Resposta Estrutura com Strategy
Emanuel Gomes e Marco Antonio

1. **Interface** que define o contrato de compressão:

```
interface CompressionStrategy {
    void compress(String fileName);
}
```

2. **Estratégias concretas**:

```


class RarCompression implements CompressionStrategy {
    @Override
    public void compress(String fileName) {
        System.out.println("Comprimindo " + fileName + " em formato
RAR...");
    }
}

class SevenZCompression implements CompressionStrategy {
    @Override
    public void compress(String fileName) {
        System.out.println("Comprimindo " + fileName + " em formato
7Z...");
    }
}

class ZipCompression implements CompressionStrategy {
    @Override
    public void compress(String fileName) {
        System.out.println("Comprimindo " + fileName + " em formato
ZIP...");
    }
}
```

3. **Classe Contexto (`Compressor`)** que usa a estratégia:

```
class Compressor {
    private CompressionStrategy strategy;

    public void setStrategy(CompressionStrategy strategy) {
        this.strategy = strategy;
    }
```

```
    public void compressFile(String fileName) {
        if (strategy == null) {
            System.out.println("Nenhuma estratégia definida!");
            return;
        }
        strategy.compress(fileName);
    }
}
```

4. Exemplo de uso no `main`:

```
public class Main {
    public static void main(String[] args) {
        Compressor compressor = new Compressor();

        // ZIP
        compressor.setStrategy(new ZipCompression());
        compressor.compressFile("documento.txt");

        // RAR
        compressor.setStrategy(new RarCompression());
        compressor.compressFile("imagem.png");

        // 7Z
        compressor.setStrategy(new SevenZCompression());
        compressor.compressFile("video.mp4");
    }
}
```

Diagrama Simplificado (UML)

```
              +----------------------+
              |  CompressionStrategy |
              |----------------------|
              | + compress(fileName) |
              +----------^-----------+
                         |
    ---------------------------------------------
    |                    |                      |
+----------------+   +----------------+   +----------------+
| RarCompression |   | SevenZCompression |   | ZipCompression |
|----------------|   |-------------------|   |----------------|
| + compress()   |   | + compress()      |   | + compress()   |
+----------------+   +-------------------+   +----------------+

                   +----------------+
```

```
                    |   Compressor   |
                    |----------------|
                    | - strategy     |
                    |----------------|
                    | + setStrategy()|
                    | + compressFile()|
                    +----------------+
```

Justificativa

- O **padrão Strategy** elimina condicionais (`if/else` ou `switch`) centralizadas, movendo a lógica para **classes específicas de cada algoritmo**.
- O contexto (`Compressor`) não precisa conhecer os detalhes de cada compressão, apenas delega para a estratégia configurada.
- A manutenção melhora, pois para **adicionar um novo formato** (ex.: `TarCompression`), basta criar uma nova classe que implemente `CompressionStrategy`, sem alterar código existente (aplicando **Open/Closed Principle**).
- O sistema fica mais **flexível**: a escolha do algoritmo pode ser **alterada em tempo de execução**.