Задача 1

Есть ли в реализации класса какие-то проблемы и недостатки? Если есть,  то перечислите какие и предложите изменения:

public class Processor

{

    public void ProcessFile(string filename)

    {

        Stream fileStream = File.OpenRead(filename);

        Console.WriteLine(ReadAllContent(fileStream));

        fileStream.Close();

    }

    public string ReadAllContent(Stream stream)

    {

        StreamReader streamReader = new StreamReader(stream);

        return streamReader.ReadToEnd();

    }

}

Ответ: Необходимо также закрыть объект streamReader. (streamReader.Close())

Задача 2

Будет ли работать реализация GetFileHandler в многопоточной среде. Поясните свой ответ

  public class FileHandler

    {

        public static Stream GetFileHandler()

        {

            if (\_file!=null)

                return \_file;

            lock(\_lock)

            {

                if (\_file != null)

                    return \_file;

                \_file = File.OpenRead("name");

                return \_file;

            }

        }

        private static object \_lock = new object();

        private static Stream \_file;

    }

Ответ. Да. Без ключевого слова [lock] в методе GetfileHandler в разных потоках переменная \_file может возвращать разное значение. **lock** не позволит ни одному потоку войти в раздел кода где изменяется \_file в тот момент, когда в нем находится другой поток. Таким образом, значение в \_file будет одинаково во всех потоков.

Задача 3

Есть класс, который занимается обработкой файлов разного формата. Предполагается, что будут добавляться новые форматы файлов. Какие вы видите проблемы в реализации.

Проанализируйте реализацию данного класса и предложите рефакторинг, позволяющий расширять форматы обрабатываемых файлов.

  public class FileProcessor

    {

        enum FileType

        {

            Html,

  Text

        }

        public void ProcessFile(string fileName)

        {

            StreamReader fileStream = new StreamReader(File.OpenRead(fileName));

            string fileContent = fileStream.ReadToEnd();

            if (fileContent.IndexOf("<html") != -1)

                ProcessHtmlFile(fileContent);

            else ProcessTextFile(fileContent);

            ProcessFile(fileContent,fileContent.IndexOf("<html")!= -1?FileType.Html:FileType.Text);

            fileStream.Close();

        }

        private void ProcessFile(string content,  FileType fileType)

        {

            switch(fileType)

            {

                case FileType.Html:

                    ProcessHtmlFile(content);

                    break;

                case FileType.Text:

                    ProcessTextFile(content);

                    break;

                default:

                    throw new Exception("Unknown file format");

            }

        }

        private void ProcessHtmlFile(string content)

        {

            ....

        }

        private void ProcessTextFile(string content)

        {

            ....

        }

    }

Ответ. При расширении придется изменять enum FileType + добавлять дополнительные методы в класс FileProcessor. Это приведет к дополнительной модификации каждый раз, когда будет добавлен новый процессор. (Если процессоров станет много – то FileProcessor вырастет).

Рефакторинг включит в себя создание контейнера-словаря простых классов процессоров, реализующих интерфейс

(interface IProcessor { void Process(string content); })

В качестве ключей словаря можно использовать enum, в качестве значений – процессоры.

Можно также использовать IocContainer для регистрации классов- процессоров.

Задача 4

Есть  клиент, который считывает сообщения из сети по TCP/IP. Сообщения представляют собой XML и бывают трех типов (A,B,C).Клиент занимается обработкой сообщений.

В процессе обработки сообщений происходит их разбор и сохранение в постоянном файловом хранилище. Кроме того, подсчитывается статистика  по обработке сообщений. Предложите свой вариант дизайна реализации(достаточно показать идею).

Ответ. Для каждого типа сообщения необходимо свой тип обработчика (MessageHandler<T>, где T – тип сообщения).

Также необходим класс, который в зависимости от типа входящего сообщения будет вызывать нужный обработчик (напр. MessageDispatcher) и вычислять статистику.

Для автоматического создания все классы:типы сообщений, Handler’ы и dispatcher регистрируются в IoC-контейнере.

Создается TcpClient, который начинает принимать сообщения (в сообщении передается xml). Пусть в структуре xml существует тег [messageType]. Анилизируя этот тег Ioc-контейнер будет создавать нужное сообщение. Далее сообщение будет передаваться параметром в Dispatcher.Dispatch (IMessage msg). Dispatcher найдет нужный обработчик для сообщения и запустит его выполнение, также посчитав статистику.

Подробная реализация находится по адресу: <https://github.com/BlackMoon/Luxoft/tree/master/Lk>