**Introdução**

Este relatório descreve a solução desenvolvida para o problema de encontrar a maior sequência de caixas aninhadas, utilizando um catálogo de caixas com dimensões especificadas. O problema envolve determinar quantas caixas podem ser colocadas uma dentro da outra, de modo que cada caixa subsequente seja menor em todas as dimensões que a anterior.

**Descrição do Problema**

O objetivo é automatizar a tarefa de identificar a maior sequência de caixas aninhadas a partir de um catálogo de caixas de papelão. Cada caixa é caracterizada por três dimensões: largura, altura e comprimento. A tarefa consiste em encontrar o comprimento máximo de uma sequência onde cada caixa subsequente pode ser completamente inserida na anterior.

**Descrição da Solução**

**Implementação**

A solução foi implementada em Java e dividida em três classes principais: Caixa, Gerenciador e Main.

A classe Caixa representa uma caixa de papelão com dimensões que são ordenadas automaticamente no momento da criação. Ela implementa métodos para comparação entre caixas e verificação de aninhamento. A classe Gerenciador é responsável por ler um arquivo contendo as dimensões das caixas, armazená-las em uma lista, ordená-las e calcular a maior sequência de caixas aninhadas usando programação dinâmica. Por fim, a classe Main contém o método main que inicializa o Gerenciador, lê o arquivo de entrada, ordena as caixas e imprime o comprimento da maior sequência aninhada.

**Estrutura de Dados e Algoritmos**

A classe Caixa armazena e ordena automaticamente suas dimensões. Além disso, implementa um método de comparação para ordenação e um método para verificar se pode conter outra caixa. A classe Gerenciador utiliza BufferedReader para ler o arquivo de entrada linha por linha, cria instâncias de Caixa com base nas dimensões lidas e as armazena em uma lista. Posteriormente, as caixas são ordenadas usando Collections.sort e a programação dinâmica é empregada para calcular o comprimento da maior sequência aninhada.

**Testes Realizados**

Foram realizados testes com diferentes conjuntos de caixas para verificar a corretude e eficiência da solução implementada. Os testes incluíram conjuntos pequenos de caixas para verificar casos simples, bem como conjuntos maiores e variados para garantir que o algoritmo fosse escalável. A verificação manual dos resultados contra os resultados esperados também foi realizada para assegurar a precisão.

**Resultados e Conclusões**

A solução desenvolvida se mostrou eficiente e robusta para encontrar a maior sequência de caixas aninhadas. Os testes realizados confirmaram que o algoritmo funciona corretamente e é capaz de lidar com diferentes tamanhos de entrada de forma adequada. A estrutura de programação dinâmica utilizada proporcionou um desempenho satisfatório, mesmo para conjuntos de dados mais extensos.

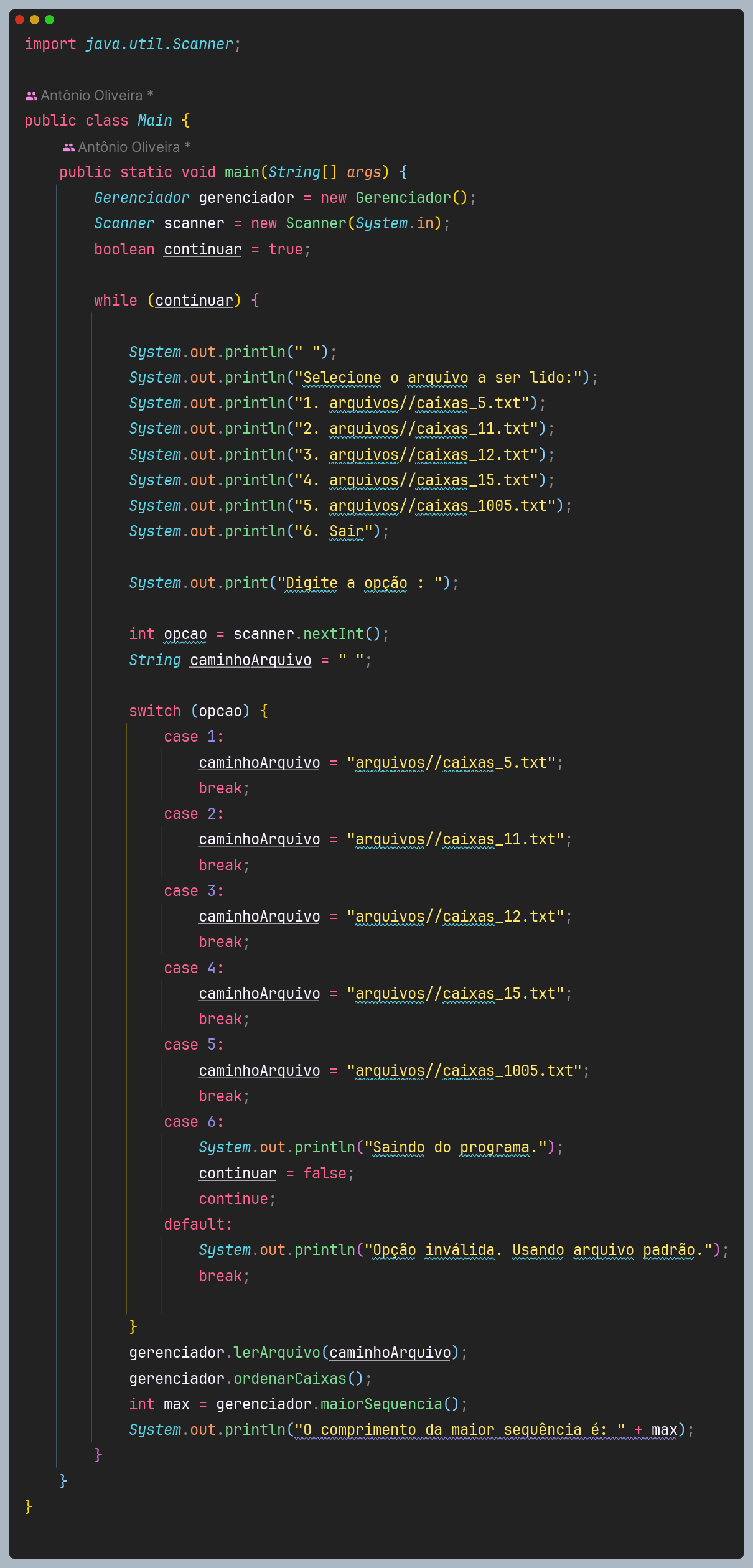
**Recomendações Futuras**

Para melhorar ainda mais a solução, poderiam ser considerados os seguintes pontos:

* Implementar técnicas de otimização para reduzir o tempo de execução em casos extremamente grandes.
* Adicionar validações adicionais para garantir que os dados de entrada estejam corretos e completos.
* Expandir a solução para lidar com formatos diferentes de entrada de dados, como arquivos CSV ou bancos de dados.

**Fotos do Código**

(Adicionar aqui imagens ou capturas de tela do código implementado, destacando as partes mais importantes e explicando brevemente cada uma).



Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

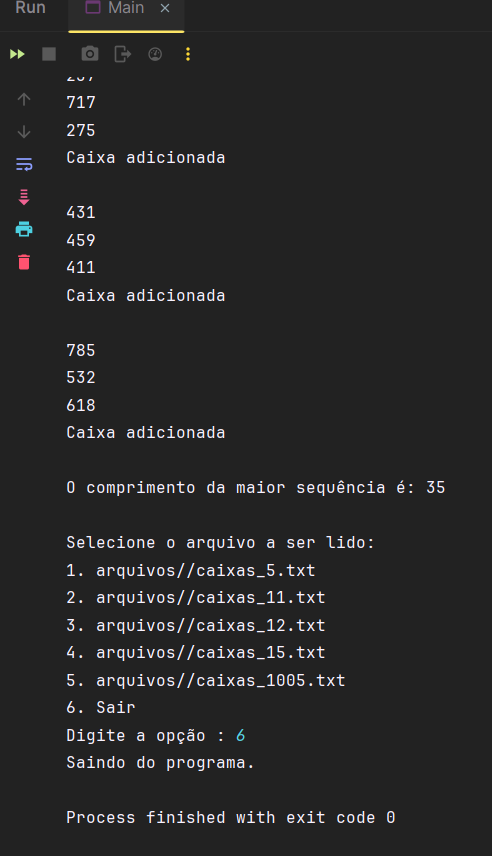
**Testes Realizados**

**Texto

Descrição gerada automaticamente**

**Tela de celular com fundo preto

Descrição gerada automaticamente**

****