Quiz 1

Debrief

Quiz 1 as presented

- Some slight improvements to the Java style here:

```
import java.util.function.DoubleFunction;

public class Newton {
    public static void main(String[] args) {
        // Newton's Approximation to solve cos(x) = x
        newton("cos(x) - x = 0", 1.0, 200, (double x ) -> Math.cos(x) - x , (double x ) -> - Math.sin(x) -
1 );
    }

    private static void newton(final String equation, final double x0, final int maxTries, final
DoubleFunction<Double> f, final DoubleFunction<Double> dfbydx) {
        double x = x0;
        int tries = maxTries;
        for (; tries > 0; tries--) {
            final double y = f.apply(x);
            if (Math.abs(y) < 1E-7) {
                System.out.println("The solution to "+equation+" is: " + x);
                System.exit(0);
            }
            x = x - y / dfbydx.apply(x);
        }
    }
}
```

- Quiz 1 is available on Blackboard. It shouldn't take you long to finish it now.

Student comments

1. The parameter values of the variables are hard-coded in the program. If the user wants to change the value at run-time, it is not possible. For example, the initial value x0, maxTries, f(x) and f'(x) are directly present in the function of attribute instead of the user being asked at run-time.

   Yes, this is true, although at least they are in the *main* program and not hard-coded in the *newton* method. We would like to be able to vary the parameters at run-time. That should make it easier to allow them to vary at run-time.

2. Also, the value of x_0 is not random but closer to the first zero which is not specified.

   It would be better if the value of x_0 was random, perhaps chosen from a user-defined range. That way, some entropy would be introduced to the system to help prevent always finding a local minimum rather than the global minimum.

3. The value of x chosen should be not be at a stationary point or zero, otherwise, the solution to the equation will become indeterminate in reality.

   This is essentially the same point as (2).

5. The program is not updated with comments to enable the user knows the restrictions on values or the use of each section of the code.

   True although I'm not a fan of unnecessary comments.

6. The derivative of the function is to be computed by the user and only then it can be used in the program, unless the user knows math to compute the derivative, the program cannot proceed.

   True although I don't have a good idea how to avoid this. Note that we can immediately see the limitation of the method: the function must be differentiable!

Student comments

1.It does not provide users with the error info to let them know what is wrong when the program b reaks. For example, it the maxTries are too small, the for loop will end before the program being a ble to print out the answer.

Yes, in my opinion, this is the most serious issue.

2.It is impossible to know if the "dfbydx" is compatible with “f”.

Yes, this is a significant problem. I’m not sure we can fix that in Java. We need a functional programming language like Scala.

3. The real problem is that this program does not notice that "dfbydx" cannot be 0!

Yes, the program will throw an exception in this case. In some ways, this is better than going into an infinite loop but it’s never good to throw an exception unless we handle it well (here, we don’t).

5. There should be a *toString* method to show the medium steps/logs/values to help users understand the algorithm.

Yes, logging is always helpful when a program fails.

6. Tolerance 1E-7 is too large for double values. The precision should be changed to 1E-15 for most possible accurate result.

Yes, this gets to the core of one of the problems. It’s not so much that it’s too large (or too small). The real issue is that different functions will need different tolerances. One tolerance can never cover all possibilities. Making a tolerance as small as 1E-15, BTW, could easily cause the algorithm to hit maxTries.

Student comments

1. The "Newton" method is static,which is not ‘object-oriented'.
   Yes, good.
2. There's no return value of this method, but only ouput, and the output can not be used by user
   A very good point. It also means that the algorithm cannot be unit-tested.
3. When we use exactly 200 times with"maxTries" but without getting a right answer, there's no message to that effect to the user.
   I thought this was the most egregious error. But there are many :)
4. n my opinion it is acceptable if a program is not user-friendly because a program that directly appeared to users should be well encapsulated and decorated with interactions. Programs are black boxes for users, but not for programmers. Thus, in order to debug or understand this algorithm, there should be functions used to reset the parameters and all possible results of this program should be presented by output.
   This is an interesting take on the aspect of user-friendliness. Of course, you are right. My use of “user” was very much the programmer here, the one invoking *newton,* rather than the one launching the *main* program. But I didn’t make that very clear. But one essential tenet of software is the concept of a contract (usually this takes the form of an API—application programming interface). But between the program and the end-user, there is also a contract, one that includes issues such as response time. We will be heavy users of APIs in this class.

My issues

1. A minor issue is that it isn't as object-oriented as it could be. We should make better use of classes. We could, for instance, have a class which basically encapsulates the problem we're trying to solve, including *f, dfbydx* and default values for x_0, the tolerance and *maxTries*.
2. Like many algorithms, there are quite a few configuration values that have to be compatible. We need a better handle on these, as many students pointed out. However, the big problem to me is that the behavior may not be clear from the result.
3. If we do hit *maxTries*, it could be because of one of several problems:
    1. the tolerance may be impossibly small;
    2. the algorithm might be oscillating (or diverging) instead of converging;
    3. *maxTries* itself could be too small for the problem being worked on;
    4. because we hit a local minimum and we need to try again from a different starting point.