Consider the following two interfaces:

1. `IFrameCallBack`

```
public interface IFrameCallback
{
    public void FrameReceived( IntPtr pFrame, int width, int height );
}
```

`FrameReceived` is a callback method that will be called by a native library every time a new frame is available from an image capture device i.e. a camera.

`pFrame` is the pointer to the raw bytes of the frame, `width` and `height` denotes the size of the frame in pixels.

`pFrame` is reused by the native library to store the next frame as soon as the callback completes.

This callback method is guaranteed to be called periodically depending on the frame rate configured on the capture device, in this case it is 30FPS.

2. `IValueReporter`

```
public interface IValueReporter
{
    public void Report( double value );
}
```

Report is a callback method that is implemented by an external library that will plot the given value in a real-time chart.

A software intern is assigned a task to write a program to fetch the frame from the camera, calculate the average pixel value, and stream back the calculated value..

This is the code the intern came up with:

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
```

```
using System.Timers;

namespace Formulatrix.Intern.GrabTheFrame;

public interface IFrameCallback
{
  public void FrameReceived( IntPtr pFrame, int pixelWidth, int pixelHeight );
}

public interface IValueReporter
{
  public void Report( double value );
}

public class FrameCalculateAndStream
{
  private IValueReporter _reporter;
  private Queue<Frame> _receivedFrames = new Queue<Frame>();
  private Timer _timer;

  public FrameCalculateAndStream( FrameGrabber fg, IValueReporter vr )
  {
    fg.OnFrameUpdated += HandleFrameUpdated;
    _timer = new Timer( 1000 / 30 );
    _timer.Elapsed += OnTimerElapsed;
    _reporter = vr;
  }

  private void HandleFrameUpdated( Frame frame )
  {
    _receivedFrames.Enqueue( frame );
  }

  private void OnTimerElapsed( object sender, ElapsedEventArgs e )
  {
    if( _receivedFrames.Count > 0 )
    {
      Frame frame = _receivedFrames.Dequeue();
      byte[] raw = frame.GetRawData();

      // https://stackoverflow.com/questions/29312223/finding-the-arithmetic-mean-of-an-array-c-sharp
      int sum = 0;
      for( int i = 0; i < raw.Length; i++ )
```

```
        sum += raw[i];

      int result = sum / raw.Length; // result now has the average of those numbers.

      _reporter.Report( result );
    }
  }

  public void StartStreaming()
  {
    _timer.Enabled = true;
  }
}

public class FrameGrabber : IFrameCallback
{
  private byte[] _buffer;
  public delegate void FrameUpdateHandler( Frame rawFrame );
  public event FrameUpdateHandler OnFrameUpdated;

  public void FrameReceived( IntPtr frame, int width, int height )
  {
    if( _buffer == null )
      _buffer = new byte[width * height];

    // https://stackoverflow.com/questions/5486938/c-sharp-how-to-get-byte-from-intptr
    Marshal.Copy( frame, _buffer, 0, width * height );

    Frame bufferedFrame = new Frame( _buffer );
    OnFrameUpdated( bufferedFrame );
    bufferedFrame.Dispose();
  }
}

public class Frame : IDisposable
{
  private bool _disposed;
  private byte[] _rawBuffer;

  public Frame( byte[] raw )
  {
    _rawBuffer = raw;
  }
```

```
  public byte[] GetRawData()
  {
    if( _disposed )
      throw new ObjectDisposedException( "underlying buffer has changed, should not be used anymore" );

    return _rawBuffer;
  }

  public void Dispose()
  {
    _disposed = true;
  }
}
```

As a software engineer who has more experience, you are tasked to review the intern's code.

1. What would you say about the above code?
2. What sorts of problems does this code have?
3. How can this code be improved?

If you were to be the one originally assigned to this task, or you are given the opportunity to reimplement from scratch, how would you do it? Provide a sample code.