Advanced Programming (FEB23007-14)

3. Exercise

Deadline for submission: 2014-02-08 23:59 CET

Instructions

Include in each source file (in class documentation, @author) your names and student numbers. Submit the exercise as a zip file containing _only_ the source files in root of the zip. Submit via blackboard. Note that incorrectly submitted or non-compiling exercises are automatically awarded 0 points. Remember to document your code with Javadoc-annotations.

Exercise

The bisection algorithm (see `http://en.wikipedia.org/wiki/Bisection_method` for description and pseudo-code) is a well-known algorithm in mathematics used to find the root of a function in a particular interval. In this exercise, we want to implement this algorithm in Java and apply it on linear and quadratic functions. A linear function is described as

$$f(x) = ax + b,$$

where a and b are commonly known as the slope and intersect respectively, whereas a quadratic function can be described as

$$f(x) = ax^2 + bx + c.$$

We will make use of an interface, since the functions have similar functionalities. For making the program in an object-oriented manner, you should implement (at least) the following standard/unit testing classes:

- `Function`: an interface class containing the general function evaluation method with signature `double evaluate(double at).`
- `LinearFunction`: a class representing a linear function. The class should have a constructor that takes the slope and intercept as input, be immutable, and implement `Function`.
- `LinearFunctionTest`: a class for unit testing `LinearFunction` public methods.
- `QuadraticFunction`: a class representing a quadratic function. The class should have a constructor that takes the coefficients a, b, and c as inputs, be immutable, and implement `Function`.
- `QuadraticFunctionTest`: a class for unit testing `QuadraticFunction` public methods.
- `BiSectionMethod`: a class that contains the bisection method. The class should contain a static bisection method that takes as input a `Function` and the interval for evaluation. It should return the value at which the function evalutes as 0. Make sure to document and check for the pre-condition for a valid interval, as the root of $f(\cdot)$ can only be found in an interval $[x_1, x_2]$ where $f(x_1)$ and $f(x_2)$ have different signs. Make the bisection method parameters TOL and NMAX as public constants within the class (`public static final`).
- `BiSectionMethodTest`: a class for unit testing `BiSectionMethod` public methods.
- `Main`: a main class that constructs functions $f(x) = 2x + 3$, $g(x) = 7x^2 - 3x - 10$ and applies the bisection algorithm on them at different intervals.