1.

Task: 99 Bottles problem

20

The theory class introduced the problem.

Goals

1. To give you hands on experience of unit testing.
2. To let you program considering just readability.
3. To convince you that it is easier to develop and debug using unit test than by manual testing

Materials

You are given a code that contains

1. A test class with one fully implemented test case

```
class ShamelessGreenTest {
   @Test
   void verse0() {
        String expected = "No more bottles of beer on the wall, no more bottles of beer.\n" +
             "Go to the store and buy some more, 99 bottles of beer on the wall.\n";
       ShamelessGreen shamelessGreen = new ShamelessGreen();
       assertEquals(expected, shamelessGreen.verse(0));
   }
}
```

To do

1. Implement the 99 bottles of beer code. Check the correctness of the code using test cases.
2. Implement the verse method. The goal is to make the verse method as readable as possible. As per discussion in the class, one way to define readability is, *how well the code reflects the problem it is solving*. From the code, we want to easily identify -
   a. How many verse variants are there?
   b. Which verses are most alike? In what way?
   c. Which verses are most different, and in what way?

2. Implement other methods to test the whole song and check the correctness as well.

Restrictions

You are not allowed to write the main method. Run the test cases instead.

Rubrics for marking

Complete code without main method	10
Writing test cases (partially)	5
Wrting test cases that can cover whole song (You do not have to write every test cases from 3-99)	5

2.

20

Task: Shape problem

You have to draw different shapes like circles, rectangles, etc. in a GUI application. The circles and rectangles must be drawn in a particular order. A list of the circles and rectangles will be created in the appropriate order and the program must walk the list in that order and draw each circle or rectangle.

Goals

1. To understand the problem of bad code (i.e., naturally comes to your mind)
2. To understand the benefit of the Open Close Principle (OCP)
3. To use unit testing for debugging and testing purposes.

Materials

You can think of a Canvas class that is able to draw shapes. For simplicity use just text to identify different shapes.

To do

1. You have to write two solutions. One is following the OCP and the other is not. First complete the bad code.

2. Add another shape Triangle to draw without breaking the existing code.
3. Write the problems of bad code in the format of comments.

Restrictions

You are not allowed to write the main method. Run the test cases instead.