INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E
TECNOLOGIA DE SANTA CATARINA
CÂMPUS SÃO JOSÉ
DEPARTAMENTO TELECOMUNICAÇÕES

GABRIELE BUENO DOS ANJOS
LUCAS PACHECO KEMPFER
NICOLAS CARDOSO DA SILVA

RELATÓRIO DO PROJETO BATALHA NAVAL

GRUPO B

SÃO JOSÉ
2025

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem como objetivo apresentar o desafio e a resolução do projeto final da disciplina de Pensamento Computacional e Algoritmos, referente à primeira fase do curso de Análise e Desenvolvimento de Sistemas (ADS), desenvolvido durante o segundo semestre de 2025.

O projeto final tem como função consolidar os conhecimentos adquiridos ao longo da disciplina por meio da solução de um desafio, que envolve análise do problema, modelagem algorítmica, implementação em Java e comunicação técnica, a fim de desenvolver o engajamento, a autonomia, o pensamento crítico e a colaboração entre os estudantes.

2 DESAFIO

Desafio 3 — Geração e Validação de Tabuleiros de Batalha Naval.

O objeto do projeto foi o desenvolvimento de um programa em Java capaz de operar em dois modos, Geração e Validação, de um tabuleiro de batalha naval.

Geração: o programa deve receber uma entrada “G” e ser capaz de gerar um tabuleiro de batalha naval, assim como corretamente posicionar as peças de embarcações, sob as seguintes regras:

- Gerar um tabuleiro de dimensões 10 × 10;
- Posicionar aleatoriamente um único navio de cada tipo, na horizontal ou vertical, sem sobreposições e dentro das margens do tabuleiro;
- Imprimir o tabuleiro resultante, utilizando o símbolo correspondente do navio para casas ocupadas e “ . “ para casas vazias, separados por espaços.

Validação: o programa deve receber uma entrada “V”, e um tabuleiro via redirecionamento de entrada, e ser capaz de ler o tabuleiro, verificar sua validade, segundo as regras, e retornar uma mensagem de validação. Um tabuleiro é considerado válido quando:

- Possui exatamente 10 linhas, cada uma contendo 10 elementos separados por espaço;
- Contém apenas símbolos válidos de navio, ocupando a quantidade correta de casas, nas posições corretas de casas, horizontal ou verticalmente;
- Não há sobreposição de navios;

Tabela 1: Navios utilizados no tabuleiro.

Navio	Tamanho	Símbolo
Porta-aviões	5	P
Encouraçado	4	E
Cruzador	3	C
Submarino	3	S
Contratorpedeiro	2	N

3 EXECUÇÃO DO PROGRAMA

O programa é executado no modo Geração de Tabuleiro ao chamar “java batalhaNaval G” na linha de comando.

Para executar o modo de validação é necessário executar “java batalhaNaval V” complementado pelo símbolo de redirecionamento “<” seguido de uma entrada a ser validada.

4 REPRESENTAÇÃO ALGORÍTMICA EM PSEUDOCÓDIGO

4.1 Método estático inicializarTabuleiro

```
static void inicializarTabuleiroVazio ()
        para cada linha = 0 e menor que 10,
        para cada coluna = 0 e menor que 10,
                preencha " ." para a posição linha/coluna do tabuleiro
                some 1 em coluna, refaça
        some 1 em linha, refaça
```

4.2 Método estático barcoCabe

static boolean barcoCabe (recebe: se é vertical ou horizontal, número de linha, número de coluna, tamanho do barco)

Se for vertical, então
Para uma linha aleatória, a altura da linha tem espaço necessário pro barco no tabuleiro?
barco cabe true or false

Senão,
Para uma coluna aleatória, a posição da coluna tem espaço necessário pro barco no tabuleiro?
barco cabe true or false

4.3 Método estático tabuleiroVago

static boolean tabuleiroVago (recebe: se é vertical ou horizontal, número de linha, número de coluna, tamanho do barco)

Se for vertical, então
Para i = 0, enquanto i for menor que o tamanho do barco,
a posição de linha e coluna que eu quero colocar o barco é diferente de " . " ?
Se for, tabuleiroVago é falso
Mas se essa posição estiver livre, confira a debaixo

Senão, sendo horizontal
Para i = 0, enquanto i for menor que o tamanho do barco,
a posição de linha e coluna que eu quero colocar o barco é diferente de " . " ?
Se for, tabuleiroVago é falso
Mas se essa posição estiver livre, confira a da direita

4.4 Método estático colocarBarco

static void colocarBarco (recebe: tamanho do barco, símbolo do barco)

Não temos nenhum barco colocado.

Enquanto o barco não estiver colocado, continue tentando uma posição.

Para uma linha aleatória, e uma coluna aleatória, sendo horizontal ou vertical, se o barco couber e também a posição estiver livre, pode preencher as peças.

Para i = 0, enquanto i for menor que o tamanho do barco,
Se for vertical,
Atualize a posição do tabuleiro e às peças abaixo com o símbolo pelo tamanho do barco correspondente
Se for horizontal,
Atualize a posição do tabuleiro e às peças a direita com o símbolo pelo tamanho do barco correspondente

Após, considere o barco colocado.

4.5 Método estático imprimirTabuleiro

static void imprimirTabuleiro()
    para cada linha = 0 e menor que 10,
    para cada coluna = 0 e menor que 10,
        imprima a posição linha/coluna do tabuleiro
        some 1 em coluna, refaça
    some 1 em linha, refaça
    pule uma linha

4.6 Método estático lerTabuleiro

static boolean lerTabuleiro ( )

Criar um leitor de entrada (Scanner).

Para linha = 0, enquanto linha for menor que o tamanho do tabuleiro:

    Se não existir uma próxima linha de entrada:
        Exibir mensagem "INVÁLIDO (dimensões incorretas)"
        Retornar falso

  Ler uma linha de texto.
  Separar a linha em partes usando espaço como delimitador.

  Se a quantidade de partes for diferente do tamanho do tabuleiro:
    Exibir mensagem "INVÁLIDO (dimensões incorretas)"
    Retornar falso

  Para coluna = 0, enquanto a coluna for menor que o tamanho do tabuleiro:
    Armazenar a parte correspondente na posição linha/coluna do tabuleiro

Todas as linhas estando corretas, retornar verdadeiro

4.7 Método estático validarSimbolos

static boolean validarSimbolos ( )

Para linha = 0, enquanto linhal for menor que o tamanho do tabuleiro:
Para coluna = 0, enquanto coluna for menor que o tamanho do tabuleiro:

Obter o símbolo na posição linha/coluna do tabuleiro

Se o símbolo não for um dos seguintes: "P", "E", "C", "S", "N", ".":
Exibir mensagem "INVÁLIDO (navio desconhecido)"
Retornar falso

Todas os símbolos estando corretos, retornar verdadeiro.

4.8 Método estático dfs

static void dfs (recebe: matriz de posições, matriz de visitados, linha inicial, coluna inicial)

Criar dois arranjos auxiliares para deslocamento:
um para variação de linhas {-1, +1, 0, 0}
um para variação de colunas {0, 0, -1, +1}

Criar uma pilha para armazenar posições a serem visitadas

Inserir na pilha a posição inicial (linha inicial, coluna inicial)
Marcar a posição inicial como visitada na matriz de visitados

Enquanto a pilha não estiver vazia:

Remover uma posição da pilha
Obter a linha atual e a coluna atual dessa posição

Para cada uma das quatro direções possíveis:

Calcular a nova linha somando o deslocamento
Calcular a nova coluna somando o deslocamento

Verificar se a nova posição está dentro dos limites do tabuleiro

Se a posição estiver dentro do tabuleiro:

Se existir navio nessa posição E ela ainda não foi visitada:
Marcar a posição como visitada
Inserir a nova posição na pilha

4.9 Método estático validarFormatoNavio

static boolean validarFormatoNavio (recebe: símbolo do navio, tamanho esperado)

Criar uma matriz booleana 10x10 para marcar as posições onde o símbolo aparece
Inicializar um contador de ocorrências com valor zero

Inicializar variáveis para:
menor linha, maior linha, menor coluna e maior coluna
(com valores iniciais fora do intervalo do tabuleiro)

Para linha = 0, enquanto linha for menor que 10:
Para coluna = 0, enquanto coluna for menor que 10:
Se a posição do tabuleiro contiver o símbolo do navio:
Marcar a posição correspondente como verdadeira na matriz de posições
Incrementar o contador de ocorrências
Atualizar menor e maior linha, se necessário
Atualizar menor e maior coluna, se necessário

// 1) navio faltando
Se o contador de ocorrências for igual a zero:
Exibir mensagem "INVÁLIDO (navio faltando)"
Retornar falso

Determinar se o navio está em uma única linha
Determinar se o navio está em uma única coluna

// 2) navio na diagonal
Para linha = 0, enquanto linha for menor que 9:
Para coluna = 0, enquanto coluna for menor que 9:

Se existir símbolo do navio na posição atual
E existir símbolo do navio na posição diagonal inferior direita
E não existir símbolo nas posições adjacentes horizontal e vertical:
Exibir mensagem "INVÁLIDO (navio na diagonal)"
Retornar falso

Se existir símbolo do navio na posição inferior
E existir símbolo do navio na posição diagonal direita

```
        E não existir símbolo nas posições adjacentes correspondentes:
            Exibir mensagem "INVÁLIDO (navio na diagonal)"
            Retornar falso

// 3) conectividade do navio
Criar uma matriz booleana de visitados
Inicializar contador de componentes com valor zero

Para linha = 0, enquanto linha for menor que 10:
    Para coluna = 0, enquanto coluna for menor que 10:
        Se a posição contiver o símbolo do navio
        E ainda não tiver sido visitada:
            Incrementar o número de componentes
            Executar o método dfs a partir dessa posição

Se existir mais de um componente conectado:
    Inicializar indicadores de separação por água e por outro navio

    Se o navio estiver na mesma linha:
        Percorrer as colunas entre a menor e a maior coluna do navio
        Se encontrar uma posição diferente do símbolo:
            Se a posição contiver ".", marcar separação por água
            Senão, marcar separação por outro navio

    Senão, estando na mesma coluna:
        Percorrer as linhas entre a menor e a maior linha do navio
        Se encontrar uma posição diferente do símbolo:
            Se a posição contiver ".", marcar separação por água
            Senão, marcar separação por outro navio

    Se a separação for causada por outro navio:
        Exibir mensagem "INVÁLIDO (sobreposição de navios)"
        Retornar falso
    Senão:
        Exibir mensagem "INVÁLIDO (múltiplos navios do mesmo tipo)"
        Retornar falso

// 4) continuidade do navio (sem buracos)
Se o navio estiver em uma única linha:
    Percorrer todas as colunas entre a menor e a maior coluna
    Se alguma posição não contiver o símbolo:
        Se a posição contiver ".", exibir "INVÁLIDO (múltiplos navios do mesmo tipo)"
        Senão, exibir "INVÁLIDO (sobreposição de navios)"
        Retornar falso
```

Senão, estando em uma única coluna:
    Percorrer todas as linhas entre a menor e a maior linha
    Se alguma posição não contiver o símbolo:
        Se a posição contiver ".", exibir "INVÁLIDO (múltiplos navios do mesmo tipo)"
        Senão, exibir "INVÁLIDO (sobreposição de navios)"
        Retornar falso

// 5) tamanho do navio
Se o número de ocorrências for diferente do tamanho esperado:
    Exibir mensagem "INVÁLIDO (tamanho incorreto de navio)"
    Retornar falso

Se todas as validações forem satisfeitas:
    Retornar verdadeiro

4.10 Método estático validarNavios

static boolean validarNavios ( )

Valida o navio do tipo "P", que deve ocupar 5 posições no tabuleiro
Se a validação falhar:
    Retornar falso

Valida o navio do tipo "E", que deve ocupar 4 posições no tabuleiro
Se a validação falhar:
    Retornar falso

Valida o navio do tipo "C", que deve ocupar 3 posições no tabuleiro
Se a validação falhar:
    Retornar falso

Valida o navio do tipo "S", que deve ocupar 3 posições no tabuleiro
Se a validação falhar:
    Retornar falso

Valida o navio do tipo "N", que deve ocupar 2 posições no tabuleiro
Se a validação falhar:
    Retornar falso

Se todos os navios forem validados corretamente:
    Retorna verdadeiro

4.11 Método main

public static void main(String[] args) {

Se não houver argumentos
imprima a instrução de uso
Fim do programa

Se o primeiro argumento for “G”
chama o método que inicializa o tabuleiro vazio,
e enquanto o número de barcos colocados for menor que o número total de barcos, chama o método que coloca o barco no tabuleiro
chama o método que imprime o tabuleiro

Se o primeiro argumento for “V”
chama o método que inicializa o tabuleiro vazio
chama o método que lê o tabuleiro
se a leitura do tabuleiro falhar
fim do programa
chama o método que valida os símbolos do tabuleiro
se a validação dos símbolos falhar
fim do programa
chama o método que valida todos os navios do tabuleiro
se a validação dos navios falhar
fim do programa
imprima “VÁLIDO”

Senão, o primeiro argumento for diferente de “G” ou “V”, imprima a instrução de uso.
}

5 CONSIDERAÇÕES

Uma das principais decisões adotadas foi a separação do código em diferentes **métodos estáticos**, priorizando a clareza e organização ao longo do código.

Outro ponto foi a criação das **variáveis globais** da classe, onde “int tamanhoTabuleiro”, “String vazio”, “int[ ] tamanhoBarco” e “String[ ] simbolos” são do tipo **FINAL**, pois constituem valores fixos e inalteráveis para a correta implementação da proposta.

Por outro lado, o arranjo “String[ ][ ] tabuleiro” foi definido como estático da classe, para poder ser atualizado pelos métodos da classe. O uso de uma **matriz bidimensional de strings**, um arranjo de arranjo, foi necessário para poder mapear e acessar as coordenadas linhas/colunas de forma eficaz.

Para a geração automática do tabuleiro, optou-se por utilizar a classe Random, permitindo que a posição e a orientação dos navios fossem escolhidas de forma aleatória. Antes da inserção de cada navio, o sistema **realiza duas verificações**: se o navio cabe dentro dos limites do tabuleiro e se as posições escolhidas estão livres. Essa decisão evita sobreposição de embarcações e garante a validade estrutural do tabuleiro gerado.

O método responsável pela colocação dos navios utiliza um **laço while**, que continua tentando gerar posições aleatórias até que uma posição válida seja encontrada. Essa escolha assegura que todos os navios sejam posicionados corretamente, independentemente da aleatoriedade, sem interromper a execução do programa.

O método lerTabuleiro é responsável pela **leitura da entrada padrão**, utilizando o redirecionamento de entrada conforme especificado no enunciado. Esse método realiza a validação das dimensões do tabuleiro analisando sua conformidade.

O método validarSimbolos percorre todas as posições do tabuleiro e verifica se cada elemento pertence ao conjunto de **símbolos permitidos** (P, E, C, S, N ou .). Essa validação isolada garante que navios desconhecidos ou caracteres inválidos

sejam detectados antes de qualquer análise estrutural mais complexa, respeitando o princípio de validação incremental.

O método validarFormatoNavio é responsável por fazer a **validação de um navio específico**, verificando sua existência, orientação (horizontal ou vertical), conectividade, ausência de diagonais, continuidade e tamanho correto. Enquanto o método dfs realiza uma busca a partir de uma posição inicial para marcar todas as casas conectadas de um mesmo navio. Esse método é utilizado para identificar componentes conectados e **detectar separações indevidas**. Por último, o método validarNavios coordena a **validação de todos os navios** do tabuleiro, chamando validarFormatoNavio para cada tipo exigido. O tabuleiro só é considerado válido se todos os navios atenderem às regras.

No método main, foi adotado o uso de **argumentos de linha de comando** para definir o modo de execução do programa. O modo **G** é responsável pela geração automática do tabuleiro, enquanto o modo **V** foi planejado para a validação de um tabuleiro fornecido externamente por meio de um arquivo texto, conforme solicitado no desafio.