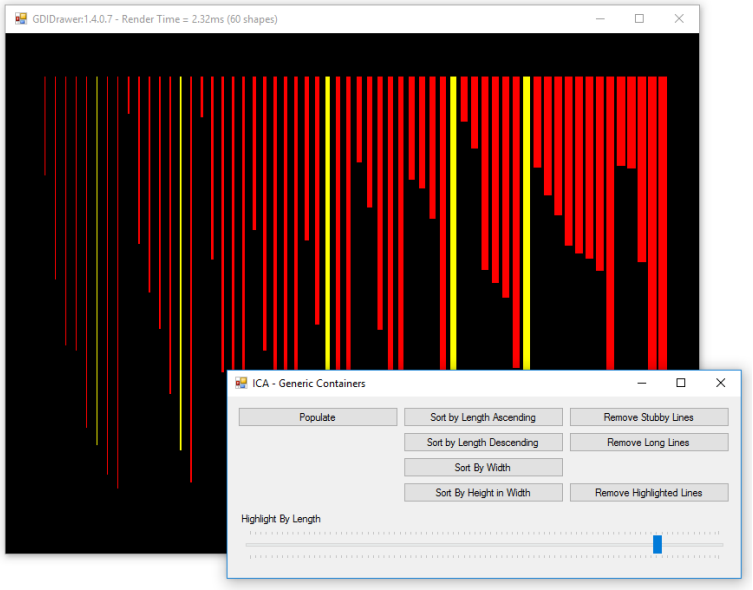
CMPE2300 – ICA #07 – Funline – Predicates/Comparisons/Lambdas (2017)

In this simple ICA, you will exercise some basic operations of the `List` collection.



FunLine

You will create a new user-defined class called `FunLine`. This class will contain fields for length and width as `int`, and a `bool` as a highlight flag, all with `internal` access.

Create a constructor to accept and initialize the length and width. Assume the highlight flag will start `false`.

Add a public method for rendering that will accept a `CDrawer` and a `Point`. Render the line (as a rectangle) starting at the position specified, extending to the width and height of this line. Render in yellow if highlighted, red if not.

Main Form

In your main form class, add fields for a `CDrawer`, a static `Random`, and a `List<FunLine>`.

Add a rendering method that will step through all the `FunLine` elements in the list and render each one. Render them in the order they appear in the list.
The coordinates for each line will be `45 + index * 12, 50`.

To the 'Populate' button event handler, clear the list. Add 60 random `FunLine` objects to the list. `FunLine` objects should be 25-500 (inclusive) pixels in length, and 1-10 (inclusive) pixels wide. Render.

For the 'Sort by Length Ascending' button, call `Sort` in your list. Provide no arguments to `Sort`. Modify your `FunLine` class to make this operation work correctly. Render.

For the 'Sort by Length Descending' button, call `Sort` in your list. This time, provide a `System.Comparison<FunLine>` compliant function to order the `FunLine` objects in descending order of length. Place the static implementation of this function in your `FunLine` class. Render.

For the 'Sort By Width' button, call `Sort` in your list. This time, provide a `System.Comparison<FunLine>` compliant lambda expression to order the `FunLine` objects in ascending order of width. Render.

For the 'Sort By Height in Width' button, call `Sort` in your list. This time, provide a `System.Comparison<FunLine>` compliant function to order the `FunLine` objects in ascending order of height in width. Place the static implementation of this function in your `FunLine` class. Render.

For the 'Remove Stubby Lines' button, call `RemoveAll` in your list. This time, provide a `System.Predicate<FunLine>` compliant function to indicate `FunLine` objects that have a length that is less than `75`. Render.

For the 'Remove Long Lines' button, call `RemoveAll` in your list. This time, provide a `System.Predicate<FunLine>` compliant lambda expression to indicate `FunLine` objects that have a length that is greater than `325`. Render.

The track bar is designed to highlight lines that are close to the length indicated by the track bar position. Set the track bar range to match the min and max length of `FunLine` objects at any time the list of lines is modified. Create a method to contain this code, as you will be calling it from several places. You must use the `Min` and `Max` extension methods in this method to find the shortest and longest line.

In the scroll event for the track bar, do the following:

- Use the `ForEach` *method* of `List` with a lambda expression to clear the highlighting on all lines
- Use the `FindAll` method of `List` with a lambda expression to identify lines that are within 10 pixels of the track bar `Value`. Set the highlight flag for the identified lines (the resultant collection).
- Render

The 'Remove Highlighted Lines' button will do just that – with a lambda expression. Render.

Your code will be visually inspected to ensure that you have implemented all components of this assignment as directed.