IFBA – Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Ciência da Computação
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas
INF011 – Padrões de Projeto
Prof.: Frederico Barboza **Data:** 13/11/2023

Aluno: _____ Nota: _____

IIª Avaliação – 2023.2

As questões desta avaliação são baseadas no seguinte cenário de classes:

*Sua empresa está construindo um **middleware** de troca de mensagens baseado em protocolos da camada de aplicação da internet já bem estabelecidos. O **middleware** foi construído de forma a dar suporte a diversos protocolos, tais como FTP, SMTP, HTTP, etc. O próximo passo é garantir que as mensagens trocadas através destes protocolos (representadas pelas classes Request e Response), possam ser enviadas pelos hosts utilizando diversos mecanismos de criptografia e/ou compactação. A granularidade desejada é no nível de mensagem. De modo que uma mesma conexão aberta, possa ter a primeira mensagem sendo enviada como texto puro, enquanto a seguinte pode ir como texto criptografado usando **AES**, a seguinte ser compactada usando **Huffman**, e, na sequência, termos uma mensagem criptografada e compactada usando **RSA** e **LZ78**, por exemplo. Um requisito é que os algoritmos de compactação e criptografia possam ser combinados livremente (inclusive com uma mensagem sendo criptografada duas ou mais vezes e/ou compactada duas ou mais vezes).*
O diagrama que apresenta as interfaces do modelo está apresentado na figura 1.

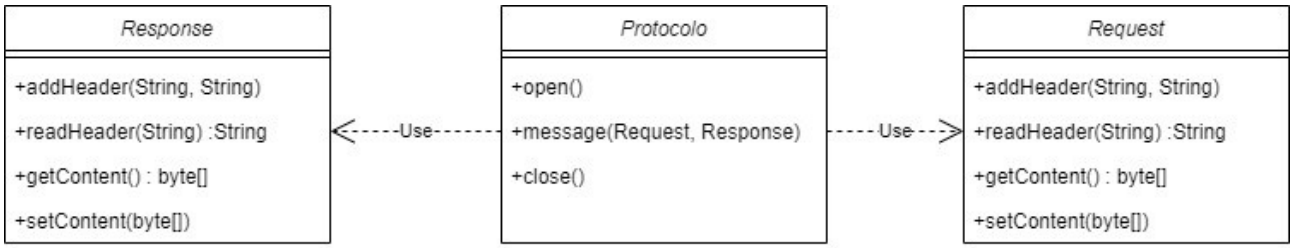

Figura 1: Interfaces do Middleware

*Para garantir, o requisito apresentado, o processo de compactação e de criptografia deve alterar a funcionalidade fornecida pelo método **`getContent()`**, modificando o conteúdo retornado, através do uso dos métodos de compactação e/ou criptografia desejado.*
O código, para o exemplo da introdução, deve seguir a seguinte estrutura

```
ProtocoloAbstractFactory af = new HTTPFactory();
Protocolo protocolo = af.getProtocolo("http.inf011.ifba.edu.br", Integer.valueOf(80));
Request request = af.getRequest("REQUEST");
Response response = af.getResponse();
protocolo.open();
Request request1 = af.getRequest("MENSAGEM LIMPA");
protocolo.message(request1, response);
Request request2 = … // MENSAGEM QUE DEVE SER CRIPTOGRAFADA COM AES
protocolo.message(request2, response);
Request request3 = … // MENSAGEM QUE DEVE SER COMPACTADA COM HUFFMAN
protocolo.message(request3, response);
Request request4 = … // MSG QUE DEVE SER CRIPTOGRAFADA COM RSA E COMPACTADA COM LZ78
protocolo.message(request4, response);
protocolo.close();
```

Quadro 1

QUESTÃO I

Apresente uma solução baseada em padrões de projeto, que resolva a questão da aplicação dos algoritmos de compactação e criptografia para as mensagens trocadas entre os hosts, de acordo com os requisitos indicados. Apresente a solução APENAS para a interface / classes que representem as mensagens de requisição (Request), dado que, a solução pode ser aplicada por simetria as mensagens de Resposta (interface Response)

O padrão de criptografia avançada - advanced encryption standard (AES), também conhecido por seu nome original Rijndael, é uma especificação para a criptografia de dados eletrônicos estabelecida pelo instituto nacional de padrões e tecnologia dos E.U.A. (NIST) em 2001. É um dos algoritmos de criptografia mais seguros do mundo. Sua empresa possui uma classe que criptografa mensagens usando AES256. Um excerto desta classe é apresentada no código java a seguir.

```
public class AES256 {

       /*
       * Método para a inicialização do AES através de SecretKey
       * gerada de forma aleatória através de KeyGenerator
       */
        public void init(){...};

       /*
       * Método para a inicialização do AES através de chave fornecida
       * de forma expícita
       */
        public void init(byte[] chave){...};

       /*
       * Método que faz a cifragem de message usando a chave fornecida/gerada
       * por um dos métodos init. Chamar este método sem inicializar o
       * algoritmo através de uma das duas formas indicadas,
       * lança a SecurityException
        */
       public byte[] doFinal(byte[] message) throws SecurityException{...};

}
```

QUESTÃO II

Novamente, podendo modificar livremente o modelo de classes, e sem perder as alterações decorrentes da aplicação do padrão da questão I, escolha e apresente novo padrão, que permita a utilização da classe indicada acima no *middleware* que está sendo construído, permitindo o seu uso respeitando os requisitos indicados no cenário descrito previamente.

ATENÇÃO

- As soluções devem ser esboçadas em código JAVA
- Caso modifique as relações entre as classes do modelo, apenas apresente a nova assinatura da classe, por exemplo: `public class Classe extends ClasseAbstrata implements Interface1, Interface2{};`
- Identifique **CLARAMENTE** qual o padrão que será aplicado em cada questão, bem como quais são os papéis (Participantes) no padrão, assumidos pelas classes do projeto.