CS3318 Continuous Assessment

In the continuous assessment you will develop a Java library for numerical root-finding of real valued univariate functions (a function that takes a single real-valued number). Your work will be assessed as follows

- The design of the interfaces
- The documentation of the interfaces
- Testing of the implementation

In addition, you will write a JavaFX application to plot univariate functions.

Exercise 1: Designing Interfaces

Design an interface for a Univariate function. You can assume that the Double type is used to store all values. You need to decide the appropriate Java mechanism to define the interface (abstract class or interface) and if any supporting exceptions need to be defined.

Design an interface for a Univariate Function Root Solver. You can assume that the Double type will be used to store all values. You need to decide the appropriate Java mechanism to define the interface (abstract class or interface), the methods defined in the interface and if any supporting exceptions need to be defined.

Your API should enable a client to interrogate the expected performance of their chosen method. The performance is assessed in two parts, the convergence (how quickly the method should reach the root) and the reliability (if the method is guaranteed to find the root). You can assume that all methods conforming to your API will require that the client supply an initial bracket (an upper and lower bound) containing a single root.

You should provide the following exemplar implementations

- FalsePositionSolver that implements the false position method
- SecantSolver the implements the secant method

Note that the

- FalsePositionSolver convergence is linear and the method is guaranteed to find the root (if the bracket contains a root)
- SecantSolver convergence is super-linear and the method is not guaranteed to find the root (even if the bracket contains a root)

Your API should define appropriate exceptions (type and checked/unchecked).

A question to consider is how can you ensure that a client will program to the interface and not to an implementation?

Exercise 2: Documenting The API

Document using Javadoc the APIs you defined above. There is no need to document the implementations (FalsePositionSolver and SecantSolver).

Exercise 3: Unit Testing

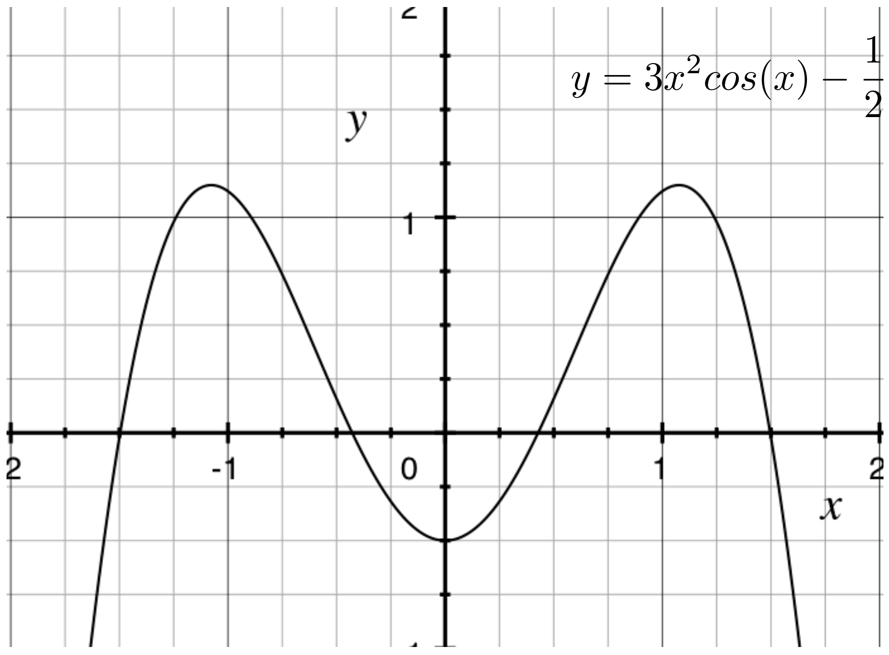
Write unit tests to test the implementation of the FalsePositionSolver class.

Exercise 4: JavaFX Function Plotter

Determining the initial bracket for a function can be difficult. Write a JavaFX application to allow a user to display a univariate real-valued function between a start value and an end value. The example below shows the plot of the function

$$y = 3x^2 cos(x) - \frac{1}{2}$$

Between x = -2 and x = 2.



Notes

Be sure to use good team-oriented development practice, self-documenting programs adhere to the Java coding idioms and conventions.

You will submit for

- **Exercise 1**: The Java source code for the APIs you defined for the Univariate Function and the Univariate Function Root Solver, and any other types your solution requires (e.g. specialised exception classes)
- **Exercise 2**: The Java source code that includes the Javadoc comments and generated HTML documents
- **Exercise 3**: The Java source code for your Junit test suite
- **Exercise 4**: The Java source code for your JavaFX application and any other resources your solution requires

Submission is via the Moodle website and can be submitted anytime from now until **Friday 1st December 2017 at 23:00**.