Project 1: FizzBuzz

Contents

New concepts: Eclipse IDE, I/O, `if` statements, loops, ... Java!

1 Program description

Write a program that plays the old fashioned FizzBuzz game, which is now a classic programming example in the computer science world. The game works as follows. Several people get in a circle and count off numbers sequentially (e.g., 1, 2, 3, 4, . . .). Before counting, two numbers are selected: a "Fizz" number (let's call it f) and a "Buzz" number (let's call it b). When counting off, the player must say "Fizz" for numbers divisible by f and "Buzz" for numbers divisible by b. Numbers divisible by both f and b must be substituted with "FizzBuzz". The game goes around the circle until someone messes up.

Your job is to write a program that gets values f and b from the user, and then prints out numbers substituting "Fizz", "Buzz"', and "FizzBuzz" as described above. The user also gets to pick the numbers at which to start and end counting. The program quits when either f or b is zero.

2 Error messages

- `ERROR: Negative numbers are not allowed! Try again:`
  when the user enters a Fizz or Buzz number that is negative
- `ERROR: Ending number cannot be less than starting number! Try again:`
  when the user enters an ending number that is smaller than the starting number

3 Required elements

Your project must use the each of the elements described in this section. You can have more functions, variables, and objects if you want, but you must at least use the elements described in this section.

3.1 Variables

- Global `public static BufferedReader` as **the only** `BufferedReader` object in the entire program for reading user input
  - Although your programs will compile and run just fine on your own computer with multiple `BufferedReader` objects, they will not run correctly on a web server with multiple `BufferedReader` objects. Since your programs will be graded on a web server, please don't have more than one `BufferedReader` for reading input from the console.

4 Helpful hints

- Remember, you will be graded on whether or not your program produces **EXACTLY** the same output as the provided solution program—including spaces, spelling, punctuation—everything counts! So, to make sure your output is identical to the solution output, copy and paste segments from the solution output (e.g., the menu) directly into your own source code.
- Test everything. What happens if you enter a negative number somewhere? Or a letter instead of a number? Or a number instead of a letter? Or a decimal number instead of an integer? Be creative!
- If a particular action makes the solution program crash (e.g., entering a letter instead of a number), then your program can crash, too. We won't test any scenarios that cause the solution program to crash, so it does not matter what your program does in those scenarios. Think of it as a freebie.