Live Coding for Minesweeper Testing:

1. Introduce minesweeperEasier, minesweeperHarder, and have them answer the quiz questions about why one is easier to test than the other.
2. Discuss minesweeperEasier testing.
   1. Show that MinesweeperMain contains all the GUI elements, MinesweeperGame handles the state.
   2. Note that there is a constructor in minesweeperTesting that is purely testing code.
   3. This is a code smell and should be fixed.
3. Pose the question to students: Why would we need both constructors? What is the difference between them.
   1. The first is random, the second allows us to provide the board. We only want to do that for testing.
   2. Code for generating the board should not be deployed when we release the application.
   3. We need to remove the test code from the production application, but how?
4. This is where interfaces are super helpful!
   1. We need to abstract away the generation of the game board, and then provide an object to our MinesweeperGame that represents the new board.
   2. Create the MinesweeperGenerator interface with students (see solution)
   3. Implement the RandomMinesweeperGenerator using code from first constructor
   4. Update MinesweeperGame to have a single constructor that takes the MinesweeperGenerator and gets the board state from it.
   5. Why are we copying the array from the generator instead of just setting it into mines directly?
      1. This ensures that no one has a reference to mines outside of our class to be able to make changes to it.
   6. Update MinesweeperMain to use the new generator
   7. Make sure the application still runs as it did before.
   8. All test code is now gone.
5. Start by testing the model:
   1. Create MinesweeperGameTest
   2. Code createMines and TestGetNeighborMineCount
   3. You have to have a game to do this, so you need to create a new game.
   4. Start by creating a new game the same way we did before, but when you get to the assert you realize you have no idea where the mines will be.
   5. This is where interface comes in handy!
      1. Implement TestGameGenerator that is not random with the students.
      2. Then use that generator in your testing.
   6. Update TestGetNeighborCount with the TestGameGenerator (and write createMines) to illustrate this.
   7. Note why we provided the information for the game size/board layout in main: both classes take different information, but the methods must have the same signature. Providing the information in the constructor solves this.

1. How do we test main?
   1. Create the MinesweeperMainTest class.
   2. Show them the setup and teardown methods.
   3. Use RandomTestGenerator here (as we want to simulate actually running the game), but note that we could use the TestGameGenerator if we wanted to.
   4. Code testInitializePanel and getButtons with them.