CS201 Database of questions for final exam, Part 1 and 2

Here are a few (more) sample questions. More questions will come Saturday evening. They convey the things I will test. In general, to study for the test, you should get the code for TicTacToe and the unit tests in the jar file sent out a few nights ago and those in TicTacToeTester. You should take the TicTacToeTester files and try to build a working tic-tac-toe program (the way you did in the LabExam). This will help you understand the concepts behind the questions in the database.

A bad strategy for testing is to try to memorize the answers. I will change the question numbers, the choices (not the questions) and the order in which the choices are presented. None of these changes will matter if you create the LabExam code yourself from the tests.

You should understand the things being tested for:

a. Compile time errors
   i. does a class exist
   ii. do its constructors exist
   iii. do the methods exist
   iv. are certain tests overridden (?) like equals or toString
   v. does a method need to throw an exception
   vi. does instance data exist
   vii. is the instance data initialized correctly
   viii. is the for loop properly written
   ix. does inheritance come into play (extends vs. implements keyword)
   x. can they throw an exception properly
   xi. can they access an array properly
   xii. can they declare an array variable properly
   xiii. can they initialize an array properly
   xiv. do they understand final, protected, private, public?

b. Run-time errors
   i. does the method return the right thing
   ii. does the method throw an exception
   iii. do local variables overshadow instance data
   iv. are the for loops testing at the right bounds (like, do they go from 0 to maxrows, when they should go from 0 to maxrows-1)
   v. are they accessing an array before initializing it?
   vi. are they accessing an array out of bounds (less than 0, off by one errors)

Only circle one answer for each question. Select the BEST answer from the selection. Do not circle more than one answer for each question or points will be deducted. If you do not know an answer, leave that question blank. Points will be deducted for wrong answers.
BoardTester0.java

1. // BoardTester0.java
2. import java.io.PrintWriter;
3. import java.io.StringWriter;
4. import junit.framework.*;
5. public class BoardTester0 extends TestCase {
6. private Board tttB;
7. private final int MAXROWS = 3;
8. private final int MAXCOLS = 3;
9. protected void setUp() throws Exception {
10. }
11. public void testConstructor() {
12. tttB = new Board(MAXROWS, MAXCOLS);
13. }
14. }

For the following questions, use the code given above.

1. **Choose the best answer.**
   a. BoardTester0 is a subclass of TestCase
   b. BoardTester0 is a superclass of TestCase
   c. BoardTester0 implements TestCase
   d. BoardTester0 inherits from TestCase
   e. 2 or more of above

2. **Assume I add the following statement to testConstructor() after line 12**
   int MAXROWS = 4;
   a. This will cause a compile-time error
   b. This will cause a run-time error
   c. This will cause later uses of MAXROW to possibly be incorrect
   d. Both a and b
   e. none of the above

3. **When the programmer works with the above code "the assignment statement to tttB in testConstructor() will ensure that a Board constructor exists." This will happen at:**
   a. run-time
   b. debug-time
   c. thread-time
   d. compile-time
   e. neither a nor b

4. **The declaration**
   private Board tttB;
   in the Board class above causes what to happen:
   a. a Board object is created whenever a BoardTester0 object is created
   b. a Board reference is created whenever a BoardTester0 object is created
c. nothing, because the data is private and therefore no one can change it
d. none of the above

5. The keyword protected in line 9 means:
   a. all subclasses of TestCase may call the method
   b. only subclasses of BoardTester0 may call the method
   c. anyone may use the method, but only subclasses of BoardTester0 may change it
d. only methods inside BoardTester0 may call the method
e. none of the above

6. Which keyword in the class above makes sure that MAXROW will not be changed after the initial assignment
   a. private
   b. public
   c. protected
   d. final
   e. int

7. Choose the best answer. What does the test in testConstructor() do?
   a. makes sure tttB is not uninitialized
   b. makes sure there is a Board constructor
   c. makes sure there is a 2-argument (int, int) Board constructor
   d. makes sure the 2-argument (int, int) Board constructor creates an NxN board
   e. none of the above

8. When does the test in testConstructor() happen:
   a. compile-time
   b. run-time
   c. on the heap
   d. one the stack
   e. none of the above

9. Write the minimum piece of code that will allow the above test to compile and pass
   ```java
   public class Board {
       public Board(int r, int c) {
   ```

10. If no Board class exists, what is the first line that will not compile in the above class:
    a. 0
    b. 4
    c. 12
    d. 6
    e. none of the above

11. If the Board class exists, but does not include any constructors, what is the first line that will not compile:
    a. 0
    b. 4
    c. 12
    d. 6
    e. none of the above
1.  // BoardTester1.java
2.  import java.io.PrintWriter;
3.  import java.io.StringWriter;
4.  import junit.framework.*;
5.  public class BoardTester1 extends TestCase {
6.    private Board tttB;
7.    private final int MAXROWS = 3;
8.    private final int MAXCOLS = 3;
9.    protected void setUp() throws Exception {
10.       tttB = new Board(MAXROWS, MAXCOLS);
11.       tttB.init();
12.    }
13.    /*
14.       a board will be constructed but not populated,
15.       we should get nulls
16.       */
17.    public void testGetRowColempty() {
18.       tttB = new Board(MAXROWS, MAXCOLS);
19.       String s = null;
20.       try {
21.           s = tttB.getSymbolAtMove(new Move(0, 0));
22.       } catch (MoveException rce) {
23.           rce.printStackTrace(System.out);
24.           assertEquals(true, false);
25.       } finally {
26.           assertEquals(null, s);
27.       }
28.    }

The following questions relate to just the code in BoardTester1.java and not the
code in BoardTester0.java.

12. As written, upon how many user-written classes does BoardTester1 depend to
    compile (don't include BoardTester1)?

   a.  0
   b.  1
   c.  2
   d.  3 (Board, MoveException, Move)
   e.  4

13. To which class does the init() method belong?

   a.  Board
   b.  Move
   c.  MoveException
   d.  Player
   e.  BoardTester1
14. Can we tell if the init() method does anything to the instance data of its class?

a. yes  
b. no  
c. 

15. The 2-arg Board constructor populates its internal representation of a tic-tac-toe board with the following kind of data:

a. references to String objects (not null)  
b. int  
c. char  
d. float  
e. none of the above

16. The assignment on line 21 should result in:

a. an exception being thrown  
b. the assignment being skipped  
c. the assignment of a blank String to s  
d. the assignment of null to s  
e. none of the above

17. Should the call to getSymbolAtMove() on line 21 result in an exception being thrown?

a. yes  
b. no  
c. cannot tell until the code runs

18. What type of exception should getSymbolAtMove() throw?

a. Exception  
b. MoveException  
c. exception  
d. moveexception  
e. none of the above

19. If getSymbolAtMove() throws an exception that is a subclass of MoveException, will the catch-block on lines 21 and 22 be executed?

a. yes  
b. no  
c. can't tell from the code  
d. 

20. If getSymbolAtMove() throws an exception that is not a subclass of MoveException, will the finally block (line 26) be executed?

a. yes  
b. no  
c. can't tell from the code
21. If the compiler indicates an error on line 22 (and nowhere else), which of the following may be a cause of the error?

a. the catch statement is syntactically wrong
b. the MoveException class does not exist
c. the getSymbolAtMove() method is not declared correctly
d. **two of the above**
e. **only one of the above**

22. If the compiler indicates an error on line 22 (and nowhere else), and it indicates that the catch statement is never reached, which of the following is the best cause of the error?

a. the catch statement is syntactically wrong
b. the MoveException class does not exist
c. the getSymbolAtMove() method is not declared correctly
d. **two of the above**
e. **only one of the above** (probably the getSymbolAtMove() method is not declared correctly, i.e., it should **throw** a MoveException and probably does not)

23. For the exception handling code in lines 20-27, which of the following code segments, if inserted into the Board class, will help BoardTester1.java compile?

a. ```java
   public String getSymbolAtMove(Move move) throws MoveException {
       int row = move.getRow();
       int col = move.getCol();
       try {
           if(goodMove(move) == false) {
               throw new MoveException();
           }
       } catch (MoveException me) {
          return tttBoard[0][0];
       }
       return tttBoard[row][col];
   }
   ```

b. ```java
   public String getSymbolAtMove(Move move) throws MoveException {
       int row = move.getRow();
       int col = move.getCol();
       if(goodMove(move) == false) {
           throw new MoveException();
       }
       return tttBoard[row][col];
   }
   ```

c. ```java
   public String getSymbolAtMove(Move move) {
       int row = move.getRow();
       int col = move.getCol();
       try {
           if(goodMove(move) == false) {
               throw new MoveException();
           }
       } catch (MoveException me) {
          return tttBoard[0][0];
       }
       return tttBoard[row][col];
   }
   ```
d. all three of a, b and c

e. only two of a, b and c (a will never throw an exception, but it will allow the compiler to pass the test, b will throw an exception if goodMove() detects a bad move, c has an exception being thrown, but it is caught in the method, so the method will never throw it and the compile can tell this from the signature.)

24. Given that setUp() is called before testGetRowColempty(), is the Board object referred to by tttB on line 21 initialized?
   a. true
   b. false

25. Given that setUp() is called before testGetRowColempty(), but a new Board object is created and not initialized, is the reference variable tttB on line 21 null (true) or not (false)?
   a. true
   b. false

26. Suppose we successfully construct a Board object as in line 18. Given the code in lines 19-27, are we expecting that a method call to getSymbolAtMove will return references to empty Strings ("") or null reference when passed a legal Move object?
   a. empty Strings
   b. null reference
   c. can't tell from code

27. Suppose we have the getSymbolAtMove(Move m) method as tested in lines 19-27. Further, suppose that a call to a method goodMove(new Move(a,b)) returns true when neither player has a piece or symbol at location a,b on the board. Which of the following lines makes this test the best (assume that code above this line has checked to make sure the move is a legal one for the board)
   a. if(board[move.getRow()][move.getCol()] != "") return false;
   b. if(board[move.getRow()][move.getCol()] != null) return false;
   c. tmp=board[move.getRow()][move.getCol()];
      if(tmp= = "" && tmp == null) return true;
   d. tmp=board[move.getRow()][move.getCol()];
      if(tmp! = "" !! tmp != null) return false;
   e. none of the above (a is wrong if we have a null on the board, b is wrong if we have an empty string on the board, c is wrong since it requires both null and empty string, d is wrong since it falsely catches a null on the first part and falsely catches an empty string on the second)

28. Which of the following statements most accurately describes testGetRowColempty()?
   a. the test is a good test to make sure an uninitialized board will return null references when each location is accessed.
   b. the test is a poor test since all we need to do is write a method which always returns null references.
   c. the test is a poor test since all we need to do is write a method which returns a null reference when Move(0,0) is passed in.
   d. the test is a poor test but at least it ensures the programmer must add the throws keyword to the signature of the method.
   e. two or more of the above (a is wrong, but b, c and d are correct)
1. // BoardTester2.java
2. import java.io.PrintWriter;
3. import java.io.StringWriter;
4. import junit.framework.*;

5. public class BoardTester2 extends TestCase {

6.   private Board tttB;
7.   private final int MAXROWS = 3;
8.   private final int MAXCOLS = 3;

9.   protected void setUp() throws Exception {
10.      tttB = new Board(MAXROWS, MAXCOLS);
11.   }
12. }
13. }
14. }
15. }
16. }

29. The "test" of the method init() in testInit() happens at

   a. package-time
   b. run-time
   c. compile-time (at run time, we aren't really testing anything)
   d. can't tell from the code
   e. two from a, b and c

30. Suppose I want the board, tttB, to be initialized for a series of seven (7) tests in one
tester class. The methods in the class would be called testInit(), testConstructor1(),
testSet(), testGet(), setUp(), testConstructor2(), testConstructor3() and
testBoardToString(). The best way to make sure the board is initialized is to:

   a. put a call to init() in testInit() and make sure it is the first test run
   b. put a call to init() at the beginning of each test
   c. put a call to init() in setUp()
   d. put a call to init() in setUp() and then call each test from setUp, one at a time.
   e. use a for loop to call init() seven times.

31. Suppose I have a series of seven (7) tests in one tester class. The methods in the
class would be called testInit(), testConstructor1(), testSet(), testGet(), setUp(),
testConstructor2(), testConstructor3() and testBoardToString(). A reasonable set of calls to the methods by the unit tester would be:

a. setUp(), testBoardToString(), testConstructor1(), testConstructor2(), testConstructor3(), testGet(), testInit(), testSet()
b. setUp(), testInit(), testConstructor3(), testConstructor2(), testConstructor1(), testSet(), testGet(), testBoardToString()
c. none of the above (a reasonable sequence would be setUp(), testBoard(), setUp(), testConstructor1(), setUp(), testConstructor2(), setUp(), etc.)

32. Given lines 10 and 13, how sure can we be that the Board class has any instance data?

a. absolutely, since the Board constructor takes 2 parameters.
b. reasonably sure, since the Board constructor takes 2 parameters and there is an init() method with no return value.
c. not sure at all, since the test is not based on any return value from a member method.
d. absolutely sure it does not, since the test is not based on any return value from a member method.
e. none of the above