Класс для осаждения информации

```
using HTMLSteganographyWinFormV2.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.IO.Compression;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HTMLSteganographyWinFormV2
{
   class Embedder
   {
      public static void EmbedMessage(XMLFile xml, string embedingMessage, int
bitsOfOneCharacter)
      {
         string message = MakeBinaryString(embedingMessage, bitsOfOneCharacter);

         int embedCounter = 0;

         message += IntToStringOneSymbolBitsMapper.map[bitsOfOneCharacter];

         for (int i = 0; i < xml.File.Length-1 && embedCounter < message.Length; i++)
         {
            if(xml.File[i] == '\"')
            {
               if(message[embedCounter] == '0')
               {
                  while (xml.File[++i] != '\");
                  embedCounter++;
               }
               else if (message[embedCounter] == '1')
               {
                  xml.File[i] = '\'';

                  while (xml.File[++i] != '\")
                  {
                     if (xml.File[i] == '\'')
                     {
                        xml.File[i] = '\';
                     }
                  }

                  xml.File[i] = '\'';

                  embedCounter++;
               }
            }
            else  if (xml.File[i] == '\'')
```

```
            {
                if (message[embedCounter] == '1')
                {
                    while (xml.File[++i] != '\"') ;
                    embedCounter++;
                }
                else if (message[embedCounter] == '0')
                {
                    xml.File[i] = '\';
                    while (xml.File[++i] != '\"')
                    {
                        if(xml.File[i] == '\')
                        {
                            xml.File[i] = '\"';
                        }
                    }

                    xml.File[i] = '\';

                    embedCounter++;
                }
            }
        }
    }

    public static string MakeBinaryString(string charString, int bitsOfOneCharacter)
    {
        StringBuilder binaryString = new StringBuilder();

        foreach (char item in charString)
        {
            int symbolCode = (int)item;
            string binarySymbolCode = Convert.ToString(symbolCode, 2);

            if (binarySymbolCode.Length != bitsOfOneCharacter)
            {
                int addedBits = bitsOfOneCharacter - binarySymbolCode.Length;
                for (int i = 0; i < addedBits; i++)
                {
                    binarySymbolCode = "0" + binarySymbolCode;//выравнивание длины строки
до 8 бит
                }
            }
            binaryString.Append(binarySymbolCode);
        }
        return binaryString.ToString();
    }

    public static void AddStegoContainerToArchive(string archive, string containerName,
string containerContent)
    {
        if (System.IO.File.Exists(archive))
```

```
            {
                System.IO.Compression.ZipArchive apcZipFile =
System.IO.Compression.ZipFile.Open(archive,
System.IO.Compression.ZipArchiveMode.Update);

                ZipArchiveEntry entry = apcZipFile.CreateEntry(containerName);

                StreamWriter writer = new StreamWriter(entry.Open());
                writer.WriteLine(containerContent);
                writer.Flush();

                apcZipFile.Dispose();
            }
        }
    }
}
```

Класс для извлечения информации

```
using HTMLSteganographyWinFormV2.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HTMLSteganographyWinFormV2
{
    class Extracter
    {
        public static string ExtractMessage(XMLFile xml, int bitsInOneCharacter)
        {
            StringBuilder extractedMessage = new StringBuilder();

            try
            {
                for (int i = 0; i < xml.File.Length - 1; i++)
                {
                    if (xml.File[i] == '\')
                    {
                        extractedMessage.Append("0");
                        while (xml.File[++i] != '\')
                        {
                            if (FileCrashedByZero(extractedMessage))
                            {
                                return RestoreMessage(extractedMessage.ToString(), bitsInOneCharacter);
                            }
                        }
                        continue;
                    }
                    if (xml.File[i] == '\'')
```

```
                {
                    extractedMessage.Append("1");
                    while (xml.File[++i] != '\"')
                    {
                        if (FileCrashedByOne(extractedMessage))
                        {
                            return RestoreMessage(extractedMessage.ToString(), bitsInOneCharacter);
                        }
                    }
                    continue;
                }
            }
        }
        catch (IndexOutOfRangeException e)
        {
            return e.Message;
        }


        return RestoreMessage(extractedMessage.ToString(), bitsInOneCharacter);
    }

    private static string RestoreMessage(string extractedMessage, int bitsInOneCharacter)
    {
        StringBuilder restoredMessage = new StringBuilder();

        for (int i = 0; i < extractedMessage.Length; i += bitsInOneCharacter)
        {
            try
            {
                string messageSymbol = extractedMessage.Substring(i, bitsInOneCharacter);
                if (messageSymbol ==
IntToStringOneSymbolBitsMapper.map[bitsInOneCharacter]) break;

                char restoredSymbol = (char)Convert.ToInt32(messageSymbol, 2);
                restoredMessage.Append(restoredSymbol);
            }
            catch (ArgumentOutOfRangeException)
            {
                return restoredMessage.ToString();
            }
        }
        return restoredMessage.ToString();
    }

    private static bool FileCrashedByOne(StringBuilder message)
    {
        bool crashingStatus = false;
        if (message.Length >= 16)
        {
            if (message[message.Length - 1] == '1'
        && message[message.Length - 2] == '1'
```

```
        && message[message.Length - 3] == '1'
        && message[message.Length - 4] == '1'
        && message[message.Length - 5] == '1'
        && message[message.Length - 6] == '1'
        && message[message.Length - 7] == '1'
        && message[message.Length - 8] == '1'
        && message[message.Length - 9] == '1'
        && message[message.Length - 10] == '1'
        && message[message.Length - 11] == '1'
        && message[message.Length - 12] == '1'
        && message[message.Length - 13] == '1'
        && message[message.Length - 14] == '1'
        && message[message.Length - 15] == '1'
        && message[message.Length - 16] == '1')
            {
                crashingStatus = true;
            }
        }


        return crashingStatus;
    }

    private static bool FileCrashedByZero(StringBuilder message)
    {
        bool crashingStatus = false;
        if (message.Length >= 16)
        {
            if (message[message.Length - 1] == '0'
        && message[message.Length - 2] == '0'
        && message[message.Length - 3] == '0'
        && message[message.Length - 4] == '0'
        && message[message.Length - 5] == '0'
        && message[message.Length - 6] == '0'
        && message[message.Length - 7] == '0'
        && message[message.Length - 8] == '0'
        && message[message.Length - 9] == '0'
        && message[message.Length - 10] == '0'
        && message[message.Length - 11] == '0'
        && message[message.Length - 12] == '0'
        && message[message.Length - 13] == '0'
        && message[message.Length - 14] == '0'
        && message[message.Length - 15] == '0'
        && message[message.Length - 16] == '0')
            {
                crashingStatus = true;
            }
        }

        return crashingStatus;
    }
}}
```

FileManager

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.IO.Compression;
using HTMLSteganographyWinFormV2.Models;

namespace HTMLSteganographyWinFormV2
{
   class FileManager
   {

      private static readonly string tempArchiveStorage = "./1.zip";
      private static readonly string xmlFileContainsMainMarkup = "word/document.xml";

      public static string ReadXMLFile(string path)
      {
         return File.ReadAllText(path);
      }

      public static void WriteXMLFile(XMLFile XMLFile, string path)
      {
         File.WriteAllText(path, XMLFile.File.ToString());
      }

      public static void CopyFileAndChangeExtentionToZip(String file)
      {
         DeleteTempArchive(tempArchiveStorage);
         File.Copy(file, Path.ChangeExtension(tempArchiveStorage, ".zip"));
      }

      public static void CopyFileAndChangeExtentionToDOCX
         (String file, string fileDestination)
      {
         File.Copy(file, Path.ChangeExtension(fileDestination, ".docx"),true);
      }



      public static void DeleteTempArchive(string tempArchive)
      {
         File.Delete(tempArchive);
      }

      public static string ReadDocumentFromZipFile(String filePath)
      {
         String fileContents = "";
         try
```

```
        {
            if (System.IO.File.Exists(filePath))
            {

                System.IO.Compression.ZipArchive apcZipFile =
System.IO.Compression.ZipFile.Open(filePath,
System.IO.Compression.ZipArchiveMode.Read);
                foreach (System.IO.Compression.ZipArchiveEntry entry in apcZipFile.Entries)
                {
                    if (entry.FullName == xmlFileContainsMainMarkup)
                    {
                        System.IO.Compression.ZipArchiveEntry zipEntry =
apcZipFile.GetEntry(entry.FullName);

                        using (System.IO.StreamReader sr = new
System.IO.StreamReader(zipEntry.Open()))
                        {
                            fileContents = sr.ReadToEnd();
                        }
                    }
                }
                apcZipFile.Dispose();
            }
        }
        catch (Exception)
        {
            throw;
        }
        return fileContents;
    }

    public static void RemoveDocumentFilefromZipArchive(string archivename, string
removingDocument)
    {
        using (Stream stream = File.Open(archivename, FileMode.Open))
        {
            using (ZipArchive archive = new ZipArchive(stream, ZipArchiveMode.Update, false,
null))
            {
                ZipArchiveEntry entry = archive.GetEntry(removingDocument);
                if (entry != null)
                {
                    entry.Delete();
                }
            }
        }
    }
}
```

Главная форма

```
using HTMLSteganographyWinFormV2.Models;
using System;
using System.Text;
using System.Windows.Forms;

namespace HTMLSteganographyWinFormV2
{
   public partial class Form1 : Form
   {
      private XMLFile xmlContainer;
      private DOCXFile docxContainer;
      private string endOfMessageFlag = "!@#";
      private XMLFile HTMLFileWithMessage = new XMLFile();
      private DOCXFile DOCXFileWithMessage = new DOCXFile();
      public Form1()
      {
         InitializeComponent();
      }

      private void openHTMLFile_Click(object sender, EventArgs e)
      {
         OpenFileDialog openFileDialog = new OpenFileDialog();
         //openFileDialog.Filter = "Markup files (*.jsp, *.html, *.xml,
*.zip)|*.jsp;*.html;*.xml;*.zip";
         openFileDialog.Title = "Select a Container";

         if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
         {
            int indexOfPoint = openFileDialog.FileName.IndexOf('.');

            string fileExctension = openFileDialog.FileName.Substring(indexOfPoint);

            string filePathToContainer = openFileDialog.FileName;

            if (fileExctension == ".xml")
            {
               xmlContainer = new XMLFile();

               xmlContainer.File = new StringBuilder(
                              FileManager.ReadXMLFile(
                                       filePathToContainer));

               containerCapacity.Text = (xmlContainer
                           .GetContainerCapacity(Convert.ToInt32(bitsInOneSymbolTextBox.Text))
                           - endOfMessageFlag.Length)
                           .ToString();

            }
            else if (fileExctension == ".docx")
            {
```

```
            try
            {
                FileManager.CopyFileAndChangeExtentionToZip(openFileDialog.FileName);

                string xmlDocument = FileManager.ReadDocumentFromZipFile("./1.zip");

                docxContainer = new DOCXFile();
                docxContainer.document = new XMLFile();
                docxContainer.document.File = new StringBuilder(xmlDocument);

                containerCapacity.Text = (docxContainer.document
                    .GetContainerCapacity(Convert.ToInt32(bitsInOneSymbolTextBox.Text))
                    - endOfMessageFlag.Length)
                    .ToString();
            }
            catch (Exception)
            {
                MessageBox.Show("Ошибка в структуре документа");
                FileManager.DeleteTempArchive("./1.zip");
            }
        }
        else
        {
            MessageBox.Show("Указан неверный тип файла");
        }

        openedContainerLabel.Text = filePathToContainer;
    }
}

private void embedMessage_TextChanged(object sender, EventArgs e)
{
    if (docxContainer != null)
    {
        containerCapacity.Text = (docxContainer.document
            .GetContainerCapacity(Convert.ToInt32(bitsInOneSymbolTextBox.Text))
            - endOfMessageFlag.Length
            - embedMessage.Text.Length)
            .ToString();

    }
    else if (xmlContainer != null)
    {
        containerCapacity.Text = (xmlContainer
            .GetContainerCapacity(Convert.ToInt32(bitsInOneSymbolTextBox.Text))
            - endOfMessageFlag.Length
            - embedMessage.Text.Length)
            .ToString();
    }
}

private void embedMessageButton_Click(object sender, EventArgs e)
```

```
    {

        if (xmlContainer != null)
        {
            if (Convert.ToInt32(containerCapacity.Text) >= 0)
            {
                Embedder.EmbedMessage(xmlContainer, embedMessage.Text,
                                Convert.ToInt32(bitsInOneSymbolTextBox.Text));

                SaveFileDialog saveFileDialog1 = new SaveFileDialog();

                saveFileDialog1.Filter = "txt files (*.html)|*.html|All files (*.*)|*.*";
                saveFileDialog1.FilterIndex = 2;
                saveFileDialog1.RestoreDirectory = true;


                if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    FileManager.WriteXMLFile(xmlContainer, saveFileDialog1.FileName);
                    MessageBox.Show("Встраивание завершено");
                }
            }
            else
            {
                MessageBox.Show("В контейнере нет места");
            }
        }
        else if (docxContainer != null)
        {
            if (Convert.ToInt32(containerCapacity.Text) >= 0)
            {
                if (embedMessage.Text.Length == 0)
                {
                    MessageBox.Show("Введите сообщение");
                    return;
                }
                Embedder.EmbedMessage(docxContainer.document, embedMessage.Text,
                                Convert.ToInt32(bitsInOneSymbolTextBox.Text));
                FileManager.RemoveDocumentFilefromZipArchive("./1.zip",
"word/document.xml");

                Embedder.AddStegoContainerToArchive("./1.zip", "word/document.xml",
docxContainer.document.File.ToString());

                SaveFileDialog saveFileDialog1 = new SaveFileDialog();

                saveFileDialog1.Filter = "txt files (*.html)|*.html|All files (*.*)|*.*";
                saveFileDialog1.FilterIndex = 2;
                saveFileDialog1.RestoreDirectory = true;

                numberOfUsedQuates.Text = Embedder.MakeBinaryString(embedMessage.Text,
Convert.ToInt32(bitsInOneSymbolTextBox.Text)).Length.ToString();
```

```
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
               //FileManager.WriteHTMLFile(xmlContainer, saveFileDialog1.FileName);
               FileManager.CopyFileAndChangeExtentionToDOCX("./1.zip",
saveFileDialog1.FileName); //по идее должно катать
               FileManager.DeleteTempArchive("./1.zip");
               MessageBox.Show("Встраивание завершено");
            }
         }
         else
         {
            MessageBox.Show("В контейнере нет места");
         }
      }


      docxContainer = null;
      xmlContainer = null;
   }

   private void extractMessageFromHTMLButton_Click(object sender, EventArgs e)
   {


      OpenFileDialog openFileDialog = new OpenFileDialog();
      openFileDialog.Filter = "Markup files (*.html, *.xml, *.docx)|*.html;*.xml;*.docx";
      openFileDialog.Title = "Select a HTML Container";

      if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
      {
         string filePathToContainer = openFileDialog.FileName;
         string fileExtention =
filePathToContainer.Substring(filePathToContainer.IndexOf('.'));

         if (fileExtention == ".xml")
         {
            HTMLFileWithMessage.File = new StringBuilder(
                  FileManager.ReadXMLFile(
                        filePathToContainer));

            string extractedMessage = Extracter.ExtractMessage(HTMLFileWithMessage,
Convert.ToInt32(bitsInOneSymbolTextBox.Text));
            openedContainerLabel.Text = filePathToContainer;
            extractedMessaageTextBox.Text = extractedMessage;
         }
         else if (fileExtention == ".docx")
         {
            FileManager.CopyFileAndChangeExtentionToZip(openFileDialog.FileName);

            string xmlDocument = FileManager.ReadDocumentFromZipFile("./1.zip");
```

```
                DOCXFileWithMessage = new DOCXFile();
                DOCXFileWithMessage.document = new XMLFile();
                DOCXFileWithMessage.document.File = new StringBuilder(xmlDocument);

                string extractedMessage =
Extracter.ExtractMessage(DOCXFileWithMessage.document,
Convert.ToInt32(bitsInOneSymbolTextBox.Text));
                openedContainerLabel.Text = filePathToContainer;
                extractedMessaageTextBox.Text = extractedMessage;
                FileManager.DeleteTempArchive("./1.zip");
            }


        }
    }

    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        FileManager.DeleteTempArchive("./1.zip");
    }
  }
}
```

Дополнительные преобразования

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HTMLSteganographyWinFormV2
{
  class IntToStringOneSymbolBitsMapper
  {
    public static Dictionary<int, string> map =
                new Dictionary<int, string>() {

        { 8, "00000000" },
        { 11, "00000000000" }
  };
  }
}
```