Today we will try to implement basic functions integration.

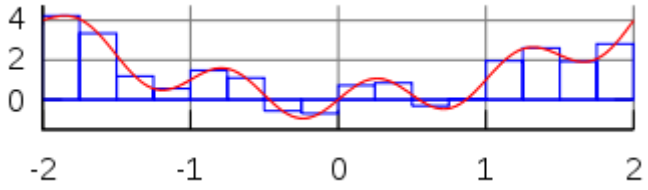
Notes: Create each class/interface in its own file.

Steps:

1. First let's prepare IFunction interface for all functions we want to use. It should have:
   - Evaluate(double x) method, will return function value evaluated at point x (e.g. Evaluate(3) for Sinus should return sin(3).
   - Method Positions(double from, double to, uint n) that will return IEnumerable of function values (doubles) evaluated at n, equally spread points on defined domain (e.g. from=0, to =1, n=10 should return function values at points 0, 0.1, 0.2, …, 0.9)

   We will also need some basic function. Let's go with sinus:
   - Create sinus class with defined amplitude, frequency and phase. Implement IFunction for this class (f= Amplitude*sinus(frequency*X + phase).

   Note: We can omit boundary checks in this task. In general case evaluate method should probably somehow check if argument is in domain that can be evaluated.
2. Prepare similar class: Polynomial (quadratic function, given by a,b,c coefficients).
   - If you feel adventurous, prepare Polynomial class accepting and evaluating coefficient of any degree.
   - Prepare explicit conversion operators. One accepting double and creating const polynomial, 2nd accepting 2-element double for linear function and 3-element tuple for quadratic function.
3. Prepare two classes that will be combining our functions: FunctionSum and FunctionProduct:
   - FunctionSum should accept two IFunction objects in constructor and evaluate to sum of these two functions values (should work like e.g. x^2+sin(x)).
   - FunctionProduct should also accept two IFunction objects, but evaluate to product of function values (should work like e.g. (x^2+3)*sin(2x+1)).
4. Prepare basic integrator using rectangle numerical integration in class :
   - Example invocation can be found in Main.



   -
   - Basic idea of integration algorithm can be derived from picture above. Result would be sum of each rectangle area. One rect width is equal to selected dt (smaller -> more accurate results). Rect height is equal to value of function "somewhere" in the rectangle.
   - Result would be: split integration domain [From, To] into N parts of equal length. For each part get value of integrated function, multiply it by part length. Sum all results and return,

- Algorithm should be implemented in public static method Integrate in class RectangleIntegrator accepting 4 parameters: IFunction func, double from, double to and uint n). Where n is mentioned number of parts. Name of
- If you feel adventurous add bool flag to select if function value should be evaluated at the beginning of step, or in the middle. Set default value as beginning.

5. Write CumulativeIntegrate function that will return IEnumerator with values representing cumulative integral (sum of all restults until current step. On picture above it would be: 0, sum of 1 rectangles, sum of 2 rectangles, etc.). Function parameters the same as in standard Integration method.