Programming Technologies and Paradigms. Lab 2 presential activity. Groups 1 and 4.

1. Create a new solution with a console application project. Figure out a way to add to this solution previous lab project “Geometry” where Point2d was defined. You will find useful these examples: `utility.classes` in `encapsulation`, and `parameter.passing`, `named.optional.parameters` in `overload`. Implement the following static methods in the console application project:

1. IncrementTwoDouble. Takes four double parameters, first and second by reference. Modifies these two parameters adding third and fourth parameters value respectively, returns `void`. For instance, this call: IncrementTwoDouble(ref x,ref y,1.5,2.0) adds 1.5 to x and 2.0 to y.
2. GenerateTrajectory. Takes a length parameter (int, default value 10), start coordinates x and y (double, default value both 0.0), x and y increment (double, default value both 1.0). Returns a trajectory, stored in a Point2d array with the number of elements length, where all the points except the first one are obtained adding to the previous point in the trajectory the indicated increments using previously defined function IncrementTwoDouble. See `arrays` project in solution `encapsulation`.
3. ShowTrajectory, takes an array representing a trajectory as described previously and an optional boolean, true if only first quadrant points are to be shown (x>=0 and y>=0) false otherwise. Outputs trajectory points textual representation to console.
4. DistanceStartEnd, takes a Point2d array and, as output references, three parameters: trajectory covered distance (the summation of the distances between each point and the following one), starting and ending points. ¿What happens with the trajectory array values if returned starting and ending points properties are modified after function call? Output the trajectory to check it.
5. Define a Point2d array in Main holding a trajectory, check if the implemented functions and argument passing mechanisms work as expected.

2. Learn about `extension.methods` in `overload` solution. Extend int with the following methods (the examples are written using constants for the sake of simplicity):

1. Reverse, returns the int obtained reversing caller digits order: 123456.Reverse() returns 654321 as int.
2. IsPrime, returns true if the caller is prime, false otherwise: 13.IsPrime() returns true, 14.IsPrime() returns false as boolean.
3. Gcd and Lcm, optional: 24.Gcd(36) is 12, again an integer.
4. Concatenate, optional: 123.Concatenate(456) returns 123456 as int. Caution when the second one is negative, choose any solution for this case, drop the sign or whatever.
5. Check if everything works as expected in a console application project.

3. Use the provided source code to create a unit test project. Implement the requested class along with the methods and operators in order to pass the tests. See `operators` in `overload`.

Upload the mandatory activity before the next Lab.