DESENVOLVIMENTO DE UMA BIBLIOTECA PARA MANIPULAÇÃO DE IMAGENS PPM, PGM E PBM NA LINGUAGEM JAVA

Lígia Aurora Morais de Oliveira[1]; Rodrigo Curvello[2];

[1] Estudante de Graduação em Bacharelado em Ciência da Computação, IFC - *Campus* Rio do Sul. E-mail: ligiaurora@gmail.com.
[2] Orientador, Professor de Informática, IFC - *Campus* Rio do Sul. E-mail: rodrigo.curvello@ifc.edu.br

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de uma biblioteca (JAR) em Java para a manipulação de imagens PBM (PPM, PGM e PBM). A metodologia adotada incluiu uma pesquisa bibliográfica, com foco na coleta de informações sobre os formatos de imagem, design patterns e princípios de orientação a objetos. Com base nessa fundamentação teórica, foi possível implementar uma lib capaz de realizar diversas operações de manipulação de imagens, como conversão para tons de cinza, eliminação de canais RGB, entre outras. O projeto permitiu a aplicação prática dos conhecimentos adquiridos, reforçando a importância dos conceitos de programação orientada a objetos no desenvolvimento aplicações no âmbito da programação.

Palavras-chave: Imagens PBM; Manipulação de Imagens; Orientação a objetos em Java.

INTRODUÇÃO

Lopes (2018, p.9) afirma que a designação de formato de imagem PBM (*Portable Bitmap*) engloba três formatos de imagem para imagens a preto e branco, em escala de tons cinzentos e a cores, todos eles sem compressão e que apresentam uma estrutura comum.

Dessa forma, o formato PBM engloba três tipos de imagens:

- PBM (*Portable BitMap*) - Imagens preto e branco.
- PGM (*Portable GrayMap*) - Imagens com tom cinza.
- PPM (*Portable PixMap*) - Imagens com cores.

A estrutura dos formatos PBM, PGM e PPM são simples e eficazes para representar dados gráficos. Na variante ASCII[1], os pixels são representados por valores decimais, separados por espaços ou quebras de linha, com os dados organizados linha a linha. Comentários podem ser adicionados com "#", facilitando a inserção de metadados.

Nas variantes binárias, os pixels são armazenados em bytes contíguos. No formato PBM, usado para imagens em preto e branco, 8 pixels são agrupados em um único byte. Já nos formatos PGM e PPM, cada pixel é representado por um byte (tons de cinza) ou três bytes (cores RGB), oferecendo mais detalhes, mas aumentando o tamanho do arquivo.

Com base neste conteúdo teórico, buscou se desenvolver uma biblioteca (JAR) na linguagem Java para manipular as imagens relatadas nos parágrafos anteriores. A manipulação de imagens

[1] É uma forma de armazenar os dados das imagens usando texto legível, pois os valores dos pixels são representados como números decimais e organizados em uma sequência que segue a leitura natural da imagem.

ISSN 2318-2342

nos formatos PPM, PGM e PBM envolve várias operações que transformam as características visuais da imagem. A eliminação de canais RGB, aplicável ao formato PPM, remove um dos canais de cor (vermelho, verde ou azul), resultando em uma imagem que realça os outros dois canais. A interpolação, usada para redimensionar imagens, calcula novos valores de pixel para aumentar ou diminuir o tamanho da imagem, mantendo a qualidade visual.

Converter uma imagem colorida (PPM) para preto e branco envolve calcular a média dos valores RGB para determinar a intensidade de cada pixel. No formato PBM, essa operação transforma a imagem em uma simples representação de preto e branco. Já para converter em tons de cinza, utilizada no formato PGM, a imagem colorida é transformada em diferentes intensidades de cinza, baseando-se nos valores RGB.

A redução de imagem diminui o tamanho da imagem, mantendo uma representação proporcional dos pixels das imagens. A inversão de imagem troca as cores ou intensidades: em imagens PBM, preto se torna branco e vice-versa; em PGM, os tons de cinza são invertidos; e em PPM, as cores RGB são invertidas. Enquanto, que o histograma analisa a distribuição dos tons ou cores em uma imagem, permitindo ajustar o contraste ou identificar padrões visuais, essencial para entender a composição da imagem e melhorar sua qualidade visual.

Para atingir o objetivo de desenvolver essa biblioteca, buscou-se aplicar os conhecimentos teóricos de programação orientada a objetos, alinhado ao desenvolvimento de um diagrama de classe, além disso foi aplicado o design pattern *facade* com o intuito de fornecer uma interface mais simples ao usuário e esconder as operações mais complicada do mesmo e foram aplicados testes unitários utilizando a biblioteca JUNIT como forma de manter a qualidade do projeto e evitar possíveis erros. Por fim, foi desenvolvido a documentação da biblioteca para que o usuário possa entender o funcionamento da biblioteca e aplicar em seus projetos.

PROCEDIMENTOS METODOLÓGICOS

No presente trabalho foi empregado o método de pesquisa bibliográfica, Severino (2007, p.122) afirma que “[...] está é aquela que se realiza a partir de registros disponíveis, decorrentes de pesquisas anteriores. Utiliza-se de dados ou categorias teóricas já trabalhadas por outros pesquisadores. Os textos tornam-se fontes dos temas a serem pesquisados.”

Nesse contexto, foram pesquisadas informações sobre *design patterns* para selecionar as melhores práticas aplicáveis ao projeto, visando otimizar o desenvolvimento da biblioteca. Além disso, realizou-se um estudo sobre as imagens do tipo PBM, com o objetivo de entender suas propriedades e identificar as melhores abordagens para a manipulação desses formatos.

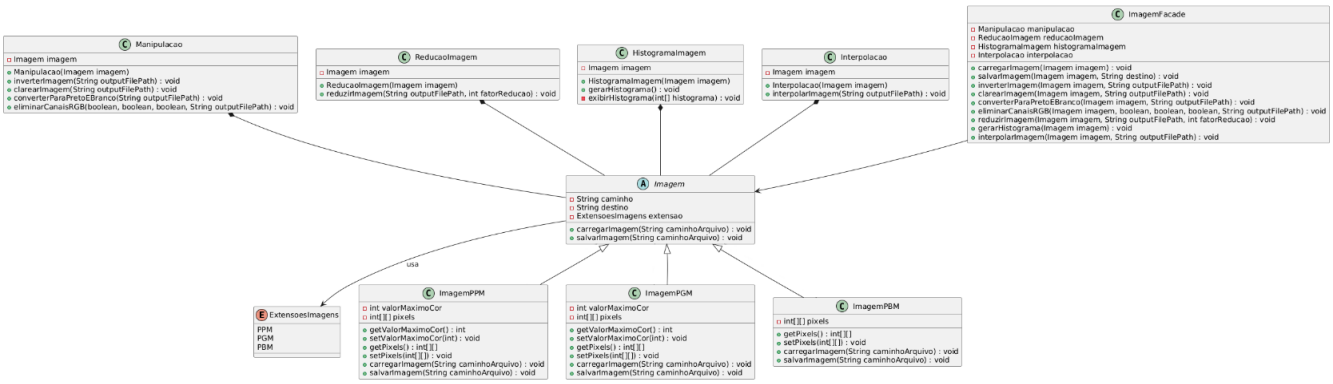
RESULTADOS E DISCUSSÃO

Para o desenvolvimento da biblioteca de manipulação de imagens nos formatos PPM, PGM e PBM, foi utilizada a linguagem Java, com foco na aplicação dos princípios de orientação a objetos. Nesse contexto, foram adotados os conceitos de SOLID, Martin (2019) afirma que [..] os princípios SOLID nos dizem como organizar as funções e estruturas de dados em classes e como essas classes devem ser interconectadas. Além disso, se fez uso do *design pattern* comportamental Facade (Fachada) para simplificar a interface de interação com o sistema, tornando o uso da biblioteca mais intuitivo e eficiente. Para garantir uma visualização clara da estrutura e das

ISSN 2318-2342

interações entre as classes, foi desenvolvido um diagrama de classes utilizando a plataforma PlantUML, o que permite uma visualização mais detalhada do projeto e facilita a compreensão das relações e responsabilidades de cada componente da biblioteca.

Figura 01 – Diagrama de Classes da Biblioteca de Manipulação de imagens



Fonte: Acervo do Autor (2024)

A classe base Imagem desempenha um papel central no design, servindo como o ponto de partida para as subclasses que lidam com formatos de imagem específicos, como ImagemPBM, ImagemPGM e ImagemPPM. Essas subclasses estendem Imagem e implementam métodos para manipular as particularidades de cada formato de imagem, garantindo que a lógica comum seja centralizada e reaproveitada.

A manipulação das imagens é realizada principalmente através da classe Manipulacao, que se encarrega de aplicar diversas transformações, como eliminação de canais RGB, inversão de imagem, e outras operações. Além disso, existem classes especializadas para tarefas específicas, como ReducaoImagem, que se ocupa da redução de tamanho das imagens, aplicando algoritmos de redimensionamento, e Interpolacao, que lida com a interpolação para aumentar ou diminuir a resolução das imagens.

A classe ImagemFacade, que implementa o design pattern Facade. Essa classe fornece uma interface unificada e simplificada para o subsistema de manipulação de imagens, escondendo a complexidade das operações subjacentes das outras classes. Isso torna o sistema mais fácil de usar e mantém a complexidade do código gerenciável.

No exemplo abaixo, código demonstra o uso da classe ImagemFacade para simplificar operações de manipulação de imagens. Na linha 13, é criada uma instância de ImagemPPM, que representa uma imagem no formato PPM, é criada com as especificações do arquivo de imagem. Em seguida, na linha 14, a imagem é carregada a partir de um arquivo usando o método carregarImagem da classe ImagemFacade. Por fim, na linha 15, a imagem é reduzida com o método reduzirImagem, que recebe a imagem, o nome do arquivo de destino e o fator de redução.

Figura 02 – Exemplificação da Classe ImagemFacade

ISSN 2318-2342

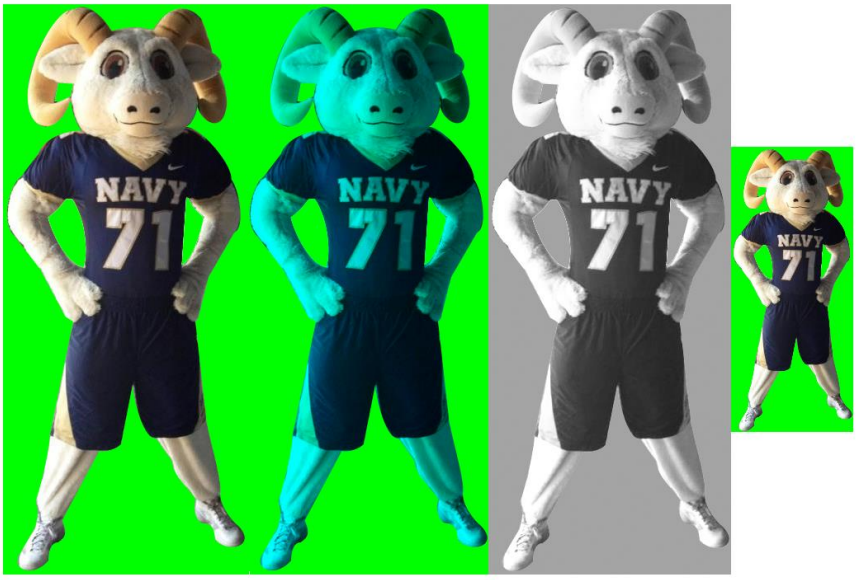
```java
public class TesteIMG {

    public static void main(String[] args) {
        try {

            ImagemFacade imagemFacade = new ImagemFacade();

            ImagemPPM imagemPPM = new ImagemPPM("exemplo.ppm", "", 298, 696, 255);
            imagemFacade.carregarImagem(imagemPPM);
            imagemFacade.reduzirImagem(imagemPPM, "imagem_PPM_reduzida", 2);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Fonte: Acervo do Autor (2024)

O código também reflete um bom uso dos conceitos de orientação a objetos, como encapsulamento, herança e polimorfismo. O encapsulamento é aplicado na forma de como cada classe gerencia suas responsabilidades e esconde os detalhes de implementação, expondo apenas os métodos necessários. A herança permite que código comum seja centralizado na classe base Imagem, enquanto o polimorfismo permite que as subclasses adaptem e se estendam o comportamento conforme necessário, garantindo flexibilidade e reutilização de código.

Para validar funcionalidade da biblioteca, foram realizados testes utilizando imagens nos formatos PPM, PGM e PBM. A imagem PPM utilizada foi obtida a partir do artigo "Manipulação de Imagens PPM e PGM com C" de Fabiano Dicheti (2023), publicado no Medium. A imagem PGM usada nos testes foram geradas a partir dessa imagem PPM, que foi manipulada para conversão em tons de cinza, enquanto que a imagem PPM foi produzida através de um outro código que não está presente nesta biblioteca. Abaixo são apresentados alguns dos resultados obtidos com a utilização da biblioteca, com converter para tons de cinza, retirar o RGB de uma imagem e redução de uma imagem.

Figura 03 – Resultado de alguma das manipulações realizadas utilizando a biblioteca



Fonte: Acervo do Autor (2024)

ISSN 2318-2342

CONSIDERAÇÕES FINAIS

Desenvolver uma biblioteca buscando seguir os princípios de SOLID e Design Patterns evidencia a complexidade que um projeto pode alcançar. À medida que se aprofunda nesses conceitos, torna-se claro que um bom sistema está ligado a uma arquitetura bem planejada. Se um software é difícil de manter, isso geralmente indica problemas no planejamento, que podem resultar em grandes complicações mesmo com pequenas alterações.

Na primeira versão, a biblioteca desenvolvida cumpre os requisitos propostos de manipulação de imagens. No entanto, para uma segunda versão, há diversas melhorias que podem ser implementadas, como por exemplo: novos métodos de manipulação ou novas funcionalidades como a criação de imagens PPM, PGM ou PBM.

Um bom teste para verificar a aplicação correta dos conceitos é justamente durante esse processo de aprimoramento. Embora o projeto não se compare a sistemas mais complexos, ele oferece uma visão clara de como a arquitetura influencia a quantidade de mudanças necessárias ao modificar ou adicionar funcionalidades. O desenvolvimento deste projeto foi fundamental não apenas para aprimorar habilidades em Java, mas também para contribuir positivamente para a formação acadêmica e profissional da autora.

REFERÊNCIAS

DICHETI, Fabiano. **Manipulação de Imagens PPM e PGM com C**. *Medium*, 15 maio 2023. Disponível em: https://medium.com/@fabianodicheti/manipulação-de-imagens-ppm-e-pgm-com-c. Acesso em: 16 ago. 2024.

LOPES JOÃO. **Formatos de Imagem**. Departamento de Engenharia de Informática, 2018.Disponível em: https://disciplinas.ist.utl.pt/~leiccg.daemon/textos/livro/Formatos%20de%20Imagem.pdf. Acessado: 18 ago. 2024.

MARTIN, Robert. **Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software**. Tradução: Samantha Batista, Rio de Janeiro: Alta Books, 2020.

MIRANDA, Paulo. **Princípios de Desenvolvimento de Algoritmos**. EP no. 1. São Paulo: Instituto de Matemática e Estatística (IME), Universidade de São Paulo (USP). Disponível em: http://www.vision.ime.usp.br/~pmiranda/mac122_2s13/EPs/ep1.pdf Acessado:18 ago. 2024.

SEVERINO, Antônio Joaquim. **Metodologia do trabalho científico**. 23. ed. rev. E atual. São Paulo: Cortez, 2007.

ISSN 2318-2342