CONGRESSO CATARINENSE DE
CIÊNCIA DA COMPUTAÇÃO
2022. Ano 9, Vol. 9. ISSN: 2319-0655

Lib IMGFactory – Java Programação Orientada a Objetos 2

Leonardo Felix Scandura
Instituto Federal Catarinense
Campus Rio do Sul
leofelixscandura@gmail.com

Abstract. *This article describes the process of creating the JAVA IMGFactory library. For this, it will be explored which structure was chosen to be used in the creation of the library, presenting the class diagram assembled for the project and from there, presenting the chosen Design Patterns, the library support, the configurations made and the way to execute the methods present in the IMGFactory project.*
Key-words: *JAVA; library; Design Pattern.*

Resumo. *Este artigo descreve o processo da criação da biblioteca JAVA IMGFactory, para isso será explorado qual foi a estruturação escolhida para ser utilizada na criação da biblioteca, apresentando o diagrama de classes montado para o projeto e a partir disso apresentando os Design Patterns escolhidos, a biblioteca de apoio, as configurações feitas e a forma de executar os métodos presentes no projeto IMGFactory.*
Palavras-chave: *JAVA; Biblioteca; Design Pattern.*

1. Introdução

A criação de um projeto em JAVA vai além de desenvolver simples códigos, existe um grande desafio que é proposto quando falamos em manter ou atualizar códigos em projetos que possuem dimensões maiores do que visto em situações de estudo, como ANICHE (2015, p. 2) diz “Pensar em um sistema orientado a objetos é, portanto, mais do que pensar em código. É desenhar cada peca de um quebra-cabeça e pensar em como todas elas se encaixarão juntas.”. Ministrar uma tarefa complicada como essa gera grandes desafios que a solução, muitas vezes, parece incerta.

A arquitetura de software nós ajuda a pensar em soluções e estratégias para nos ajudar no desenvolvimento, além disso a condicionar boas praticas na hora de estruturar o projeto, a arquitetura de software como Martin (2019, p. 43) descreve “O objetivo da arquitetura de software é minimizar os recursos humanos necessários para construir e manter um determinado sistema.”, sempre vão existir possíveis melhorias e se o projeto for pensado para ser fácil de manter e adicionar novas melhorias possuímos uma boa arquitetura.

O projeto da biblioteca IMGFactory teve como idealização utilizar as melhores praticas de desenvolvimento de um código limpo, utilizando os princípios do SOLID e a

adaptação de problemas comuns através de Design Pattern, assim serão explicados as maiores decisões e o motivo delas ao decorrer do artigo.

2. Metodologia

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."

Dentro deste cenário, buscou-se encontrar informações sobre SOLID, DESIGN PATTERNS, JAVA, entre outros. Foram consultados os mais diversos tipos de fontes, como por exemplo: Livros clássicos, matérias em blogs de informática, frameworks, conteúdo que foi ministrado para assim estabelecer uma descrição confiável das informações para o desenvolvimento deste artigo e criação de uma biblioteca JAVA.

3. Desenvolvimento

O objetivo central do trabalho é o desenvolvimento de uma biblioteca ou framework em JAVA que procura ajudar no desenvolvimento de outros projetos, buscando uma estrutura com baixo acoplamento para que seja possível futuras melhorias. Além disso pensando no projeto, o modo de desenvolvimento foi sempre a busca pela melhor estrutura a parir dos conceitos do SOLID, boas praticas de programação e DESIGN PATTERNS.

Outro objetivo também estabelecido, deu-se através do motivo da escolha de realizar uma biblioteca que foca no tratamento de imagem, pois como podemos notar imagens sem tratamento causam lentidões em sites e plataformas e que não são agradáveis para utilizadores e além disso podem ocupar grandes espaços de armazenamento em servidores. Outro ponto a ser comentado é a forma de realizar conversões, redimensionar imagens e alterar as cores, que o projeto da biblioteca busca facilitar para ser realizado através do JAVA, isso possibilita que uma imagem passada para alguma função possa ser tratada de forma fácil.

3.1. Estrutura do Projeto – Design Patterns e Diagrama de Classe.

Por se tratar de uma biblioteca o desafio é a criação de algo fácil de ser utilizado e que consiga realizar a tarefa passada com eficiência. Para isso existem alguns conceitos que facilitam na criação de uma estrutura de projeto eficiente, são esses conceitos: o SOLID e os DESIGN PATTERNS. O SOLID nos dita princípios de organização de projeto, de acordo com Martin (2019, p.126) "Os princípios SOLID nos dizem como organizar as

funções e estruturas de dados em classes e como essas classes devem ser interconectadas." E como o objetivo é a criação de um software que tolere mudanças e melhorias, que seja fácil de entender e possa ser usado em diversos usos para diferentes softwares, torna-se viável a utilização desses conceitos.

A modelagem de um software, representa um grande desafio, pois são diversas variáveis e formas de se fazer, além disso a falta de experiência torna ainda mais desafiador, como Guerra (2014, p.7) diz "O desenvolvedor com menor experiência conhece uma quantidade menor de soluções, o que muitas vezes o leva a adotar um projeto inadequado às necessidades do software.", a maior forma de resolver este problema é pedindo ajuda para programadores mais experientes que já enfrentaram problemas parecidos, e a melhor forma de se fazer isso é através dos DESIGN PATTERNS, que são soluções que funcionam e foram descritas por programadores mais experientes, facilitando assim o desenvolvimento de um software.

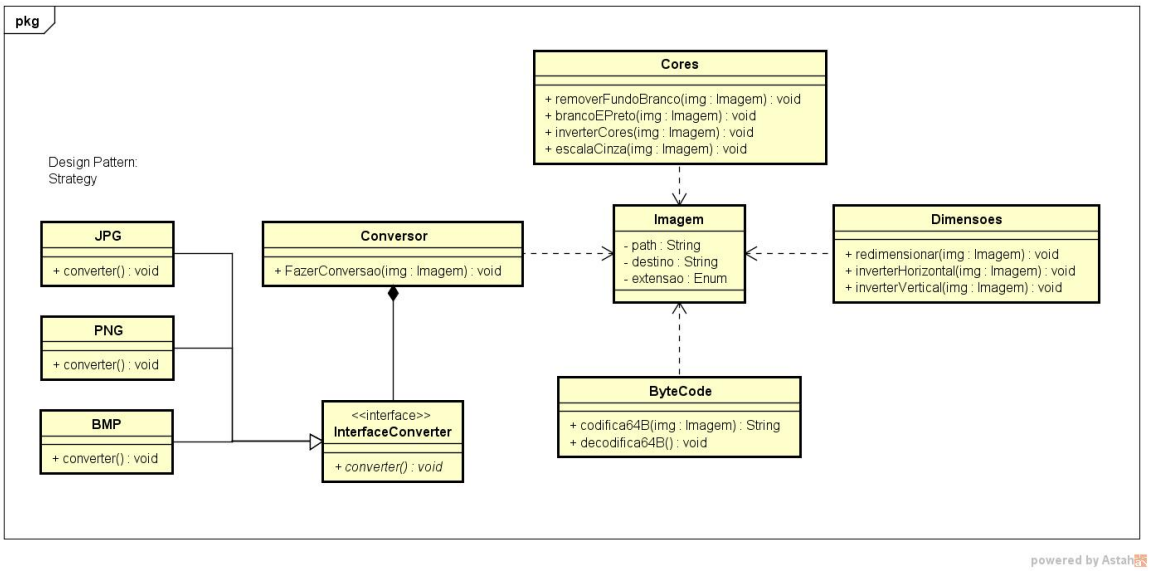

Figura 1: Diagrama UML

Podemos notar que a classe principal da biblioteca é a classe Imagem, e funciona dessa forma pois sem a imagem não é possível realizar as funções que trabalham com a imagem, causando assim uma dependência de todas as ferramentas propostas, existe somente uma função que foge dessa regra que é a de decodificação de BYTECODE.

Quando falamos de imagem é comum pensar no que olhamos, porém uma imagem é mais que isso, pois existe um mecanismo binário por trás de cada foto. Sem entrar muito nos detalhes no funcionamento de uma imagem, podemos utilizar o que chamamos de Base 64 para converter imagens em bytes que possam ser seguros de serem trabalhados sem ser corrompido, e que facilitam a forma de escrita, e que podem acabar gerando alguma vantagem em casos de uso de XML ou em casos de uso de transferência de dados que utilizam apenas a transferência de texto, por isso foi adicionado ao projeto a classe

ByteCode que possui dois métodos o codifica64B e o decodifica64, que fazem, respectivamente, a conversão de uma imagem em um texto base64 a outra faz a conversão da base64 para uma imagem.

3.1.2. Design Pattern Strategy

Para seguir os conceitos propostos pelo SOLID, as funções foram divididas a partir do principio de cada uma, apesar de todas realizarem trabalhos na imagem cada uma faz algo diferente, por exemplo a classe Cores, que realizam somente trabalhos relacionados a cores. Além disso utilizando o DESIGN PATTERN STRATEGY, foi possível evitar que a classe conversor adquira um corpo gigantesco, ao pensar em como o projeto foi construído a conversão pode ser realizada utilizando o STRATEGY por ser um método com algoritmo intercambiável para cada conversão (BMP, JPG e PNG) de acordo com Guerra (2014, p.14) "A solução proposta pelo padrão consiste em delegar a execução do algoritmo para uma instancia que compõe a classe principal".

Ao utilizar Strategy, adquiriu-se a vantagem de adicionar ou modificar os algoritmos de conversão diretamente para cada extensão, não precisando assim alterar a classe principal de conversão, pois basta a classe ter a assinatura da interface (InterfaceConverter) que ela já pode ser considerada um novo tipo de extensão que pode ser utilizado na hora de fazer uma conversão.

3.2. Estrutura do Projeto – Framework de Apoio

O projeto Marvin tem como coordenador o Gabriel Ambrósio Archanjo ex-aluno da universidade estadual de Campinas, o projeto Marvin já teve citações em livros, teses de Mestrado, PHDs e artigos. O Marvin framework conta com um leque de plug-ins que ajudam no tratamento de imagens, indo desde plug-ins para cores até os que alteram as dimensões de uma imagem. Dois grandes destaques do Marvin é o reconhecimento de vídeo, possibilitando detecção de objetos e algumas utilizações uteis para Machine Learning. E o segundo destaque é na parte de representação de fractal a partir de matemática discreta, exemplos que são apresentados no site deles é o sistema de Lindenmayer, conjunto de Mandelbrot e JuliaSet

A utilização dos plug-ins do Marvin possibilitou a construção de algumas classes, como a classe Cores e a classe Dimensões e também ajudou na construção de algumas conversões.

3.3 Estrutura do Projeto – Configuração

A classe principal é a classe Imagem que possui três atributos e são os seguintes:

- Path: Responsável por definir o caminho até a imagem que será trabalhada.

(Exemplo: “/user/exemplo/imagens/imagem1.jpg”)

- Destino: Responsável por definir o caminho da imagem, para salvar as imagens trabalhadas. Ainda está em teste por conter um algoritmo responsável por definir o Destino ao cadastrar um Path.

(Exemplo: “/user/exemplo/imagens/”)

- Extensão: ENUM responsável por definir a extensão da imagem.

(Atualmente existe apenas JPG, PNG e BMP pois a biblioteca só realiza tratamento deste tipo de imagem)

Está classe ainda possui melhorias possíveis, como melhorias futuras a biblioteca pretende não se limitar a somente três extensões e sim a maior quantidade possível, além disso o atributo Destino deve, futuramente, aceitar tanto um destino que será passado quanto o atual funcionamento onde é salvo no mesmo local da imagem sem tratamento.

A conversão de imagem acontece a partir da seguinte execução:

```
1   Imagem batman = new Imagem();
2
3   batman.setPath("/src/junitTestes/IMGTeste/batman.jpg");
4   batman.setExtensao(Extensoes.JPG);
5
6       Conversor.fazerConversao(batman, "batmanPNG", new PNG());
```

Na linha 1 “um” é criado um novo objeto tipo Imagem, e para definir o PATH e a Extensão desta imagem basta usar os métodos SET, na linha 6 “seis” acontece a conversão chamando o método estático fazerConversao da classe Conversor passando o objeto Imagem (no caso o objeto batman), o nome do arquivo que vai ser gerado (no caso batmanPNG) e por fim basta passar a classe que vai compor a instancia da interface. Como existem também o conversor para BMP, bastaria alterar o “new PNG()” para “new BMP()” que aconteceria a conversão.

O funcionamento da biblioteca é simples, e os métodos seguem a mesma forma de funcionamento, deve ser passado um objeto imagem e o nome do arquivo que será salvo a partir da imagem trabalhada, podemos explorar mais uma estrutura que demonstra a utilização da maioria das ferramentas, no caso para passar uma imagem para uma escala de cinza:

```
1   Imagem paisagem = new Imagem();
2   paisagem.setPath("/src/junitTestes/IMGTeste/800x729.jpg");
3
4       //deixa a imagem com escala de cinza
5       Cores.escalaCinza(paisagem, "paisagemCinza");
```

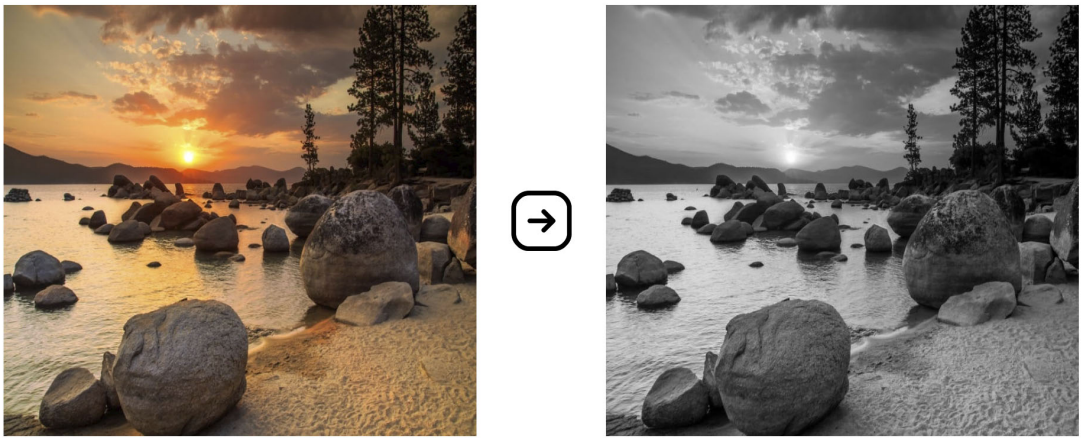

Figura 2 - Exemplo Escala de Cinza

4. Conclusão

Ao criar uma biblioteca utilizando os conceitos de SOLID e Design Patterns, é notório a complexidade que um projeto pode ter, e se aprofundando mais nestes assuntos é possível notar que um bom sistema anda de mãos dadas com uma boa arquitetura, pois se é difícil de manter é por que existe um problema no planejamento do software que sempre acarretará em problemas grandes ao fazer uma pequena alteração.

Para a primeira versão a biblioteca desenvolvida atende os requisitos propostos, e para uma segunda versão existem muitas melhorias a ser feitas, uma boa maneira de descobrir se os conceitos foram bem aplicados é neste processo de melhoria, obvio que nem se compara a sistemas maiores e mais complexos, mas é possível ter uma visão da arquitetura fazendo efeito pela quantidade de mudanças a serem feitas ao alterar ou adicionar algum funcionamento.

5. Referências Bibliográficas

Archanjo, G. A., Andrijauskas, F and Munoz, D. **"Marvin--A Tool for Image Processing Algorithm Development"**, Technical Posters Proceedings of XXI Brazilian Symposium of Computer Graphics and Image Processing, 2008.

SEVERINO, Antônio Joaquim. **Metodologia do Trabalho Científico**. São Paulo: Cortez, 2007.

GUERRA, Eduardo. **Design Patterns com Java: Projeto Orientado a Objetos Guiado por Padrões.** São Paulo: Casa do Código, 2014.

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

ANICHE, Maurício. **Orientação a Objetos e SOLID para Ninjas: Projetando Classes Flexíveis**. São Paulo: Casa do Código, 2015.