Real FizzBuzz

The problem

Please write code delivering the requirements of the steps that follow. The requirements don't mention a command line application, and we are not looking for one. Do write code that you would be happy delivering to a paying client.

You should not find this test to be particularly difficult. It is designed to be a straightforward coding exercise, and it should take you no more than 90 minutes.

Step 1:

Write a class that produces the following for any contiguous range of integers:

- the number
- 'fizz' for numbers that are multiples of 3
- 'buzz' for numbers that are multiples of 5
- 'fizzbuzz' for numbers that are multiples of 15

Being careful to avoid trailing spaces.

e.g. Running the program with a range from 1-20 should produce the following result:

```
1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16 17 fizz 19 buzz
```

Archive this as a separate .zip file then continue on to step two.

Step 2:

Please enhance your existing FizzBuzz solution to perform the following:

- If the number contains a three you must output the text 'lucky'. This overrides any existing behaviour

e.g. Running the program with a range from 1-20 should produce the following result:

```
1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz
```

Archive this as a separate .zip file then continue on to step three.

Step 3:

Finally, please enhance your existing solution to perform the following:

- Produce a report at the end of the output showing how many times the following were printed:
    - fizz
    - buzz
    - fizzbuzz
    - lucky
    - an integer

e.g. Running the program with a range from 1-20 should produce the following result:

```
1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz
fizz: 4 buzz: 3 fizzbuzz: 1 lucky: 2 integer: 10
```

(Integer is 10 because there were 10 numbers that were not altered).

Archive this as a separate .zip file and send all three zip files as email attachments to **derby.benson@intersection.com**, following the guidance set out below.

Please include the version number `925cf7f189d1e404cfd55df6e8398d831df0f45a` in a README file with your submission so we know what version of the instructions you've been given.

Note: Please *don't* include the compiled code in the zip files. We won't use it, and it causes trouble with many email systems.

Please do not submit Dropbox or Google Drive links. They don't meet the requirements of our anonymous submission policy.

Note for Gradle Users: Normal best practice would be to include the gradle wrapper in your build script. However, this causes the project to include .jar and .bat files, which can cause trouble with a lot of email systems. You don't need to include a gradle wrapper with your project. The reviewer will have the latest version of gradle installed on their system if gradle is used.

What we are looking for:

Test Coverage: The solution should be developed "test-first", and should have excellent unit tests and test coverage.

Build file: Please provide an automated build file that compiles your code and runs the tests. For java submissions, a Gradle or Maven build file is ideal. Submissions without an automated build will not be accepted.

Simplicity: We value simplicity as an architectural virtue and a development practice. Solutions should reflect the difficulty of the assigned task, and should not be overly complex. Layers of abstraction, patterns, or architectural features that aren't called for should not be included.

Self-explanatory code: The solution you produce must speak for itself. Multiple paragraphs explaining the solution are a sign that it isn't straightforward enough to understand purely by reading code, and are not appropriate.

This version number: As mentioned above, please include the version number `925cf7f189d1e404cfd55df6e8398d831df0f45a` in a README file with your submission so we know what version of the instructions you've been given.