

**Universidade Estadual de Maringá (UEM)**  
**Departamento de Informática (DIN)**  
**Curso: Bacharel em Informática**

**Relatório avaliação comparativa**

Integrantes [alunos]	R.A.
Gabriel Valentim De Oliveira Dacie	118419
John William Vicente	118237
Yuri Pires Alves	118792

Disciplina: 9900 - PARADIGMA DE P.I.M.P.E ORIE.A OBJETOS - 2022  
Profª Lilian Passos Scatalon

# **1 - Facilidade de leitura e escrita**

## **1.1 Java**

Sobre a leitura e escrita em Java, ela possui muitas partes semelhantes ou parecidas trazidas da sintaxe de C++ e C, facilitando a leitura de desenvolvedores já habituados com essas duas, porém por sua vez Java também possui uma sintaxe que em alguns pontos pode ser mais complexa e sistêmica exigindo mais processos para a realização de uma tarefa simples. Um outro ponto que pode ser citado que acabou nos afetando bastante enquanto fazíamos a codificação do trabalho foi o fato de que tivemos que criar várias classes para a execução do programa, e isso em um contexto de leitura do algoritmo pode afetar e muito, pois por conta da grande quantidade de classes é fácil se perder e acaba deixando o código com uma grande quantidade de linhas, fazendo com que o programador passe muito tempo revisando o código. Porém, pelo fato do Java ser orientado a objetos e dar mais liberdade para o programador poder realizar suas abstrações também possui seus lados positivos, um exemplo disso é que ela nos proporcionou uma maior facilidade na escrita dos casos de testes, pois, por conta de separarmos cada função do programa em uma classe, caso precisássemos usá-la novamente em algum momento no programa ela estaria de fácil acesso.

Agora destacando um dos pontos positivos que tivemos ao usar o Java na elaboração do trabalho foi que as palavras reservadas e a legibilidade dos métodos presentes nesta linguagem são de fácil entendimento por serem palavras de origem inglesa escritas sem abreviações. Outro ponto que podemos citar, tanto para Java quanto Rust é o fato de ambas terem todos os tipos necessários para a elaboração de programas, assim facilitando tanto a escrita como a leitura, que foi algo que nos ajudou por conta de muitas vezes termos que ficar revisando o trabalho. Como Java possui uma maior quantidade de métodos internos comparado a Rust, tivemos uma facilidade a mais no quesito leitura e escrita, um exemplo é que em Java existe um conjunto rico de métodos que fazem parte da estrutura pilha, enquanto em Rust tivemos que escrever alguns métodos para atender às nossas necessidades.

## **1.2 Rust**

Sobre a leitura e escrita em Rust, ela também teve bastante influência das linguagens de programação C e C++, mas ela foi mais além apresentando uma sintaxe mais limpa e mais bonita que suas predecessoras, porém para iniciantes na linguagem ela tende a ser confusa (como foi o nosso caso), outro ponto é que diferente de Java, ela não exigiu um grande número de classes sendo criado apenas arquivos separados com funções dentro deles onde no momento que precisávamos de alguma dessas funções elas eram chamadas deixando o código com uma maior facilidade na leitura e localização o que também se refletiu diretamente no tempo que ganhamos no momento das verificações do código onde não perderíamos tanto tempo como em Java, porém isso também teve seu lado negativo pois essas funções só poderiam ser chamadas dentro da main (não sabemos se isso é uma verdade absoluta mas por sermos iniciantes em Rust foi a maneira que

encontramos para contornar os problemas), caso existisse uma função que dependeria de outra ela teria que ser reescrita para funcionar o que gerou um acréscimo de linhas desnecessárias algo totalmente diferente de Java que por conta da sua facilidade de reutilizar código nos ajudou nesse critério.

Mesmo sendo citado acima a sintaxe de Rust como um ponto positivo, ela possui seus problemas, um deles são suas palavras reservadas e métodos que foram abreviadas e isso acaba causando certa desordem a programadores que são iniciantes na linguagem como era o nosso caso, acabamos ficando muito perdidos nesse quesito dentro do trabalho, outro ponto seria o fato das variáveis serem imutáveis por padrão, nos fazendo adotar um novo estilo de abordagem para escrita do código tendo a necessidade de fazer várias conversões de tipo, um exemplo seria o fato de existir dois tipos de string em Rust e tendo assim que fazer conversões entre elas, prejudicando assim a legibilidade.

## 2 - Confiabilidade

### 2.1 Java

Java possui um ótimo esquema de confiabilidade, desde a verificação de tipos onde tanto na execução do programa dentro do terminal como na escrita dele dentro da IDE, algo que foi de grande ajuda para nós na criação do algoritmo pois era fácil visualizar onde estávamos errando dentro do código, sem contar as diversas possibilidades de tratamento de exceções que o Java ofereceu para nós, que foi algo de extrema importância dentro do trabalho. Podemos citar também como ponto positivo as mensagens de erro no Java que nos mostrava um grande panorama do erro e nos poupava tempo no momento da resolução deste erro.

### 2.2 Rust

Rust, por se tratar de uma linguagem de mais baixo nível, oferece uma melhor confiabilidade em gestão e segurança tanto de memória quanto de concorrência sendo esse um dos seus pontos mais fortes, por conta de seu modelo de *ownership*. O código roda nativamente no processador, o que torna sua performance melhor. Um outro fator que agrega na confiabilidade de Rust e que nos ajudou bastante durante o desenvolvimento do trabalho foi as mensagens de erros que são muito descritivas sendo ainda mais descritivas que as de Java, que foi algo que nos fez poupar muito tempo no momento da codificação do algoritmo.

## 3 - Custo de execução

### 3.1 Java

Mesmo Java tendo um custo de execução maior que o de Rust por não manipular as variáveis não utilizadas e eliminá-las da memória, ela ainda possui funcionalidades ligadas a manutenção das variáveis criadas, como o *Garbage collection*, que possui um peso no custo de execução, além disso Java acaba tendo um custo maior de execução pois ela é

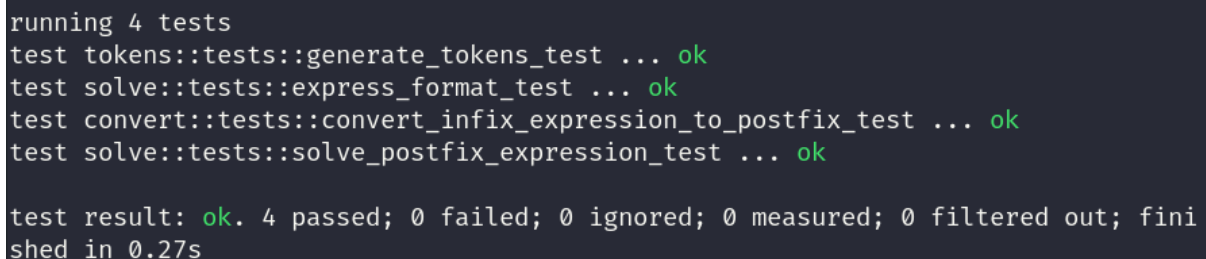
executada em uma máquina virtual, além de manter processos de execução e fazer a comunicação com o sistema operacional.

## 3.2 Rust

Rust acaba sendo mais rápido que Java, pois manipula a memória em tempo de execução, fazendo com que uma variável que não seja mais utilizada seja eliminada automaticamente. Isso permite que durante a execução da aplicação o acesso à memória seja realizado de forma mais rápida e isso fez grande diferença no nosso trabalho principalmente na parte dos testes onde eles eram executados quase que instantaneamente.

## 3.3 Diferenças no tempo de execução entre Java e Rust

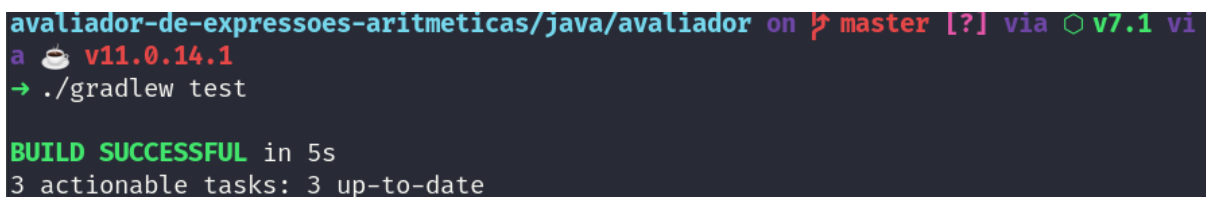
Um exemplo para provarmos nosso ponto citado acima, é um teste que nós mesmos fizemos em ambas as linguagens usando os teste automatizados como base, onde nos testes que fizemos em Rust teve um tempo de execução de 0.27 segundos, enquanto o mesmo teste executado em Java teve um tempo de execução de 5 segundos. Segue abaixo as imagens dos testes:



```
running 4 tests
test tokens::tests::generate_tokens_test ... ok
test solve::tests::express_format_test ... ok
test convert::tests::convert_infix_expression_to_postfix_test ... ok
test solve::tests::solve_postfix_expression_test ... ok

test result: ok. 4 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.27s
```

**Imagem 1** - Tempo de execução dos testes automatizados em Rust.



```
avaliador-de-expressoes-aritmeticas/java/avaliador on master [?] via v7.1 vi
a ☕ v11.0.14.1
→ ./gradlew test

BUILD SUCCESSFUL in 5s
3 actionable tasks: 3 up-to-date
```

**Imagem 2** - Tempo de execução dos testes automatizados em Java.