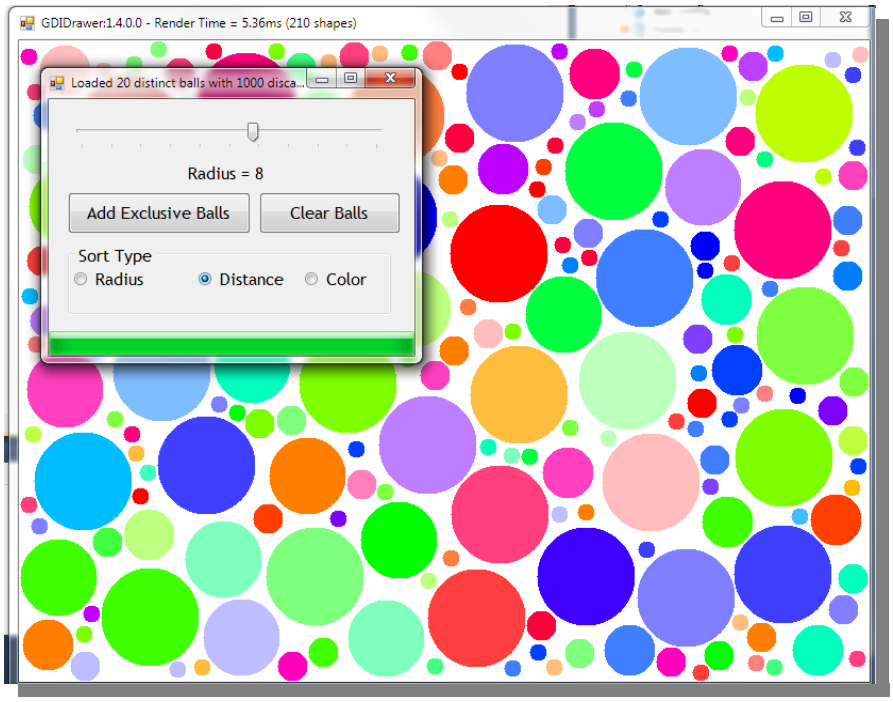
CMPE2300 – ICA05 – Be Ordering My Balls

In this ICA you implement the **IComparable** interface to allow your balls to be sorted AND alter the way our balls radius is handled.



Start by copying your ica04, and we will add some functionality via the **IComparable** interface.

Now, indicate the support of the **IComparable** interface on the class definition

Add a public enumeration called ESortType with values eRadius, eDistance, and eColor

Add a public static Automatic property of type ESortType (We will use this to determine how to sort our Balls)

Add the **CompareTo**() method to complete your contract with the IComparable interface
If the argument is null or not a Ball, throw a new exception of type ArgumentException, with an appropriate message

- using the as operator, get a Ball reference from your object parameter
- Without using multiple returns, calculate the appropriate return value depending on the state of the current SortType property
- Radius is just that, eDistance would be the distance from the origin (0,0) to the ball center, you can use the Color.ToArgb() integer value for sorting the color.
- this method should enable us to do selective dynamic sorting, you should test each sort type as you go

Add a new Radio buttons as indicated, create a click handler for one, and bind all the radio buttons to the same handler.

- For whichever radio button is currently checked, set the appropriate value in the Balls static SortType property.
- Sort your collection
- Now to display them in order slowly so we can see the ordering, set loading to true and,
- for each ball, AddBall() the ball, Sleep() for 1ms, then set loading to false (this will render each ball enabling us to see the sort condition easily

*Yes, by putting the Loading = false within the loop, it will fire unnecessarily (thereby calling Render() of the CDrawer), but it will allow us to see the Balls rendered in slow-mo in their current order in the list.