INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS SÃO JOSÉ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DIEGO BLANCO RODA

GIOVANA BRUNAH REGINA LOSI

PENSAMENTO COMPUTACIONAL E ALGORITMOS
DESAFIO: DECODIFICADOR DE RESISTÊNCIA COM VALIDAÇÃO DE ENTRADA

SÃO JOSÉ
2025

1. Descrição do problema e das regras do desafio

O projeto teve como objetivo o desenvolvimento de um programa em Java capaz de decodificar resistores a partir do código de cores, conforme o padrão eletrônico internacional. O sistema recebeu como entrada as cores das faixas de um resistor e produziu, como saída, o valor correspondente da resistência elétrica.

O programa foi desenvolvido para interpretar resistores com:

- 4 faixas: dois dígitos significativos, multiplicador e tolerância;
- 5 faixas: três dígitos significativos, multiplicador e tolerância; e
- 6 faixas: três dígitos significativos, multiplicador, tolerância e coeficiente de temperatura.

Conforme as regras do desafio:

- a entrada pôde ser fornecida por argumentos de linha de comando ou por entrada padrão (stdin);
- todas as cores informadas foram validadas conforme a posição ocupada no resistor;
- apenas cores válidas foram aceitas em cada faixa; e
- ao identificar a primeira entrada inválida, o programa exibiu a mensagem de erro especificada e encerrou imediatamente a execução;

A saída apresentou:

- o valor inteiro da resistência em Ohms; e
- o valor formatado da resistência, com unidade apropriada.

2 Instruções de execução do programa

O programa foi desenvolvido em Java e executado exclusivamente por meio do terminal integrado do Visual Studio Code, em ambientes Linux e Windows (PowerShell), com a Java Virtual Machine (JVM) previamente instalada e configurada.

2.1 Compilação

A compilação do projeto foi realizada, a partir do diretório raiz, com os seguintes comandos:

Linux:

```
javac src/calculo/*.java src/formatacao/*.java src/Projeto/*.java
```

Windows:

```
javac src\calculo\*.java src\formatacao\*.java src\Projeto\*.java
```

2.2 Execução com argumentos (um resistor)

A execução com argumentos permitiu informar diretamente as cores de um único resistor.

Linux:

```
java -cp src Projeto.DecodificadorResistor amarelo roxo vermelho ouro
```

Windows:

```
java -cp src Projeto.DecodificadorResistor amarelo roxo vermelho ouro
```

2.3 Execução em lote (entrada padrão)

Igualmente foi possível executar o programa em modo de processamento em lote, utilizando a entrada padrão a partir de um arquivo de texto, no qual cada linha representou um resistor.

Linux:

```
java -cp src Projeto.DecodificadorResistor < resistores.txt
```

Windows:

```
Get-Content resistores.txt | java -cp src Projeto.DecodificadorResistor
```

3. Representação algorítmica da solução

O desafio permitia a utilização de pseudocódigo ou fluxograma para representar a solução algorítmica. No aspecto, optou-se pelo pseudocódigo, por possibilitar uma descrição mais clara e direta do fluxo lógico do programa, facilitando a compreensão da solução independentemente da linguagem de programação adotada.

```
INÍCIO

VERIFICAR se existem argumentos de linha de comando

SE existirem argumentos
   processar resistor com os argumentos informados
SENÃO
   LER todas as linhas da entrada padrão
```

```
    PARA cada linha lida
        SE linha não for vazia
            separar cores
            processar resistor
        FIM SE
    FIM PARA
FIM SE

FIM

PROCEDIMENTO processar resistor
    remover espaços e padronizar cores
    verificar se o número de faixas é válido
    verificar se as cores são válidas para cada posição

    calcular os dígitos significativos
    aplicar o multiplicador correspondente
    determinar tolerância e coeficiente de temperatura

    exibir o valor da resistência em Ohms
    exibir o valor formatado com unidade

SE ocorrer erro
    exibir mensagem exigida
    finalizar execução

FIM PROCEDIMENTO
```

4. Comentários e decisões importantes do projeto

O desenvolvimento do projeto exigiu a definição de estratégias técnicas e organizacionais que orientaram a implementação da solução. Nesta seção são descritas as principais decisões tomadas ao longo do trabalho, com ênfase na divisão de responsabilidades, na integração do código e nas práticas adotadas para garantir a robustez do sistema.

4.1 Divisão de responsabilidades na equipe

Desde o início do desenvolvimento, a dupla definiu uma divisão proporcional, clara e complementar de responsabilidades, organizada por camadas do sistema.

O acadêmico Diego ficou responsável pela classe DecodificadorResistor, correspondente à camada de controle, incluindo leitura da entrada, normalização, validações, controle do fluxo principal e tratamento centralizado de erros, enquanto a acadêmica Giovanna restou incumbida das classes CalculoResistencia e FormatadorResistencia, correspondentes às camadas de lógica de negócio e apresentação, abrangendo cálculo elétrico, conversões numéricas, aplicação de

multiplicadores, tolerância, coeficiente de temperatura e formatação rigorosa da saída.

A divisão nestes moldes permitiu desenvolvimento paralelo e equilibrado, evitando sobreposição excessiva e garantindo qualidade técnica em todas as partes do sistema.

4.2 Integração e merge do projeto

Após a conclusão das implementações iniciais sob responsabilidade individual, o projeto passou por um processo estruturado de integração, utilizando o sistema de controle de versão Git. As contribuições foram unificadas por meio de merge na branch principal (main), consolidando o código em uma única base funcional.

A ocasião representou uma etapa crítica do desenvolvimento, porquanto exigiu verificação da compatibilidade entre a lógica de validação, cálculo e formatação, ajustes de interfaces entre métodos das diferentes classes, bem como correção de inconsistências identificadas apenas após a integração completa.

Após o merge, o desenvolvimento passou a ocorrer de forma totalmente colaborativa, com ambos os integrantes atuando sobre a base integrada. Nessa fase, foram realizados:

- ajustes finos de integração entre as camadas do sistema;
- refatorações para melhorar legibilidade e coerência do código;
- correções pontuais identificadas durante testes conjuntos;
- validação do comportamento global do programa.

O processo de integração garantiu que o sistema funcionasse de forma coesa, com todas as partes desenvolvidas separadamente operando corretamente em conjunto.

4.3 Desenvolvimento incremental

O projeto foi desenvolvido seguindo uma abordagem incremental e iterativa, devidamente documentada por meio do histórico de commits no repositório Git.

Inicialmente, o sistema apresentava apenas uma estrutura básica de leitura da entrada, sem validações rigorosas e sem cálculo completo da resistência.

A partir dessa base inicial, o código evoluiu gradualmente, incorporando novas funcionalidades de forma controlada, dentre as quais se destacam:

- normalização da entrada para eliminar variações de escrita;
- validação antecipada do número de faixas do resistor;
- implementação progressiva da validação das cores conforme a posição ocupada;
- separação explícita do fluxo de execução para resistores de 4, 5 e 6 faixas;
- integração incremental das classes de cálculo e formatação;
- ajustes sucessivos para garantir aderência total ao formato exigido pelo desafio.

Cada nova funcionalidade foi introduzida somente após a estabilização da anterior, permitindo a identificação precisa da origem de erros, correções pontuais sem impacto em outras partes do sistema e controle efetivo da evolução do código.

O uso do versionamento possibilitou ainda o registro das decisões técnicas tomadas ao longo do desenvolvimento, evidenciando um processo organizado e metodológico.

4.4 Validação e tratamento de erros

A validação da entrada e o tratamento de erros foram observados como elementos centrais da qualidade do sistema. A estratégia adotada baseou-se na execução de validações em camadas, sempre antes da realização de qualquer cálculo.

A etapa de verificação ocorreu em três níveis principais:

Nível 1 - Validação do número de faixas: o sistema verificou se a quantidade de cores informada correspondia a resistores de 4, 5 ou 6 faixas. Entradas fora desse padrão foram rejeitadas imediatamente;

Nível 2 - Validação das cores conforme a posição: cada faixa do resistor possui restrições específicas quanto às cores permitidas. Para garantir isso, foram definidos conjuntos de cores válidas distintos para:

- dígitos significativos;
- multiplicador;
- tolerância; e
- coeficiente de temperatura.

Essa abordagem impediu combinações fisicamente impossíveis e garantiu conformidade com o padrão eletrônico; e

Nível 3 - Validação integrada ao fluxo principal: somente após todas as validações serem satisfeitas o sistema prosseguiu para o cálculo da resistência.

O tratamento de erros foi padronizado por meio do uso de exceções (IllegalArgumentException). Ao detectar qualquer inconsistência a mensagem de erro especificada no enunciado foi exibida e o programa foi encerrado imediatamente, evitando execuções parciais ou saídas inconsistentes, assegurando previsibilidade, clareza e aderência rigorosa às regras do desafio.

5. Considerações finais

O projeto foi concluído com êxito, atendendo integralmente aos requisitos estabelecidos no desafio proposto. A solução desenvolvida demonstrou não apenas o correto funcionamento do sistema, mas também a aplicação consistente de conceitos fundamentais de programação, organização de código e boas práticas de desenvolvimento de software.

A adoção de uma abordagem incremental e iterativa permitiu que o código evoluísse de forma controlada, com correções pontuais e refatorações realizadas sempre que inconsistências eram identificadas. O processo facilitou a depuração, reduziu riscos de erros acumulados e contribuiu para a estabilidade da versão final entregue.

A divisão equilibrada de responsabilidades entre os integrantes possibilitou o desenvolvimento paralelo e aprofundado de cada camada do sistema, ao mesmo tempo em que o uso de controle de versão com Git viabilizou a integração segura das partes desenvolvidas individualmente. O processo de *merge* e os ajustes colaborativos posteriores evidenciaram a importância da comunicação e da cooperação no desenvolvimento de soluções em equipe.

Outro aspecto relevante foi a ênfase na validação rigorosa da entrada e no tratamento adequado de erros, o que garantiu previsibilidade no comportamento do sistema e aderência estrita às regras do desafio. A validação por camadas e o encerramento imediato em caso de erro contribuíram para a confiabilidade e a robustez do programa.

Por fim, o projeto proporcionou a consolidação de conhecimentos técnicos

relacionados à linguagem Java, ao uso do terminal integrado do VS Code e à aplicação prática de conceitos teóricos, a exemplo da validação de dados, modularização e formatação de saída. A solução final apresentada mostrou-se organizada, consistente e adequada ao contexto proposto.