Acadêmicos: Luis Gustavo Demetrio; Paulo Henrique Demetrio.

Trabalho Final

Implementação do Jogo Batalha Naval

Blumenau, 03 de junho de 2025.

Funcionalidades implementadas:

1.

```
private BatalhaNaval(){
    Scanner scan = new Scanner(System.in);
    Random random = new Random();
    char[][] tabuleiroExibe = new char[8][8];
    char[][] tabuleiroRegistra = new char[8][8];
    int[] acertosEtentativas = new int[2];
    int acertos = 0, tentativas = 30;
    populaTabuleiro(tabuleiroExibe);
    populaTabuleiro(tabuleiroRegistra);
    registrarNavios(tabuleiroRegistra, random);
    jogar(tentativas, acertos, tabuleiroExibe, tabuleiroRegistra, acertosEtentativas, scan);
    scan.close();
}
```

Com a classe “BatalhaNaval()”, o computador faz a declaração das variáveis a serem utilizadas, contendo o Scanner para leitura das informações, Random para geração de valores aleatórios, tabuleiroExibe como um tabuleiro que será mostrado ao usuário, tabuleiroRegistra que terá a informação dos navios espalhados no quadrante, além da contagem de acertos e tentativas do usuário.

Após, busca pelas funções que preencherão o jogo. Popular o tabuleiro, passando como parâmetro ambos os tabuleiros mencionados; Registrar os navios, passando somente o tabuleiroRegistra com a random.

Com o cenário pronto, é buscada a função jogar, passando como parâmetro as informações que serão utilizadas.

2.

```
private static char[][] populaTabuleiro(char tabuleiro[][]) {
    for (int i = 0; i < tabuleiro.length; i++) {
        for (int j = 0; j < tabuleiro.length; j++) {
            tabuleiro[i][j] = '~';
        }
    }
    return tabuleiro;
}
```

Com a classe “populaTabuleiro()”, é feito um laço de repetição duas vezes do tamanho do tabuleiro (8x8), preenchendo-o com o caractere referente à representação de água (~). O laço é feito duas vezes, por ser necessário preencher tanto as linhas quanto as colunas. O primeiro laço com índice “i” percorre pelas linhas, enquanto o laço de índice “j” percorre pelas colunas.

Após executar os laços, retorna com o tabuleiro já preenchido com “água”.

3.

```
private static char[][] registrarNavios(char tabuleiro[][], Random random) {
    int navios = 1;
    while (navios < 11) {
        int linha = random.nextInt(bound:8);
        int coluna = random.nextInt(bound:8);
        if (tabuleiro[linha][coluna] != 'N') {
            tabuleiro[linha][coluna] = 'N';
            navios++;
        }
    }
    return tabuleiro;
}
```

Com a classe “registrarNavios()” que recebe por parâmetro o tabuleiroRegistra e a random, é incializada uma variável tipo inteiro “navios” que começa com valor 1. Após, executa um laço “while” enquanto esse valor for menor que 11.

Dentro do laço, é gerado um valor aleatório pela função da random tanto para linha quanto coluna de númeiro inteiro até 8 e verificado se, dentro do tabuleiro, essas coordenadas possuem valor diferente de ‘N’. Em caso afirmativo, é preenchido por ‘N’, registrando assim um “navio” na coordenada. Após, incrementa a variável “navios”. Por fim, retorna o tabuleiroRegistra com os navios registrados em posições aleatórias.

4.

```
private static void jogar(int tentativas, int acertos, char[][] tabuleiroExibe, char[][] tabuleiroRegistra,
        int acertosEtentativas[], Scanner scan) {
    menu(tentativas);
    do {
        if (tentativas == 0 && acertos < 10) {
            System.out.println(x:"Suas tentativas acabaram e ainda há navios não afundados. Você perdeu!");
            System.out.println("Navios encontrados: " + acertos + "/10");
            System.out.println(x:"Posições dos navios:");
            mostrarTabuleiro(tabuleiroRegistra);
            break;

        } else if (acertos == 10) {
            System.out.println(x:"**********");
            System.out.println(x:"Parabéns! Você afundou todos os navios e venceu!");
            System.out.println("Navios encontrados: " + acertos + "/10");
            mostrarTabuleiro(tabuleiroRegistra);
            break;

        } else {
            System.out.println(x:"---------------------------------------");
            System.out.println("Você ainda possui " + tentativas + " tentativas.");
            System.out.println("Navios encontrados: " + acertos + "/10");
            mostrarTabuleiro(tabuleiroExibe);
            acertosEtentativas = atacar(tabuleiroRegistra, tabuleiroExibe, scan, tentativas, acertos);
            tentativas = acertosEtentativas[0];
            acertos = acertosEtentativas[1];
        }

    } while (tentativas >= 0 && acertos <= 10);
    System.out.println();
    System.out.println(x:"Fim do jogo!");
}
```

A classe “jogar()” recebe os dados por parâmetros e primeiramente, chama a classe menu. Após, faz validação pelo laço “do – while”, repetindo enquanto a expressão dentro do while for verdadeira.

É feito duas validações pelo “if”, podendo encerrar o jogo caso a primeira validação for correta, resultando na derrota do usuário por não ter encontrado todos os navios e as tentativas terem terminado, mostrando o tabuleiro e as posições dos navios. Pela segunda validação, declara a vitória por ter encontrado todos os navios e mostra o tabuleiro com as posições. Por fim, dá continuidade ao jogo informando quantidade de tentativas restantes e navios encontrados até então. Mostra o tabuleiro ainda com água e chama a função de “atacar()” para que o usuário ataque a próxima coordenada.

Ao finalizar o laço de repetição, é declarado “Fim do jogo!”.

5.

```
private static void menu(int tentativas) {
    System.out.println(x:"*********************************");
    System.out.println(x:"Bem-vindo ao jogo Batalha Naval!");
    System.out.println(x:"Foram distribuídos 10 navios em um tabuleiro 8x8.");
    System.out.println("Você tem " + tentativas + " tiros para encontrá-los.");
    System.out.println(x:"Legenda: '~' = Água, A = Navio Afundado, X = Tiro na água");
}
```

Com a função “menu()”, é impresso na tela a apresentação do jogo com alguns dados relevantes sobre o funcionamento, como quantidade de navios, tamanho do tabuleiro, tentativas e legenda.

6.

```
private static void mostrarTabuleiro(char tabuleiro[][]) {
    System.out.println(x:"\n  0 1 2 3 4 5 6 7");
    for (int i = 0; i < tabuleiro.length; i++) {
        System.out.print(i + " ");
        for (int j = 0; j < tabuleiro[i].length; j++) {
            System.out.print(tabuleiro[i][j] + " ");
        }
        System.out.println();
    }
}
```

Com a função “mostrarTabuleiro()”, será impresso primeiramente uma linha contendo as posições das colunas do tabuleiro. Após, é feito laço de repetição com o “for” para imprimir o número referente à linha.

Dentro desse laço, é impresso com o segundo laço “for” o conteúdo que existe dentro do vetor tabuleiroExibe. Inicialmente, será impresso todos os conteúdos como “~”.

7.

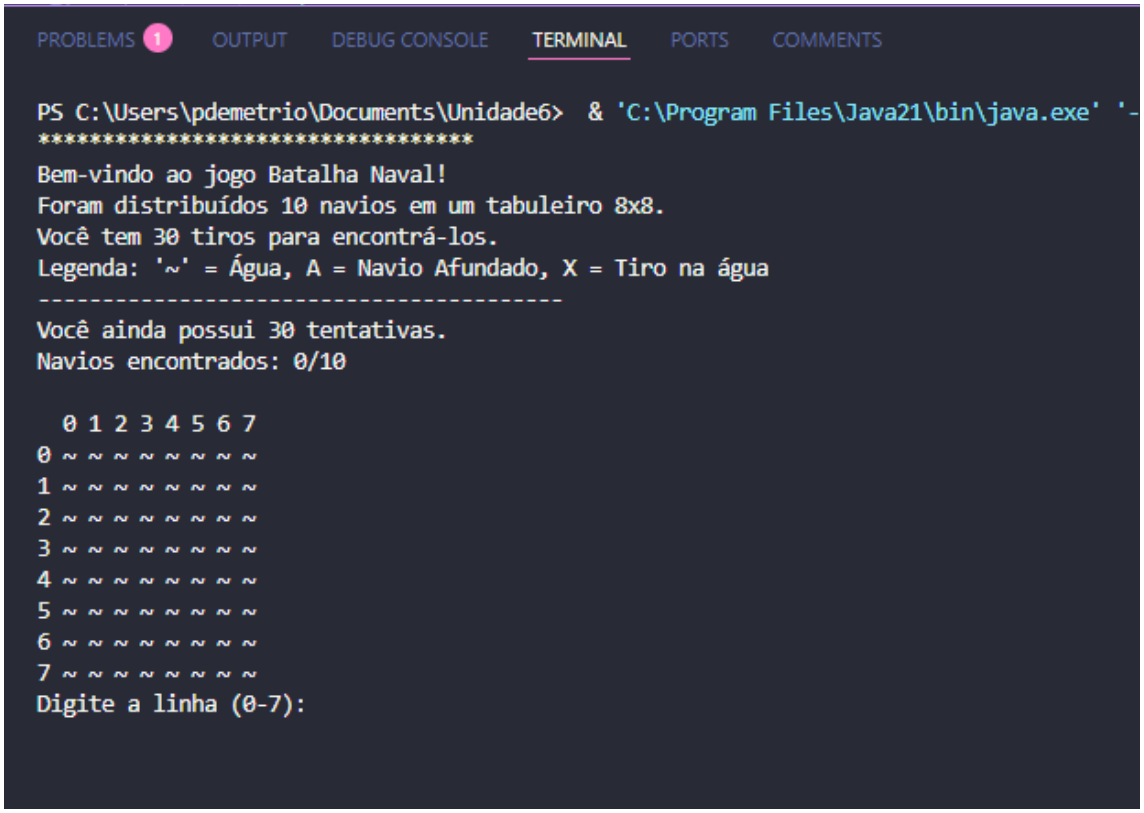
```java
private static int[] atacar(char[][] tabuleiroRegistra, char[][] tabuleiroExibe, Scanner scan, int tentativas,
        int acertos) {
    System.out.print(s:"Digite a linha (0-7): ");
    int linhaAtaque = scan.nextInt();
    System.out.print(s:"Digite a coluna (0-7): ");
    int colunaAtaque = scan.nextInt();

    if (linhaAtaque < 0 || linhaAtaque >= 8 || colunaAtaque < 0 || colunaAtaque >= 8) {
        System.out.println(x:"-----------------");
        System.out.println(x:"Posição inválida!");
    } else if (tabuleiroExibe[linhaAtaque][colunaAtaque] == 'A' || tabuleiroExibe[linhaAtaque][colunaAtaque] == 'X') {
        System.out.println(x:"----------------------------");
        System.out.println(x:"Você já atirou nesta posição!");
    } else if (tabuleiroRegistra[linhaAtaque][colunaAtaque] == 'N') {
        System.out.println(x:"----------------------");
        System.out.println(x:"Você afundou um navio!");
        tabuleiroExibe[linhaAtaque][colunaAtaque] = 'A';
        acertos++;
    } else {
        System.out.println(x:"-------------");
        System.out.println(x:"Tiro na água!");
        tabuleiroExibe[linhaAtaque][colunaAtaque] = 'X';
    }
    tentativas--;
    return new int[] {tentativas, acertos};
}
```

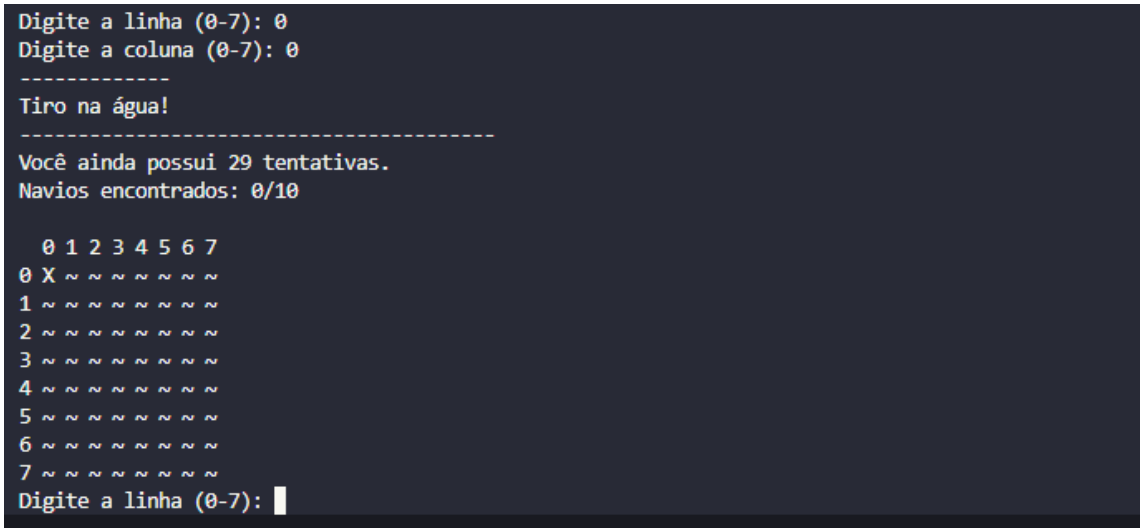
Com a função “atacar()”, é solicitado ao usuário inicialmente que insira linha e coluna que deseja atacar no tabuleiro. Após, validado pela primeira condição do “if” se a linha e coluna estão no quadrante do tabuleiro (8x8). Se não estiver, informa o usuário que a posição é inválida. Pela segunda condição, é validado se já existe algo na linha como “A” ou “X”, demonstrando que a posição já foi alvo anteriormente e informando o usuário a respeito. Pela terceira condição, é validado se na linha existe um “N” pelo tabuleiroRegistra, pois caso houver, é avisado ao usuário que atingiu um navio, atualizado o registro da coordenada para “A” e acrescentado à variável “acertos”. Caso não esteja em nenhuma das condições, será avisado que não acertou nenhum navio e o tiro foi na água, atualizando o tabuleiroExibe com um “X” na coordenada indicada.

Após as validações, é decrementado da variável tentativas e retornado.

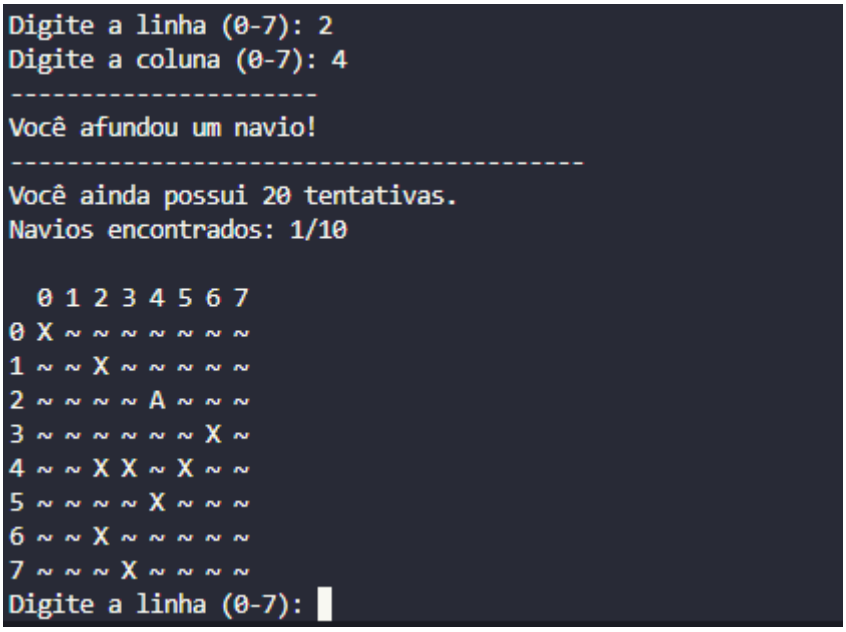
Inicializando o jogo:



Usuário ataca a coordenada 0, 0 e recebe um retorno sobre a ação feita. Após, é solicitado que ataque novamente:



Após acertar um navio, é mostrado ao usuário, atualizado no tabuleiro e informado quantidade de tentativas restantes e navios afundados:



Atacando uma coordenada atacada anteriormente, feita tratativa para avisar ao usuário. Após todas as tentativas utilizadas e sem atingir os 10 navios, informado que o usuário perdeu e mostrada as posições dos navios: