Inheritance Practice

These questions are severely contrived, and are not necessarily something you would see in practice. Their purpose is to determine if you can follow instructions carefully and execute the required implementations, as you will see during the exam for real.

Question 1

- Create a .NET Forms application.
- Derive a new class from **`Timer`**.
- Add a public automatic property called **`Ticks`**, **`private set`**, initialized to zero.
- Create only one constructor that accepts an **`int`**. Use this value to set the timer **`Interval`**. Subscribe to the **`Tick`** event, and enable the timer.
- In the **`Tick`** hander, increase the **`Ticks`** value.
- In the main form, create a member of your derived timer type, initialized to **`null`**.
- In the **`Load`** event, create the derived instance with **`100`** as the interval.
- If the user clicks in the form, show the **`Ticks`** count in the caption.

Question 2

- Create in interface called **`ISummy`** that will define one function: returns **`object`**, called **`GetSum`**.
- Create a class called **`Ints`** that conforms to **`ISummy`**.
- **`Ints`** will contain a **`static Random`** object.
- **`Ints`** will contain a **`List`** of **`int`**.
- In the constructor, accept an **`int`**, populate the collection with a number of random values from **`0`** to **`99`**, where the count matches the argument provided.
- Implement **`GetSum`**, returning the sum of the collection.
- Implement **`ToString`**, return the sum in square brackets.
- Test **`Ints`** to ensure that it works as intended.

Additionally

- Create a class that derives from **`Ints`** called **`ShowInts`**. **`ShowInts`** will also conform to **`ISummy`**.
- Using the minimum quantity of additional code, add a constructor that satisfies the base class requirements.
- Using the minimum quantity of additional code, provide a **`ToString`** implementation that will return the collection sum, as well as all the collection elements, comma separated. This will appear in the form:
  `[446] : 52, 85, 63, 44, 31, 2, 36, 82, 25, 26`

Additionally

- As test code, create a **`List`** of **`ISummy`**. Populate the list with **`100`** **`ISummy`** types, with a **`50`**/**`50`** chance of each element being **`Ints`** or **`ShowInts`**. Each instance will contain **`10`** elements.
- Use **`string.Join`** to show the contents of the collection in the console, with "**`\r\n`**" as the separator.