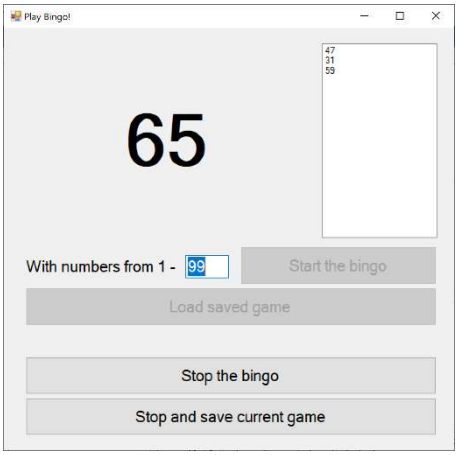
Assignment 1: Refactoring a God-class



This assignment requires you to extend an existing Bingo application. The bingo app display unique random numbers between a user defined range and will stop automatically when all numbers have been shown.

Currently, the Bingo app allows a user to:

- Start a new game
- Load and start a previous game
- Stop a game
- Stop and save a game

This must be extended with requirements to keep track of all Bingo game sessions:

- **FR1: Possibility to view prior Bingo games**
  When a game is stopped the data of that game should be kept available to the user to view & restart at a later point.
- **FR2: View and restart any of the prior Bingo games**
  A user can select and re-start an old game
- **FR3: Saving and loading should save all Bingo games**
  The save and load functionalities should save and load all the Bingo games.

Step 1

Download the Bingo application from Canvas and analyze it so you understand what the application does and how the code matches with the current functionalities. You will notice that the application works correctly but does not apply proper OO design. Everything is in the Form-class (i.e. the God-class) and there is quite some code duplication.

Based on the current state of the code, can you imagine how you can include the new requirements? What might be the reasons that it is not going to be easy to do that?

Instead of continuing with the badly design code, you should first refactor it to a proper object-oriented design; continuing with a bad base (i.e. the foundation of your application) will usually result in to messy and har to maintain code.

The first step is to split the code up into the *three layers*: 1) Presentation layer, 2) Logic layer and 3) Data access layer. Get started by determining what new classes you need, and then start moving code from the Form-class into these classes.

Step 2

The next step is to determine how you should implement the *FR1: possibility to view prior Bingo game* and *FR2*: *View and restart any of the prior Bingo games*. If you did step 1 correctly, you should have one class with the logic for a Bingo-game; we will refer to this class as *Bingo*-class.
The Bingo-class responsibility is to keep track of which numbers have been drawn and drawing the next number. In other words, it represents one Bingo game session.

Try to determine how we can implement *FR1* and *FR2* while keeping the *single responsibility principle* in mind. At some point you will conclude that a new class has to be introduces and this class is responsible for managing all the Bingo game session. Create this class and decide for yourself

what members it should have. When you are done with the class, you should start changing the GUI of the application to make use of the new class.
For example, include a ComboBox with a button where all Bingo game session are shown and the user can select/view them.

Step 3

At this point only *FR3: Saving and loading should save all Bingo games* is left. If you did step 1 correctly, you have one class saving and loading a *Bingo*-object. To implement *FR3* you will have to change that *data access* class.

For this requirement there are no technical constraints about what the format of the serialization of all the Bingo game session must be (i.e. does not have to be a text-file). Determine for yourself how you want to serialize the objects and implement it.