UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO ACADEMICO DE EXATAS E NATURAIS – CCEN
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO - DSC
SISTEMAS DA INFORMAÇÃO

AIMÊ NAÍSA DE SOUZA

BATALHA NAVAL - DOCUMENTAÇÃO

BLUMENAU
2025

SUMÁRIO

1 DESCRIÇÃO DAS FUNCIONALIDADES

O programa implementa uma versão simplificada do jogo Batalha Naval (8x8) utilizando apenas os conteúdos permitidos pela disciplina de Introdução a Programação do Professor Ricardo Voigt:

- **Matrizes bidimensionais**
- **Laços while**
- **Condicionais if e switch/case**
- **Classe Random para sorteio das posições dos navios**
- **Entrada de dados com Scanner**
- **Nenhum atributo de classe** (todas as variáveis estão dentro de métodos)
- **Somente uma classe principal**
- **Tabuleiro visível separado do tabuleiro real**

1.1 O que o programa faz:

- Cria dois tabuleiros 8×8:
  - **tabuleiroVisivel** → mostrado ao jogador
  - **tabuleiroNavios** → contém os navios escondidos

- Preenche ambos com água (~).
- Posiciona **10 navios com tamanhos diferentes**, de forma aleatória, sem encostarem uns nos outros:
  - 1. Porta-aviões (4 células)
  - 2. Cruzadores (3 células)
  - 3. Destroyers (2 células)
  - 4. Submarinos (1 célula)

- O jogador tem **30 tentativas** para encontrar todas as partes dos navios.
- Após cada jogada, o jogo exibe:
  - Número da tentativa
  - Quantidade de acertos
  - Taxa de acerto (%)
  - Feedback (ACERTOU / ERROU)
  - Aviso de navio afundado

- O jogo termina quando:
  - Todas as células dos navios forem destruídas (**vitória**)
  - Ou quando as 30 tentativas acabarem (**derrota**)

- Ao final, o programa exibe:
    - Estatísticas completas
    - Quantidade de navios afundados por tipo
    - Tabuleiro final revelado

2 LÓGICAS DE POSICIONAMENTO DOS NAVIOS

O posicionamento dos navios é realizado utilizando exclusivamente **matrizes**, **laços while** e a classe **Random**, conforme os conteúdos permitidos pela disciplina.
O processo garante que todos os navios sejam posicionados de forma **válida**, **aleatória** e **sem encostar uns nos outros**.

2.1 Escolha da Posição Inicial

Para cada navio, o programa sorteia três informações fundamentais:

1. **Orientação do navio**
    a. `0` = horizontal
    b. `1` = vertical

2. **Linha inicial** (valor entre 0 e 7)

3. **Coluna inicial** (valor entre 0 e 7)

Com base nesses valores e no tamanho do navio, o programa calcula a posição da última célula que ele ocuparia. Esse cálculo determina se será possível posicionar o navio sem ultrapassar os limites do tabuleiro.

2.2 Verificação se o Navio cabe no Tabuleiro

Antes de tentar posicionar o navio, o programa verifica se ele permanece totalmente dentro do mapa.

- **Navio Horizintal:** colunaFinal = colunaIncial + tamanhoNavio - 1
    - Sendo a sua posição sendo válida apenas se: **colunaFinal < tamanho.**
- **Navio Vertical:** linhaFinal = linhaInicial + tamanhoNavio - 1
    - Sendo a sua posição sendo válida apenas se: **linhaFinal < tamanho**.

Se o navio ultrapassar os limites do Tabuleiro, uma nova posição aleatória será sorteadas, realizando então, novas validações e verificações.

2.3 Verificação de Área Livre (“Regra dos navios não podem se encostar”)

Além de encaixar-se dentro do tabuleiro, é necessário que o navio respeite a regra de **não encostar em nenhum outro navio, nem mesmo pela diagonal.** Para que isso ocorra, o programa executa os seguintes procedimentos para cada célula que o navio ocuparia.

1. Define uma área ao redor da célula, representada por um quadrado entre:

    - Linha – 1 até linha + 1.
    - Coluna – 1 até coluna +1.

2. Ajusta os limites para evitar índices fora da matriz:

    - Exemplo: Linha = 0, então linha –1 vira 0.

3. Percorre toda essa área usando dois laços **while.**

4. Se qualquer posição nessa área não contiver água **(~)**, o local é considerado inválido.

Isso impede totalmente que os navios fiquem:

- Encostados pela Lateral;
- Encostados por Cima ou por Baixo;
- Encostados pela Diagonal.
-

2.4 Resultado da Verificação

Se todas as células da matriz e suas áreas vizinhas estiverem livres, são realizadas as seguintes validações e ações:

- O posicionamento é aprovado;
- O navio é colocado no tabuleiro;
- Suas células recebem caractere correspondente **(P, C, D ou S);**
- O programa avança para o próximo navio.

Se qualquer verificação falhar:

- A posição é rejeitada;
- Uma nova tentativa completamente aleatória é realizada;

Esse processo de tentativa e validação ocorre continuamente até que todos os navios sejam posicionados de forma válida e sem encostar

3 DETECÇÕES DE NAVIOS AFUNDADOS

A detecção de navios afundados é realizada por meio de um controle simples e eficiente que acompanha quantas partes de cada navio ainda restam no tabuleiro. Para isso, o programa utiliza um vetor auxiliar chamado **celulasRestantes[]**, que armazena a quantidade de células que compõem cada navio.

3.1 Estrutura de Controle

O vetor de células restantes possui as seguintes características:

- Seu tamanho é igual ao número total de navios dos jogos **(10 navios);**
- Cada índice do vetor corresponde a um navio específico;
- Cada posição armazena inicialmente o tamanho do respectivo navio:
  - 4, 3, 3, 2, 2, 2, 1, 1, 1, 1;
- Conforme o jogador acerta as partes do navio, esses valores são subtraídos.

Esse vetor torna possível saber **exatamente** quando o navio foi completamende destruídos pelo usuário (jogador).

3.1 Funcionamento da Detecção de Afundamento

Quando o jogador escolhe uma coordenada para atacar e acerta uma parte do navio, o programa realiza três etapas essenciais, sendo elas:

1. **Identificação de qual navio foi atingido:** Cada célula é ocupada por um navio que possui um ID correspondente, que é armazenado na matriz **tabuleriroIds[linha][coluna].**
2. **Redução da quantidade de células restantes:** Após identificar o ID, o programa reduz o contado daquele navio. Como cada navio tem seu próprio ídice dentro do vetor, isso funciona mesmo com navios repetidos.
3. **Verificação se o navio foi totalmente atingido:** Depois de reduzir o contador, o programa verifica se as células restantes baseada no ID é igual a 0. Quando esse valor chega a zero, significa que **todas as partes do navio foram atingidas ao longo do jogo**, sendo assim, retornando que o navio foi atingido.

3.3 Feedback do Jogador

Assim que o navio é completamente destruído pelo o usuario (jogador), o programa imprime automaticamente a seguinte mensagem:

- AFUNDOU! Você destruiu um **<tipo de navio>!**

O tipo de navio é obtido a partir do vetor **tipoNavios[],** que contém as seguintes informações:

- P, C, C, D, D, D, S, S, S, S

Essa lógica permite que o **seja identificado automaticamente o tipo de navio, exibindo ao usuário a mensagem correta, e que funcione para navios de qualquer tamanho, sem condições especiais.**

4 CÁLCULO DA PONTUAÇÃO

Embora o enunciado do trabalho não exija um sistema de pontuação numérica completa, ele solicita que o programa apresente informações estatísticas ao final da partida — incluindo taxa de acertos, erros e situação dos navios. Nesta implementação, o programa exibe todas as estatísticas essenciais para avaliar o desempenho do jogador, seguindo rigidamente as especificações.

4.1 Cálculo de Taxa de Acertos

A taxa de acertos é calculada a cada jogada, refletindo o desempenho que usuário (jogador) possue ao longo da partida. A fórmula utilizada e aplicada é:

- Taxa = (acertos * 100.0) / (acertos + erros)

Caso tratados:

- Se o usuário ainda não realizou nenhuma jogada, a taxa é definida como 0%, evitando divisão por 0.

Onde é exibida:

- Durante cada tentativa (a cada jogada).
- Novamente nas Estatísticas Finais ao término do jogo,

A Taxa aparece com uma casa decimal, facilitando a leitura do desempenho do usuário.

4.2 Estatísticas Finais da Partida

Ao final do jogo – seja por vitória ou por esgotamento das 30 tentativas pré-definidas – o programa apresenta um conjunto completo de estatísticas, permitindo analisar o resultado da partida realizada.

As informações exibidas são:

- **Tentativas Utilizadas:** Quantidade de jogadas feitas até o encerramento da partida.
- **Acertos Totais:** Total de Células de navios atingidos.
- **Erros Totais:** Total de tiros que atingiram a água.
- **Taxa Final de Acerto:** Calculado com a mesma fórmula da seção anterior.

- **Quantidade de Navios Afundados:** Indica quantos dos 10 navios foram totalmente destruídos.
- **Quantidade de Navios Afundados por tipo:** Usando a função de resuma, o programa exibe:
    - **Porta-aviões**
    - **Cruzadores**
    - **Destroyers**
    - **Submarinos**

CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto permitiu aplicar, de forma prática e integrada, todos os conteúdos estudados nas unidades iniciais da disciplina de Programação. O sistema implementado atende plenamente a todos os requisitos exigidos no enunciado, utilizando exclusivamente recursos básicos da linguagem Java.

Todas as funcionalidades foram construídas dentro das seguintes limitações propostas: o programa utiliza apenas estruturas fundamentais — como matrizes, laços while, condicionais switch/case, a classe Random e a entrada de dados via Scanner — e foi implementado inteiramente em uma única classe, sem o uso de atributos globais. Além disso, o tabuleiro foi corretamente representado por uma matriz 8×8, conforme solicitado. O posicionamento dos navios foi realizado de maneira totalmente aleatória a cada execução do jogo, garantindo variedade entre as partidas. A lógica implementada assegura que os navios não se encostem, nem lateral, vertical ou diagonalmente, garantindo fidelidade às regras tradicionais da Batalha Naval. O programa também fornece feedback claro ao usuário durante todas as jogadas, seguindo o formato indicado pelo professor no enunciado.

Dessa forma, o trabalho cumpre integralmente todos os requisitos técnicos e funcionais propostos, demonstrando domínio dos conceitos iniciais de lógica de programação, manipulação de matrizes e controle de fluxo em Java.

REFERÊNCIAS

VOIGT, Ricardo. *Disciplina IP 2025/2 B*. Repositório GitHub. Disponível em: https://github.com/ricardovoigt/disciplina_IP_2025_2_B. Acesso em: 26 nov. 2025.