Universidade Regional de Blumenau – FURB

Centro de Ciências Exatas e Naturais – CCEN

Departamento de Sistemas e Computação – DSC

Disciplina: Introdução à Programação

Trabalho Final da Disciplina

Implementação do Jogo Batalha Naval

Objetivo

Desenvolver uma aplicação em Java que simule o jogo Batalha Naval, jogado via terminal. A aplicação deverá permitir que o jogador ataque coordenadas no tabuleiro e receba feedback sobre os ataques, incluindo se acertou ou errou navios, se afundou navios, e fornecer estatísticas ao final do jogo.

Descrição

Você deverá implementar o jogo Batalha Naval em um tabuleiro 8x8 que contém **navios de diferentes tamanhos** posicionados aleatoriamente. O jogador poderá atacar posições no tabuleiro até que todos os navios sejam destruídos ou tenha completado 30 tentativas. O jogo deve ser jogado via terminal e deve fornecer feedback detalhado ao jogador após cada ataque, além de apresentar estatísticas ao final da partida.

Requisitos

1. Estrutura do Projeto

- Utilize a linguagem Java somente com o conteúdo aprendido em sala de aula (unidade 1 até 6, matrizes e classe Random). Os únicos import devem ser: `import java.util.Scanner;` e `import java.util.Random;`
- Não pode usar atributos;
- Todo o código deve ser implementado em uma única classe.
- O código deve ser bem documentado e seguir boas práticas de programação.

2. Inicialização do Tabuleiro

- Crie um tabuleiro 8x8 representado por uma matriz bidimensional de caracteres.
- Preencha o tabuleiro com água ('~') inicialmente.

Figura 1 - Exemplo de inicialização

```
  0 1 2 3 4 5 6 7
0 ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~
```

```
2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
```

3. Posicionamento dos Navios (MELHORADO)

- **Posicione aleatoriamente os seguintes navios no tabuleiro:**
  - 1 Porta-aviões (tamanho 4) - representado por 'P'
  - 2 Cruzadores (tamanho 3) - representado por 'C'
  - 3 Destroyers (tamanho 2) - representado por 'D'
  - 4 Submarinos (tamanho 1) - representado por 'S'
- Os navios devem ser posicionados horizontal ou verticalmente (escolha aleatória).
- Certifique-se de que os navios não se sobreponham.
- A posição dos navios não deve ser revelada para o jogador.
- **Total de células ocupadas: 4 + (2×3) + (3×2) + (4×1) = 20 células**

4. Interação com o Jogador

- Peça ao jogador para inserir coordenadas de ataque (linha e coluna).
- Valide as coordenadas inseridas (devem estar dentro dos limites do tabuleiro).

Figura 2 - Exemplo de jogada com feedback

```
Tentativa 5/30 | Acertos: 3 | Taxa: 60.0%

  0 1 2 3 4 5 6 7
0 ~ ~ X ~ ~ ~ ~ ~
1 ~ A ~ ~ X ~ ~ ~
2 ~ A ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~

Digite a linha (0-7): 3
Digite a coluna (0-7): 1
ACERTOU! Um navio foi atingido!
AFUNDOU! Você destruiu um Destroyer!
```

5. Feedback dos Ataques (MELHORADO)

- Informe ao jogador se o ataque acertou um navio ou errou.
- Atualize o tabuleiro com 'A' para acertos e 'X' para erros.
- **Informe quando um navio for completamente afundado** (todas as suas células foram atingidas).
- Mostre o tipo de navio afundado (Porta-aviões, Cruzador, Destroyer ou Submarino).
- Mostre o tabuleiro ao jogador após cada ataque, mas sem revelar as posições dos navios restantes.
- Mostre ao jogador se é uma casa que ele já atacou.
- Mostre ao jogador se ele jogou em uma posição inválida.
- **Exiba estatísticas a cada rodada:**
  - Número da tentativa atual
  - Total de acertos
  - Taxa de acerto (percentual)
  - Navios afundados

6. Condição de Parada

- O jogo termina quando todos os navios forem destruídos (jogador vence) ou o jogador tenha completado um total de 30 jogadas sem encontrar todos os navios (jogador perde).
- Informe ao jogador se ele venceu ou foi derrotado e revele todas as posições dos navios.

7. Estatísticas Finais (NOVO)

Ao final do jogo, apresente um resumo com:

- Total de tentativas utilizadas
- Total de acertos
- Taxa de acerto (percentual)
- Navios afundados vs total de navios
- **Pontuação final calculada como:**
  - `Pontos = (Acertos × 10) + (Navios_Afundados × 50) - (Erros × 2)`
  - Bônus de 100 pontos se vencer usando menos de 25 tentativas
- Classificação da performance:
  - Excelente: > 400 pontos
  - Bom: 300-400 pontos
  - Regular: 200-299 pontos
  - Precisa melhorar: < 200 pontos

Figura 3 - Exemplo de estatísticas finais

```
========================================
         ESTATÍSTICAS FINAIS
========================================
Status: VITÓRIA!
Tentativas usadas: 23/30
Total de acertos: 20
Total de erros: 3
Taxa de acerto: 86.96%

Navios afundados: 10/10
```

```
- Porta-aviões: 1/1
- Cruzadores: 2/2
- Destroyers: 3/3
- Submarinos: 4/4

PONTUAÇÃO FINAL: 794 pontos
- Acertos: 20 × 10 = 200
- Navios afundados: 10 × 50 = 500
- Penalidade erros: 3 × -2 = -6
- Bônus vitória rápida: +100

Classificação: EXCELENTE!
========================================
```

Entregáveis

1. Código Fonte

- Todo o código fonte do projeto no arquivo Java, junto com a documentação.
- O código deve ser entregue via AVA.

2. Documento

- Descrição das funcionalidades implementadas.
- Explicação detalhada da lógica de:
  - Posicionamento de navios multi-células
  - Detecção de navios afundados
  - Cálculo de pontuação

3. Imagens

- Capturas de telas demonstrando a funcionalidade do jogo. As imagens devem ser enviadas todas juntas em um documento .PDF com a descrição de cada uma delas.
- **Inclua capturas de:**
  - Tela inicial do jogo
  - Jogadas com acertos
  - Mensagem de navio afundado
  - Tela de estatísticas finais (vitória)
  - Tela de estatísticas finais (derrota)
  - Tabuleiro final revelado

Critérios de Avaliação

- **Correção (30%):** o programa deve funcionar corretamente e seguir as regras do jogo.
- **Qualidade do Código (25%):** o código deve ser bem estruturado, legível e seguir boas práticas de programação.
- **Funcionalidades (25%):**
  - Navios multi-células implementados corretamente
  - Sistema de detecção de navios afundados funcionando

    - Cálculo de pontuação correto
- **Interação com o Usuário (10%):** a interação deve ser intuitiva, com mensagens de erro adequadas quando necessário.
- **Documentação (10%):** A descrição das funcionalidades deve ser completa e clara.

Prazos

- **Entrega do Código Fonte e Documentação:** 02/12/25
- **Apresentação do Projeto – no máximo:** 03/12/25

Atenção: Ao apresentar o trabalho, o código fonte e documentação devem estar postados no AVA em "TrabalhoFinal_entrega"

Instruções Adicionais

- Trabalhe individualmente
- Qualquer dúvida ou problema deve ser comunicada ao professor o mais breve possível.

Dicas de Implementação

Posicionamento de Navios Multi-Células:

1. Escolha aleatoriamente uma posição inicial e uma orientação (horizontal/vertical)
2. Verifique se o navio cabe no tabuleiro na direção escolhida
3. Verifique se todas as posições estão livres (sem sobreposição)
4. Se válido, posicione o navio; senão, tente outra posição

Detecção de Navio Afundado:

1. Mantenha contadores para cada navio (células restantes)
2. Quando um acerto ocorrer, identifique qual navio foi atingido
3. Decremente o contador daquele navio
4. Quando contador chegar a zero, o navio afundou

Estrutura de Dados Sugerida:

- Matriz do tabuleiro (o que o jogador vê)
- Matriz de navios (posição real dos navios - oculta)
- Vetores para armazenar informações de cada navio:
    - Tamanho original
    - Células restantes
    - Tipo do navio

Bons códigos!