts of for loops.
- work more effectively as a team.

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 Shut the Box project in Eclipse. Run ShutTheBox.java to play the game. This is a one-player game, so each member of your team should play once or twice before getting together to answer the questions.

1. 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.

2. What constitutes winning the game?

3. What constitutes an illegal move?

4. Assuming you make no illegal moves but don't win, when does the game end?

5. If you get a score, how is your score determined?

6. Under what circumstances do you roll a different number of dice?

7. Did all team members reach consensus on your answers to the previous questions before you wrote them down? (Your analyst should be watching your team work to help answer process questions like this one.)

Arrays

Examine ShutTheBox.java.

8. Line 23 declares a variable `score`, of type int, and gives it an initial value of 45. What is the type of the variable `closed` declared on line 18?

9. On line 25, the initial value of the variable `roll` is given by the expression `StdRandom.uniform(1, 7)`. What expression gives the initial value of `closed`?

For the next several questions, you will have to temporarily add some code to the program after line 18. You may want to leave a few blank lines around your additions so that you can easily return the program to its original state after you're done.

10. What do you get if you print `closed[3]`?

11. What do you get if you print `closed[30]`?

12. What are the limits on the number you can put between the square brackets in an array indexing expression like those in the two previous questions?

13. What is printed if you add the following two lines?

```
closed[2] = true;
StdOut.println(closed[2]);
```

14. Are the members of your team treating each other with respect?

15. Is `int[]` a valid type?

16. What syntax would you use to create an array *of arrays* of booleans? There are several reasonable guesses. Experiment to find one that works.

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.

For loops

Return your program to its original state by deleting any code you added. The undo function in your editor may be useful.

17. A for loop begins with the keyword `for`. How many for loops appear in this program?

Here is a template for a while loop:

```
while (boolean expression) {
      statement
      ...
}
```

An *expression* is a piece of code that has a value. More specifically, a *boolean expression* is an expression with a value of type boolean, such as `true` or `score == 0`. A *statement* is a "complete sentence" of code that does something, like

```
StdOut.println("Shut the Box");
```

or:

```
int score = 45;
```

18. Write an analogous template for a for loop. Note that there are three parts within the parentheses, separated by semicolons.

19. How many times does the for loop on lines 19-21 run? You can determine this by adding a println statement inside the loop.

20. What is the value of the local variable `i` on the first pass through the loop?

21. What is the value of the local variable `i` on the last pass through the loop?

22. For each of the three parts of a for loop within the parentheses on the first line:

    a. Does it run only once or every pass through the loop?

        i.

        ii.

        iii.

    b. Does it run before or after the statements between the curly braces?

        i.

        ii.

        iii.

23. Is everyone on your team participating in discussions, with nobody left our or dominating? If not, what could you do to improve this?

24. How would you modify the loop so that it prints the numbers counting down from 9 to 1, instead of up from 1 to 9?

25. Is an entire for loop a statement? Explain the reasoning your team used to reach this conclusion.

26. Did your team have any conflicts while working on this activity? If so, how did you resolve them?

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.