INSTITUTO FEDERAL DE MINAS GERAIS

CAMPUS OURO BRANCO

TÉCNICO INTEGRADO EM INFORMÁTICA

NOME DOS AUTORES

- Lucas Rodrigues Moreira
- Gabriel Guimarães Barbosa
- Aysha

TÍTULO

Documentação Do Trabalho prático 5 - Batalha Naval com GUI

Trabalho apresentado na disciplina de programação II do curso técnico de informática do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais.

Professor: Saulo Henrique Cabral Silva

Ouro Branco

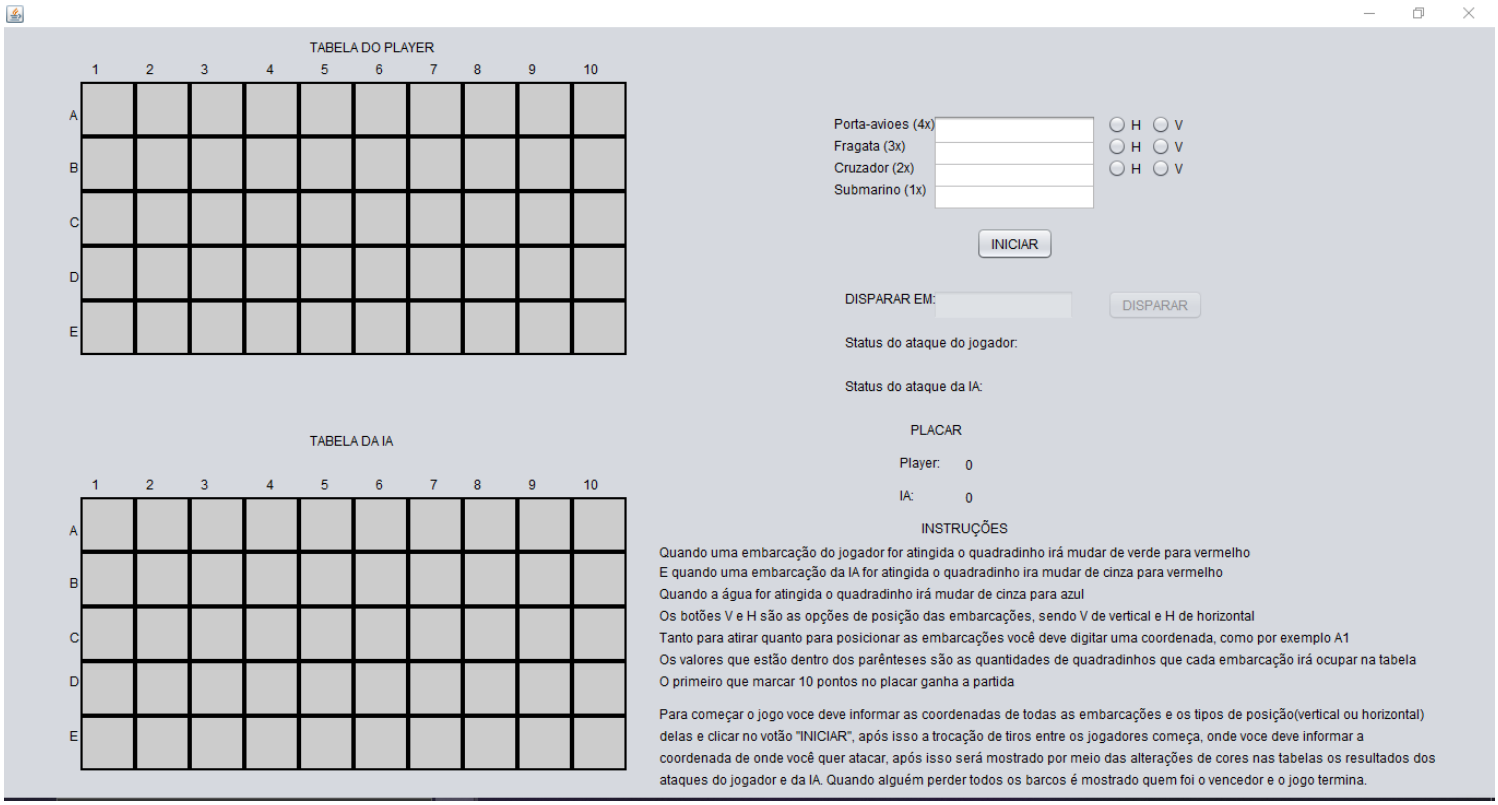
Outubro de 2023

INTRODUÇÃO

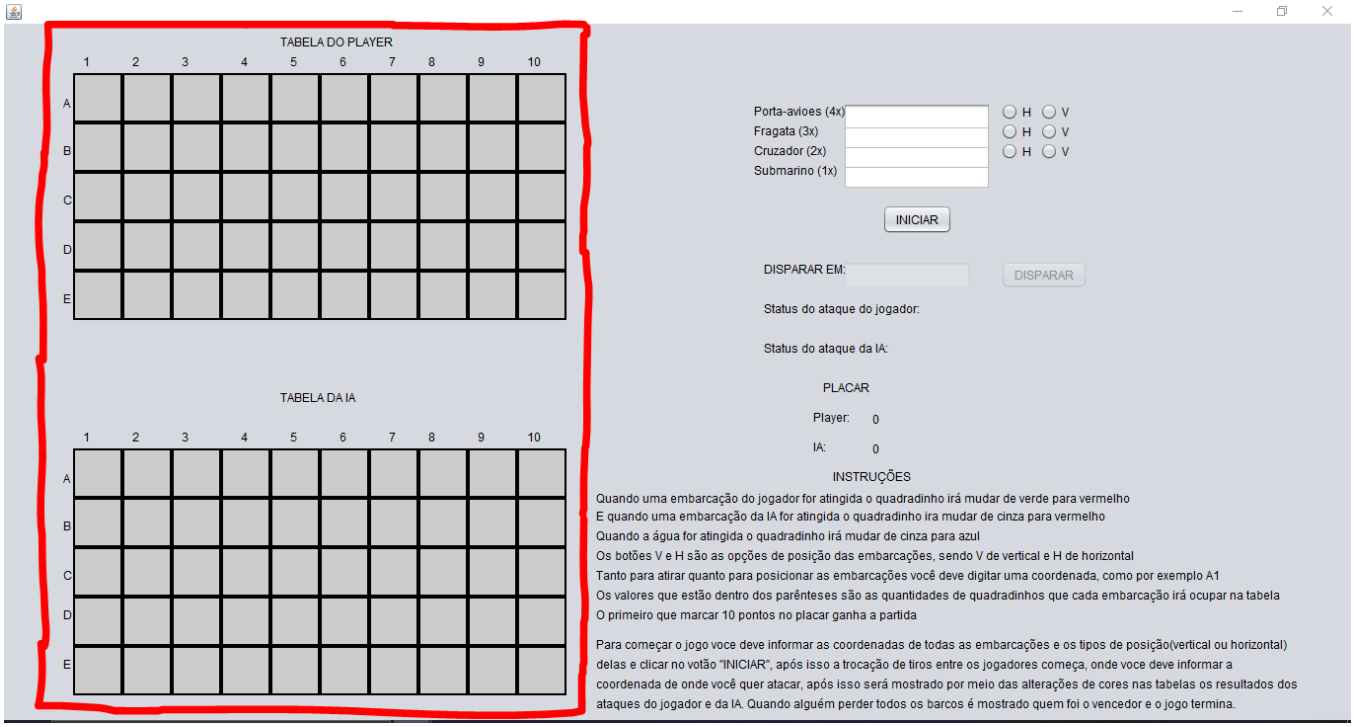
Descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa:

O programa tem a função de simular, em forma de um jogo interativo, simular um jogo de Batalha Naval entre um jogador e uma IA.

Tela do Jogo:

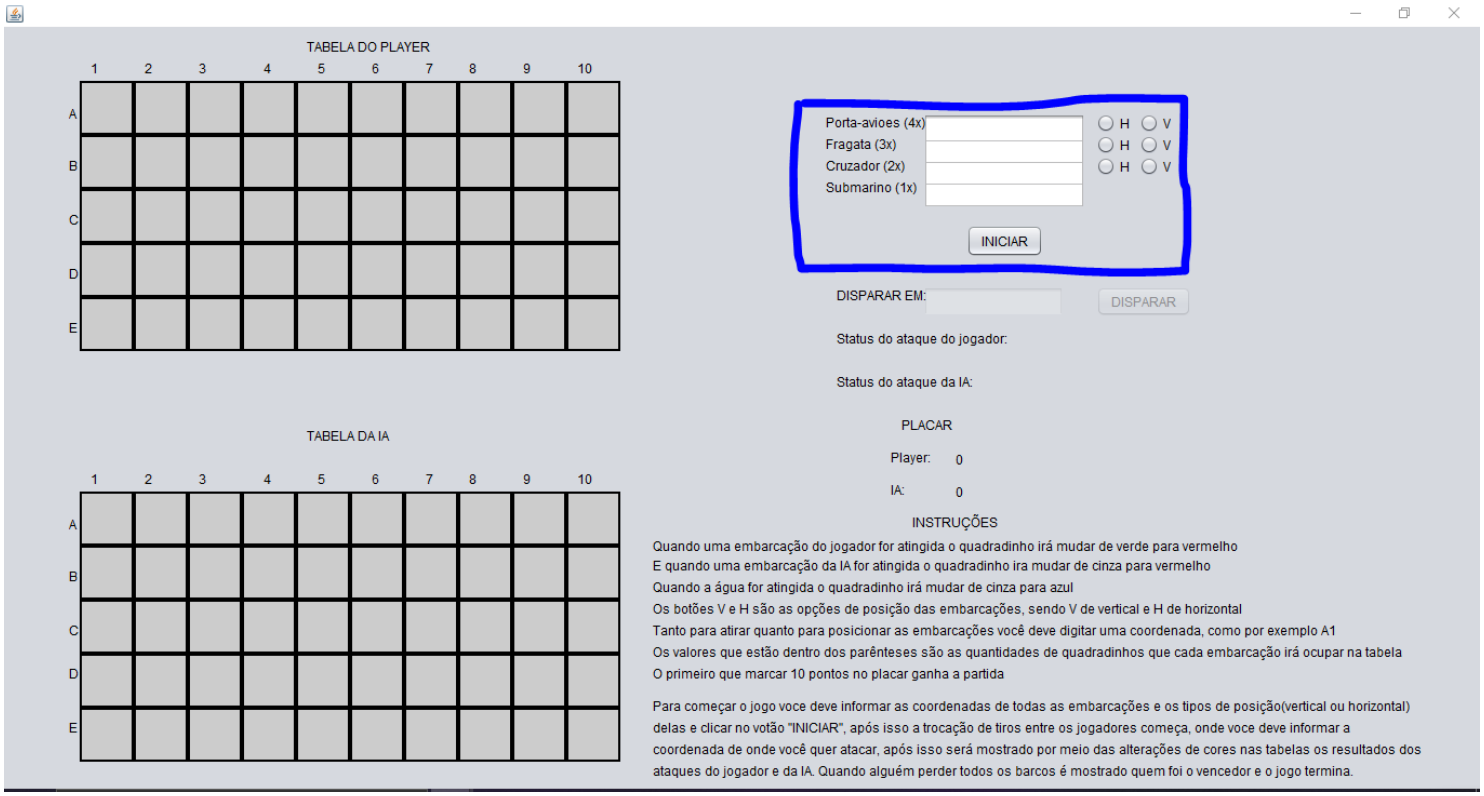


Na parte circulada em vermelho estão as tabelas de painéis do jogador e da IA:

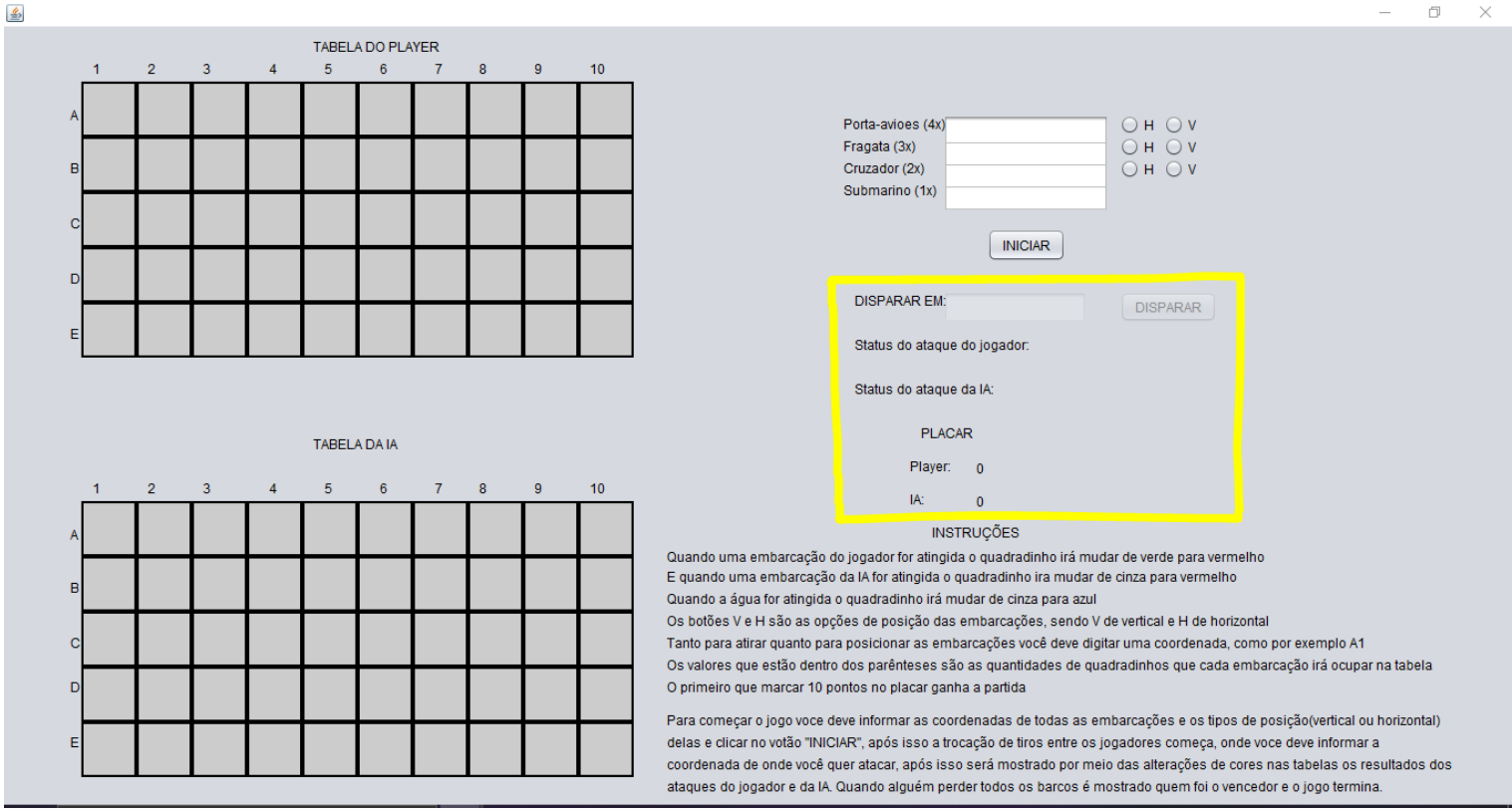


Quando criamos os painéis do jogador, definimos os nomes deles como o nome da coordenada onde ele está, por exemplo, chamamos de “A1” o painel que está na primeira linha e na primeira coluna da tabela; para os painéis da IA também definimos os nomes deles como o nome da coordenada onde ele está, mas para diferenciarmos dos painéis do jogador, antes de sua coordenada é escrito “IA”, por exemplo, chamamos de “IAA1” o painel que está na primeira linha e na primeira coluna da tabela da IA.

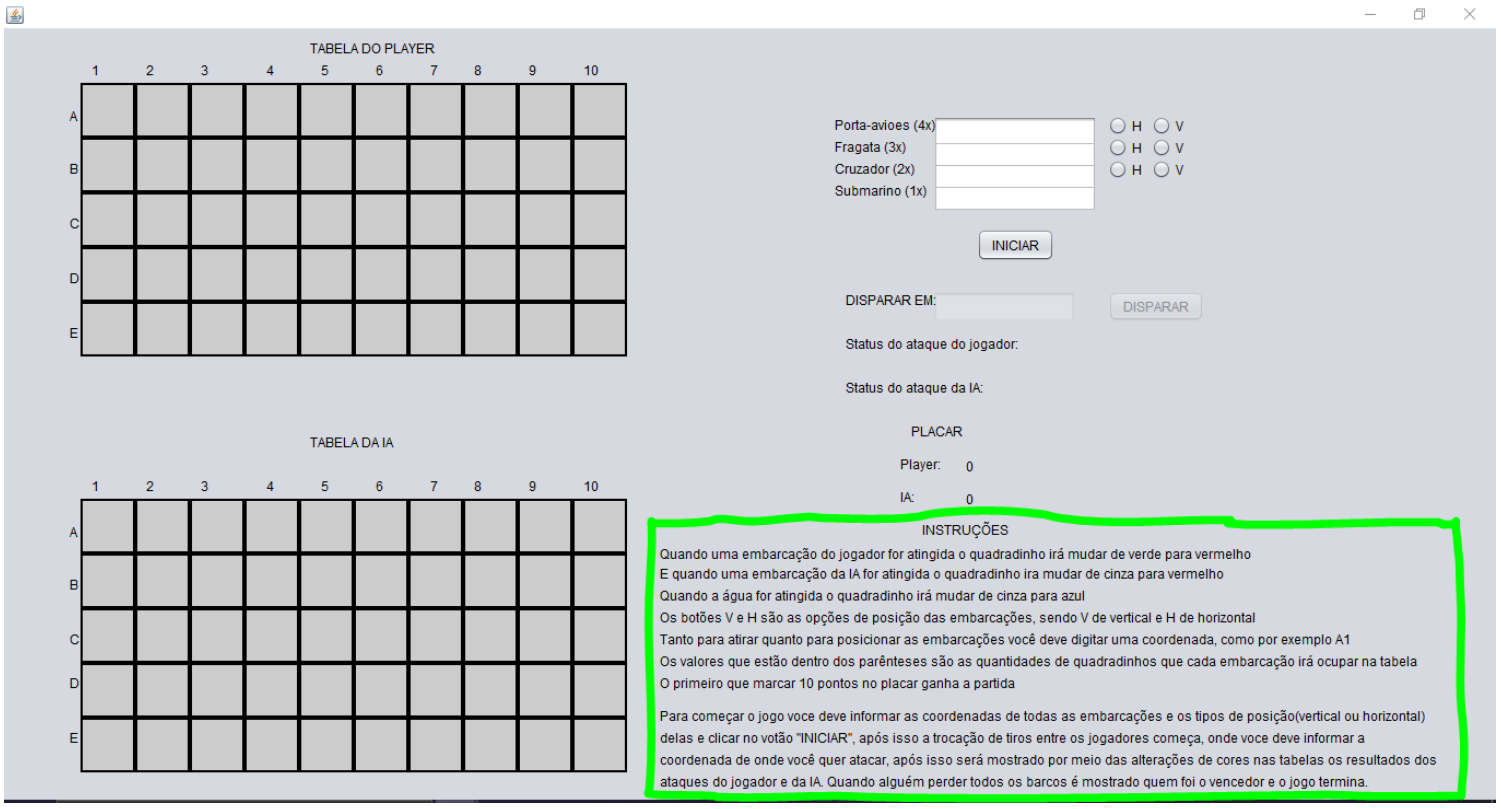
Na parte circulada em azul estão os campos onde o jogador informa as coordenadas e tipos de posição (horizontal e vertical) de onde ele quer posicionar suas embarcações e o botão onde o jogador “envia” os valores informados:



Na parte circulada em amarelo estão os campos onde o jogador informa a coordenada de onde ele quer atacar e o botão ataques. Também estão os campos onde serão mostrados os status dos ataques do jogador e da IA (ataque à embarcação ou na água) e o placar da partida (quantas embarcações cada jogador acertou):



Na parte circulada em verde estão algumas instruções sobre como o jogo funciona:



Para jogar o jogador deve informar as coordenadas onde serão adicionados as embarcações e os tipos de posições (vertical e horizontal), onde caso o jogador escolha horizontal a embarcação será posicionada da esquerda para a direita e caso o jogador escolha vertical a embarcação será posicionada de cima para baixo.

DESENVOLVIMENTO

Quando o programa é executado o método main “abre” a tela do jogo.

```
5252
5253    /*Método main que criará e mostrará a tela do jogo*/
5254    public static void main(String args[]) {
5255        /* Set the Nimbus look and feel */
5256        Look and feel setting code (optional)
5277
5278        /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
5281                new naval().setVisible(b:true);
5282            }
5283        });
5284    }
5285
```

Fazemos os imports que serão necessários na aplicação, na classe declaramos as matrizes onde os painéis do jogador e da IA ficarão armazenadas, a matriz de painéis que armazenará os status dos painéis da IA e o vetor tem barcos, que armazenará se o jogador e a IA ainda tem embarcações não destruídas.

```
Source  Design  History
1   /*Autores: Lucas Rodrigues Moreira, Gabriel Guimarães Barbos
2   package btnaval;
3
4   import java.awt.Color;
    import static java.awt.PageAttributes.MediaType.A1;
6   import java.util.Random;
    import java.util.Vector;
    import java.util.concurrent.TimeUnit;
    import java.util.logging.Level;
    import java.util.logging.Logger;
11  import javax.swing.JOptionPane;
12  import javax.swing.JPanel;
13
14  /**
15   *
16   * @author 0068949
17   */
18  public class naval extends javax.swing.JFrame {
19
20      /*Declaramos as matrizes onde os paineis do jogador e da
21      public JPanel[][] matrizPanels = new JPanel[5][10];
22      public JPanel[][] matrizPanelsIA = new JPanel[5][10];
23      public int[][] statusmatrizPanelsIA = new int[5][10];
        public Vector<Integer> tembarco = new Vector<>();
25
```

Após isso fazemos a configurações iniciais, além da configuração padrão gerada pela IDE, chamamos o método adicionarpanel, que adicionará os painéis do jogador à matriz de painéis do jogador

```
26      /*Aqui fazemos as configurações iniciais*/
27      public naval() {
28          initComponents();
29
30          /*Chamamos o método que adicionará os panéis da tabela do jogador à matriz de painéis do jogador*/
            adicionarpanels();
32
```

No método adicionarpanel adicionamos os painéis nas posições da matriz de painéis do jogador de modo que a sua posição coincida com seu nome.

```java
    /*Método que adiciona os paineis do jogador na matriz de painéis do jogadores, onde cada painel do jogador é adicionado à sua resp
    public void adicionarpanels() {
        matrizPanels[0][0] = A1;
        matrizPanels[0][1] = A2;
        matrizPanels[0][2] = A3;
        matrizPanels[0][3] = A4;
        matrizPanels[0][4] = A5;
        matrizPanels[0][5] = A6;
        matrizPanels[0][6] = A7;
        matrizPanels[0][7] = A8;
        matrizPanels[0][8] = A9;
        matrizPanels[0][9] = A10;

        matrizPanels[1][0] = B1;
        matrizPanels[1][1] = B2;
        matrizPanels[1][2] = B3;
        matrizPanels[1][3] = B4;
        matrizPanels[1][4] = B5;
        matrizPanels[1][5] = B6;
        matrizPanels[1][6] = B7;
        matrizPanels[1][7] = B8;
        matrizPanels[1][8] = B9;
        matrizPanels[1][9] = B10;

```

```java
            matrizPanels[2][0] = C1;
            matrizPanels[2][1] = C2;
            matrizPanels[2][2] = C3;
            matrizPanels[2][3] = C4;
            matrizPanels[2][4] = C5;
            matrizPanels[2][5] = C6;
            matrizPanels[2][6] = C7;
            matrizPanels[2][7] = C8;
            matrizPanels[2][8] = C9;
            matrizPanels[2][9] = C10;

            matrizPanels[3][0] = D1;
            matrizPanels[3][1] = D2;
            matrizPanels[3][2] = D3;
            matrizPanels[3][3] = D4;
            matrizPanels[3][4] = D5;
            matrizPanels[3][5] = D6;
            matrizPanels[3][6] = D7;
            matrizPanels[3][7] = D8;
            matrizPanels[3][8] = D9;
```

```
            matrizPanels[3][0] = D1;
            matrizPanels[3][1] = D2;
            matrizPanels[3][2] = D3;
            matrizPanels[3][3] = D4;
            matrizPanels[3][4] = D5;
            matrizPanels[3][5] = D6;
            matrizPanels[3][6] = D7;
            matrizPanels[3][7] = D8;
            matrizPanels[3][8] = D9;
            matrizPanels[3][9] = D10;

            matrizPanels[4][0] = E1;
            matrizPanels[4][1] = E2;
            matrizPanels[4][2] = E3;
            matrizPanels[4][3] = E4;
            matrizPanels[4][4] = E5;
            matrizPanels[4][5] = E6;
            matrizPanels[4][6] = E7;
            matrizPanels[4][7] = E8;
            matrizPanels[4][8] = E9;
            matrizPanels[4][9] = E10;

        }
```

Voltando para as configurações iniciais:

Chamamos o método adicionarpanelIA, que adicionará os painéis da IA à matriz de painéis da IA

```
            /*Chamamos o método que adicionará os panéis da tabela da IA à matriz de painéis da IA*/
            adicionapanelIA();
```

No método adicionarpanelIA adicionamos os painéis nas posições da matriz de painéis da IA de modo que a sua posição coincida com seu nome.

```
    /*Método que adiciona os painéis da IA à matriz de painéis da IA*/
    private void adicionapanelIA() {
        matrizPanelsIA[0][0] = IAA1;
        matrizPanelsIA[0][1] = IAA2;
        matrizPanelsIA[0][2] = IAA3;
        matrizPanelsIA[0][3] = IAA4;
        matrizPanelsIA[0][4] = IAA5;
        matrizPanelsIA[0][5] = IAA6;
        matrizPanelsIA[0][6] = IAA7;
        matrizPanelsIA[0][7] = IAA8;
        matrizPanelsIA[0][8] = IAA9;
        matrizPanelsIA[0][9] = IAA10;

        matrizPanelsIA[1][0] = IAB1;
        matrizPanelsIA[1][1] = IAB2;
        matrizPanelsIA[1][2] = IAB3;
        matrizPanelsIA[1][3] = IAB4;
        matrizPanelsIA[1][4] = IAB5;
        matrizPanelsIA[1][5] = IAB6;
        matrizPanelsIA[1][6] = IAB7;
        matrizPanelsIA[1][7] = IAB8;
        matrizPanelsIA[1][8] = IAB9;
        matrizPanelsIA[1][9] = IAB10;
```

```
4829
4830            matrizPanelsIA[2][0] = IAC1;
4831            matrizPanelsIA[2][1] = IAC2;
4832            matrizPanelsIA[2][2] = IAC3;
4833            matrizPanelsIA[2][3] = IAC4;
4834            matrizPanelsIA[2][4] = IAC5;
4835            matrizPanelsIA[2][5] = IAC6;
4836            matrizPanelsIA[2][6] = IAC7;
4837            matrizPanelsIA[2][7] = IAC8;
4838            matrizPanelsIA[2][8] = IAC9;
4839            matrizPanelsIA[2][9] = IAC10;
4840
4841            matrizPanelsIA[3][0] = IAD1;
4842            matrizPanelsIA[3][1] = IAD2;
4843            matrizPanelsIA[3][2] = IAD3;
4844            matrizPanelsIA[3][3] = IAD4;
4845            matrizPanelsIA[3][4] = IAD5;
4846            matrizPanelsIA[3][5] = IAD6;
4847            matrizPanelsIA[3][6] = IAD7;
4848            matrizPanelsIA[3][7] = IAD8;
4849            matrizPanelsIA[3][8] = IAD9;
4850            matrizPanelsIA[3][9] = IAD10;
```

```
4851
4852            matrizPanelsIA[4][0] = IAE1;
4853            matrizPanelsIA[4][1] = IAE2;
4854            matrizPanelsIA[4][2] = IAE3;
4855            matrizPanelsIA[4][3] = IAE4;
4856            matrizPanelsIA[4][4] = IAE5;
4857            matrizPanelsIA[4][5] = IAE6;
4858            matrizPanelsIA[4][6] = IAE7;
4859            matrizPanelsIA[4][7] = IAE8;
4860            matrizPanelsIA[4][8] = IAE9;
4861            matrizPanelsIA[4][9] = IAE10;
4862        }
```

Voltando para as configurações iniciais:

Após isso, para que não haja ataques antes dos jogadores posicionarem os barcos, desativamos o botão e o campo de texto referente aos ataques, após isso inicializamos o vetor tembarco com 2 0’s, representando o jogador e a IA, respectivamente e em seguida inicializamos os placares.

```
35
36          /*Desativamos o botão de adicionar barcos e bloqueamos o jTextField4 para que não haja ataques antes de que as embarcações sejam posicionadas*/
37          jButton2.setEnabled( b: false);
38          jTextField4.setEnabled( enabled: false);
39
40          /*Adicionamos 2 0's ao vetor tem barcos, que representarão o jogador e a IA, respectivamente*/
41          tembarco.add( e: 0);
42          tembarco.add( e: 0);
43
44          /*Setamos os valores iniciais dos placares*/
45          jLabel47.setText( text: "0");
46          jLabel48.setText( text: "0");
```

E inicializamos a matriz de status dos painéis da tabela da IA, que começam com zeros pois neles ainda não há embarcações posicionadas.

```
48          /*Preenchemos a matriz dos status das embarcações da IA com 0's, representando que alí ainda não há embarcações*/
49          for (int i = 0; i < 5; i++) {
50              for (int j = 0; j < 10; j++) {
51                  statusmatrizPanelsIA[i][j] = 0;
52              }
53          }
54      }
55
```

Agora que começamos a codificar a parte interativa do programa:

Quando o jogador apertar no botão ”INICIAR", primeiramente definimos a variável que servirá de controle para sabermos se todas as coordenadas informadas pelo jogador para posicionar embarcações ao posições válidas. Depois criamos o vetor indice, que armazenará as coordenadas informadas pelo jogador. Após isso, inicializamos as 8 posições do vetor indice (4 embarcações, 2 valores para cada embarcação (linha e coluna))

```
4864    /*Método que será executado quando o jogador clicar no botão "INICIAR"*/
        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
4866
4867        /*Criamos a variável que servirá de controle de sabermos se todas as coordenadas informadas pelo jogador para posicionar embarcações são posições válid
4868        int t = 0;
4869
4870        /*Criamos o vetor indice, que armazenará as coordenadas da matriz de paineis do jogador referentes às informadas pelo jogador*/
4872        Vector indice = new Vector<Integer>();
4873        /*Inicializamos os 8 valores(linha e coluna) das coordenadas das 4 emabrcações*/
4874        indice.add( e: 0);
4875        indice.add( e: 0);
4876        indice.add( e: 0);
4877        indice.add( e: 0);
4878        indice.add( e: 0);
4879        indice.add( e: 0);
4880        indice.add( e: 0);
4881        indice.add( e: 0);
4882
```

Usamos 2 for’s para analisarmos todas as posições da matriz de painéis do jogador. Após isso verificamos se o nome do painel é igual à coordenada informada pelo jogador para o porta-aviões, em caso afirmativo, adicionamos as coordenadas ao vetor índice, sendo a primeira referente à linha e a segunda à coluna. E em seguida atualizamos o valor da variável de controle.

```
4883        /*Criamos dois for's que passarão por todas as posições da matriz de painéis do jogador*/
4884        for (int a = 0; a < 10; a++) {
4885            for (int b = 0; b < 5; b++) {
4886
4887                /*Analizamos se o nome do painel localizado na coordenada analizada é igual à coordenada informada pelo jogador para o porta-aviões*/
4888                if (matrizPanels[b][a].getName().equals( anObject: jTextField1.getText())) {
4889
4890                    /*Adicionados as coordenadas do painel ao vetor indice*/
4891                    indice.set( index: 0,  element: b);
4892                    indice.set( index: 1,  element: a);
4893
4894                    /*Atualizamos o valor da variável controle*/
4895                    t++;
4896
4897                }
4898            }
4899        }
```

Usamos 2 for’s para analisarmos todas as posições da matriz de painéis do jogador. Após isso verificamos se o nome do painel é igual à coordenada informada pelo jogador para a fragata, em caso afirmativo, adicionamos as coordenadas ao vetor índice, sendo a primeira referente à linha e a segunda à coluna. E em seguida atualizamos o valor da variável de controle.

```
4901        /*Criamos dois for's que passarão por todas as posições da matriz de painéis do jogador*/
4902        for (int a = 0; a < 10; a++) {
4903            for (int b = 0; b < 5; b++) {
4904
4905                /*Analizamos se o nome do painel localizado na coordenada analizada é igual à coordenada informada pelo jogador para a fragata*/
4906                if (matrizPanels[b][a].getName().equals( anObject: jTextField2.getText())) {
4907
4908                    /*Adicionados as coordenadas do painel ao vetor indice*/
4909                    indice.set( index: 2,  element: b);
4910                    indice.set( index: 3,  element: a);
4911
4912                    /*Atualizamos o valor da variável controle*/
4913                    t++;
4914                }
4915            }
4916        }
4917
```

Usamos 2 for’s para analisarmos todas as posições da matriz de painéis do jogador. Após isso verificamos se o nome do painel é igual à coordenada informada pelo jogador para o cruzador, em caso afirmativo, adicionamos as coordenadas ao vetor índice, sendo a primeira referente à linha e a segunda à coluna. E em seguida atualizamos o valor da variável de controle.

```
4917
4918            /*Criamos dois for's que passarão por todas as posições da matriz de painéis do jogador*/
4919            for (int a = 0; a < 10; a++) {
4920                for (int b = 0; b < 5; b++) {
4921
4922                    /*Analizamos se o nome do painel localizado na coordenada analizada é igual à coordenada informada pelo jogador para o cruzador*/
4923                    if (matrizPanels[b][a].getName().equals( anObject: jTextField3.getText())) {
4924
4925                        /*Adicionados as coordenadas do painel ao vetor indice*/
4926                        indice.set( index: 4,  element: b);
4927                        indice.set( index: 5,  element: a);
4928
4929                        /*Atualizamos o valor da variável controle*/
4930                        t++;
4931                    }
4932                }
4933            }
4934
```

Usamos 2 for’s para analisarmos todas as posições da matriz de painéis do jogador. Após isso verificamos se o nome do painel é igual à coordenada informada pelo jogador para o submarino, em caso afirmativo, adicionamos as coordenadas ao vetor índice, sendo a primeira referente à linha e a segunda à coluna. E em seguida atualizamos o valor da variável de controle.

```
4934
4935            /*Criamos dois for's que passarão por todas as posições da matriz de painéis do jogador*/
4936            for (int a = 0; a < 10; a++) {
4937                for (int b = 0; b < 5; b++) {
4938
4939                    /*Analizamos se o nome do painel localizado na coordenada analizada é igual à coordenada informada pelo jogador para o submarino*/
4940                    if (matrizPanels[b][a].getName().equals( anObject: jTextField5.getText())) {
4941
4942                        /*Adicionados as coordenadas do painel ao vetor indice*/
4943                        indice.set( index: 6,  element: b);
4944                        indice.set( index: 7,  element: a);
4945
4946                        /*Atualizamos o valor da variável controle*/
4947                        t++;
4948                    }
4949                }
4950            }
4951
```

Verificamos se o jogador não deixou nenhum campo de texto vazio:

```
4951
4952        /*Verificamos se não há nehum campo de coordenada vazio*/
4953        if (jTextField1.getText().isBlank() == false && jTextField2.getText().isBlank() == false && jTextField3.getText().isBlank() == false && jTextField5.getTe
4954
```

```
4952    o de coordenada vazio*/
4953    ) == false && jTextField2.getText().isBlank() == false && jTextField3.getText().isBlank() == false && jTextField5.getText().isBlank() == false) {
4954
```

Em caso positivo mostramos uma mensagem para o jogador:

```
4984                /*Se há campos de coordenadas vazios, mostramos uma mensagem para o jogador*/
4985                } else {
4986
4987                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Nenhum dos espaços para embarcações podem estar vazios");
4988                }
4989            }
```

Em caso negativo continuamos com as verificações:

Verificamos se o jogador escolheu mais de um tipo de posição para todas as embarcações com mais de 2 painéis de comprimento (o submarino não tem tipos de posições, pois ele só possui um painel de comprimento), em caso afirmativo, mostramos para o jogador uma mensagem:

```
4955        /*Caso o jogador tenha selecionado, para uma mesma embarcação, a opção de horizontal e a de vertical, mostramos uma mensagem para o jogador*/
4956        if ((H1.isSelected() == true && V1.isSelected() == true) || (H2.isSelected() == true && V2.isSelected() == true) || (H3.isSelected() == true && V3.isSele
4957            JOptionPane.showMessageDialog( parentComponent: null,  message: "Embarcações não podem estar na vetical e na horizontal ao mesmo tempo");
4958
```

```
4955    selecionado, para uma mesma embarcação, a opção de horizontal e a de vertical, mostramos uma mensagem para o jogador*/
4956     true && V1.isSelected() == true) || (H2.isSelected() == true && V2.isSelected() == true) || (H3.isSelected() == true && V3.isSelected() == true)) {
4957    sageDialog( parentComponent: null,  message: "Embarcações não podem estar na vetical e na horizontal ao mesmo tempo");
4958
```

Em caso negativo continuamos com as verificações:

Verificamos se as coordenadas das 4 embarcações foram encontradas.

```
        /*Caso o jogador não tenha selecionado, para uma mesma embarcação, a opção de horizontal e a de vertical*/
        } else {

            /*Verificamos se as 4 coordenadas das embarcações informadas pelo jogador foram coordenadas válidas*/
            if (t == 4) {
```

Em caso negativo mostramos uma mensagem para o jogador:

```
            /*Caso alguma das coordenadas das embarcações informadas pelo jogador foram coordenadas inválidas, mostramos uma mensagem para o jogador*/
            } else {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Digite valores válidos");

            }

        }
```

Em caso positivo mostramos verificamos se alguma das coordenadas informadas pelo jogador são coordenadas que fazem com que alguma das embarcações ficarem com alguma parte para fora da tabela, para isso analisamos separadamente cada embarcação dentro do if de acordo com o seu comprimento e sua coordenada informada.

```
        /*Caso o jogador tenha escolhido, para cada embarcação, uma opção de modo de posição(vertical ou horizontal), mas a coordenada faz com que parte da emba
        if (((H1.isSelected() && (int) indice.get( index: 1) >= 7) || (H2.isSelected() && (int) indice.get( index: 3) >= 8) || (H3.isSelected() && (int) indice.get( i
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Digite valores válidos");

        /*Caso não haja nenhhum problema com as coordenadas e com os modos de posição(horizontal e vertical), chamamos o método que muda a cor dos painéis do j
        } else {

            mudacor(indice);

        }
```

```
 ou horizontal), mas a coordenada faz com que parte da embarcação fique para fora do tabuleiro, é mostrada uma mensagem para o jogador */
:t( index: 3) >= 8) || (H3.isSelected() && (int) indice.get( index: 5) >= 9)) || ((V1.isSelected() && (int) indice.get( index: 0) >= 2) || (V2.isSelected() && (int)


 tical), chamamos o método que muda a cor dos painéis do jogador onde há embarcações posicionadas*/
```

```
 tabuleiro, é mostrada uma mensagem para o jogador */
lected() && (int) indice.get( index: 0) >= 2) || (V2.isSelected() && (int) indice.get( index: 2) >= 3) || (V3.isSelected() && (int) indice.get( index: 4) >= 4))) {


 posicionadas*/
```

Caso as coordenadas sejam válidas chamamos o método que vai mudar a cor dos painéis do jogador onde foram posicionadas embarcações (para que o jogo fique mais dinâmico para o jogador)

```
                /*Caso não haja nenhhum problema com as coordenadas e com os modos de posição(horizontal e vertical), chamamos o método que muda a cor dos p
                } else {

                    mudacor(indice);

                }
```

Verificamos se as opções de modos de posicionar as embarcações do jogador foram selecionados:

```
    /*Método que mudará a cor dos painéis onde o jogador posicionar as embarcações*/
    private void mudacor(Vector indice) {

        /*Verificamos se as opções de modos de posicionar os barcos foram selecionados*/
        if ((H1.isSelected() || V1.isSelected()) && (H2.isSelected() || V2.isSelected()) && (H3.isSelected() || V3.isSelected())) {
```

Em caso negativo, mostramos uma mensagem para o jogador

```
                /*Caso não seja informado o tipo de posição(horizontal ou vertical) de alguma embarcação, mostramos uma mensagem para o jogador*/
            } else {

                JOptionPane.showMessageDialog( parentComponent: null,  message: "selecione um tipo de posicao para cada embarcação");

            }
```

Em caso positivo começamos o processo de mudar as cores dos painéis do jogador:

Criamos a variável comprimento, que recebera o comprimento que a respectiva embarcação deverá ter.

Agora vamos posicionar o porta-aviões:

Caso o jogador tenha escolhido posicionar o porta-aviões na horizontal, atualizamos o valor da variável comprimento para 4, que é o comprimento do porta-aviões.

```
4441                /*Criamos a variável que receberá o comprimento que cada embarcação deverá ter*/
                    int comprimento = 0;
4443
4444                /*Se o jogador escolheu posicionar o porta-aviões na horizontal*/
4445                if (H1.isSelected()) {
4446
4447                    /*Adicionamos à variavel comprimento o comprimento do porta-aviões*/
4448                    comprimento = 4;
```

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá agora existe uma embarcação posicionada, note que o índice “x” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direita na horizontal.

```
4450            /*Criamos um for que repetirá 4 vezes, correspondendo ao 4 paineis de comprimento do porta-aviões*/
4451            for (int x = 0; x < comprimento; x++) {
4452
4453                /*Alteramos a cor dos paineis localizados na coordenada informada e nas 3 à direita para verde, representando
4454                matrizPanels[(int) indice.get( index: 0)][(int) indice.get( index: 1) + x].setBackground( bg: Color.green);
4455            }
```

Depois de posicionar o porta-aviões, desativamos os botões e o campo referente à embarcação

```
4457            /*Após isso desativamos os botões e campo referente ao porta-aviões, já que esta embarcação já foi adicionada ao tabuleiro*/
4458            H1.setEnabled( b: false);
4459            V1.setEnabled( b: false);
4460            jTextField1.setEnabled( enabled: false);
4461
4462        }
4463
```

Caso o jogador tenha escolhido posicionar o porta-aviões na vertical, atualizamos o valor da variável comprimento para 4, que é o comprimento do porta-aviões.

```
4464                /*Se o jogador escolheu posicionar o porta-aviões na vertical*/
4465                if (V1.isSelected()) {
4466
4467                    /*Adicionamos à variavel comprimento o comprimento do porta-aviões*/
4468                    comprimento = 4;
```

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá agora existe uma embarcação posicionada, note que o índice “x” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
4470                /*Criamos um for que repetirá 4 vezes, correspondendo ao 4 paineis de comprimento do porta-aviões*/
4471                for (int x = 0; x < comprimento; x++) {
4472
4473                    /*Alteramos a cor dos paineis localizados na coordenada informada e nos 3 abaixo para verde, represen
4474                    matrizPanels[(int) indice.get( index: 0) + x][(int) indice.get( index: 1)].setBackground( bg: Color.green);
4475                }
```

Depois de posicionar o porta-aviões, desativamos os botões e o campo referente à embarcação

```
                    /*Após isso desativamos os botões e campo referent
                    H1.setEnabled( b: false);
                    V1.setEnabled( b: false);
                    jTextField1.setEnabled( enabled: false);

                }
```

Agora vamos posicionar a fragata:

Caso o jogador tenha escolhido posicionar a fragata na horizontal, verificamos se nos 3 painéis onde queremos colocar a fragata não há embarcações já posicionadas (se os painéis não estão verdes)

```
    /*Se o jogador escolheu posicionar a fragata na horizontal*/
    if (H2.isSelected()) {

        /*Analisamos se não existe embarcações já posicionadas nas posições onde o jogador informou para posicionar a fragata*/
        if ((matrizPanels[(int) indice.get( index: 2)][(int) indice.get( index: 3)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2)][(int
```

```
cionar a fragata*/
 && (matrizPanels[(int) indice.get( index: 2)][(int) indice.get( index: 3) + 1].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2)][(int)
```

```
:3) + 1].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2)][(int) indice.get( index: 3) + 2].getBackground() != Color.GREEN)) {
```

Em caso positivo, atualizamos o valor da variável comprimento para 3, que é o comprimento da fragata.

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá agora existe uma embarcação posicionada, note que o índice “x” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direta na horizontal.

```
                    /*Adicionamos à variavel comprimento o comprimento da fragata*/
                    comprimento = 3;

                    /*Criamos um for que repetirá 3 vezes, correspondendo ao 3 paineis de comprimento da fragata*/
                    for (int x = 0; x < comprimento; x++) {

                        /*Alteramos a cor dos paineis localizados na coordenada informada e nas 2 à direita para verde, repres
                        matrizPanels[(int) indice.get( index: 2)][(int) indice.get( index: 3) + x].setBackground( bg: Color.green);
                    }
```

Depois de posicionar a fragata, desativamos os botões e o campo referente à embarcação

```
                /*Após isso desativamos os botões e campo referente à fragata, já que esta embarcação já foi adicionada ao tabuleiro*/
                H2.setEnabled( b: false);
                V2.setEnabled( b: false);
                jTextField2.setEnabled( enabled: false);

            }

        }
```

Caso o jogador tenha escolhido posicionar a fragata na vertical, verificamos se nos 3 painéis onde queremos colocar a fragata não há embarcações já posicionadas (se os painéis não estão verdes)

```
    /*Se o jogador escolheu posicionar a fragata na vertical*/
    if (V2.isSelected()) {

        /*Analisamos se não existe embarcações já posicionadas nas posições onde o jogador informou para posicionar a fragata*/
        if ((matrizPanels[(int) indice.get( index: 2)][(int) indice.get( index: 3)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2) + 1][(
```

```
 indice.get( index: 2) + 1][(int) indice.get( index: 3)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2) + 2][(int) indice.get( index: 3)]
```

```
:t( index: 3)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 2) + 2][(int) indice.get( index: 3)].getBackground() != Color.GREEN)) {
```

Em caso positivo, atualizamos o valor da variável comprimento para 3, que é o comprimento da fragata.

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá

agora existe uma embarcação posicionada, note que o índice “x” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
4515                    /*Adicionamos à variavel comprimento o comprimento da fragata*/
4516                    comprimento = 3;
4517
4518                    /*Criamos um for que repetirá 3 vezes, correspondendo ao 3 paineis de comprimento da fragata*/
4519                    for (int x = 0; x < comprimento; x++) {
4520
4521                        /*Alteramos a cor dos paineis localizados na coordenada informada e nas 2 abaixo para verde, representa
4522                        matrizPanels[(int) indice.get( index: 2) + x][(int) indice.get( index: 3)].setBackground( bg: Color.green);
4523                    }
```

Depois de posicionar a fragata, desativamos os botões e o campo referente à embarcação

```
4524
4525                    /*Após isso desativamos os botões e campo referente à fra
4526                    H2.setEnabled( b: false);
4527                    V2.setEnabled( b: false);
4528                    jTextField2.setEnabled( enabled: false);
4529
4530                }
4531
4532            }
```

Agora vamos posicionar o cruzador:

Caso o jogador tenha escolhido posicionar o cruzador na horizontal, verificamos se nos 2 painéis onde queremos colocar o cruzador não há embarcações já posicionadas (se os painéis não estão verdes)

```
4534    /*Se o jogador escolheu posicionar o cruzador na horizontal*/
4535    if (H3.isSelected()) {
4536
4537        /*Analisamos se não existe embarcações já posicionadas nas posições onde o jogador informou para posicionar o cruzador*/
4538        if ((matrizPanels[(int) indice.get( index: 4)][(int) indice.get( index: 5)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4)][(int)
4539
```

```
4537
4538    ) indice.get( index: 4)][(int) indice.get( index: 5) + 1].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4)][(int) indice.get( index: 5) + 2
4539
```

```
4537
4538     kground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4)][(int) indice.get( index: 5) + 2].getBackground() != Color.GREEN)) {
4539
```

Em caso positivo, atualizamos o valor da variável comprimento para 2, que é o comprimento do cruzador.

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá agora existe uma embarcação posicionada, note que o índice “x” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direta na horizontal.

```
4540                    /*Adicionamos à variavel comprimento o comprimento do cruzador*/
4541                    comprimento = 2;
4542
4543                    /*Criamos um for que repetirá 2 vezes, correspondendo ao 2 paineis de comprimento do cruzador*/
4544                    for (int x = 0; x < comprimento; x++) {
4545
4546                        /*Alteramos a cor dos paineis localizados na coordenada informada e à direita para verde, representan
4547                        matrizPanels[(int) indice.get( index: 4)][(int) indice.get( index: 5) + x].setBackground( bg: Color.green);
4548                    }
```

Depois de posicionar o cruzador, desativamos os botões e o campo referente à embarcação

```
4549
4550                /*Após isso desativamos os botões e campo referente ao cruzador, já que esta embarcação já foi adicionada ao tabuleiro*/
4551                H3.setEnabled( b: false);
4552                V3.setEnabled( b: false);
4553                jTextField3.setEnabled( enabled: false);
4554
4555            }
4556
4557        }
```

Caso o jogador tenha escolhido posicionar o cruzador na vertical, verificamos se nos 2 painéis onde queremos colocar o cruzador não há embarcações já posicionadas (se os painéis não estão verdes)

```
4559    /*Se o jogador escolheu posicionar o cruzador na vertical*/
4560    if (V3.isSelected()) {
4561
4562        /*Analisamos se não existe embarcações já posicionadas nas posições onde o jogador informou para posicionar o cruzador*/
4563        if ((matrizPanels[(int) indice.get( index: 4)][(int) indice.get( index: 5)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4) + 1][(i
4564
```

```
4563    indice.get( index: 4) + 1][(int) indice.get( index: 5)].getBackground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4) + 2][(int) indice.get( index: 5)].
4564
4565
```

```
4562
4563    kground() != Color.GREEN) && (matrizPanels[(int) indice.get( index: 4) + 2][(int) indice.get( index: 5)].getBackground() != Color.GREEN)) {
4564
```

Em caso positivo, atualizamos o valor da variável comprimento para 2, que é o comprimento do cruzador.

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele mudamos (“setamos”) a cor do background do painel da respectiva coordenada para verde, indicando que lá agora existe uma embarcação posicionada, note que o índice “x” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
4564
4565                    /*Adicionamos à variavel comprimento o comprimento do cruzador*/
4566                    comprimento = 2;
4567
4568                    /*Criamos um for que repetirá 2 vezes, correspondendo ao 2 paineis de comprimento do cruzador*/
4569                    for (int x = 0; x < comprimento; x++) {
4570
4571                        /*Alteramos a cor dos paineis localizados na coordenada informada e uma abaixo para verde, representan
4572                        matrizPanels[(int) indice.get( index: 4) + x][(int) indice.get( index: 5)].setBackground( bg: Color.green);
4573                    }
4574
```

Depois de posicionar o cruzador, desativamos os botões e o campo referente à embarcação

```
4574
4575                        /*Após isso desativamos os botões e campo referente ao cruzador, já que e
4576                        H3.setEnabled( b: false);
4577                        V3.setEnabled( b: false);
4578                        jTextField3.setEnabled( enabled: false);
4579
4580                    }
4581
4582                }
4583
```

Por fim, vamos posicionar o submarino:

Verificamos se no painel onde queremos colocar o submarino não há embarcações já posicionadas (se os painéis não estão verdes)

Em caso positivo, mudamos a cor do painel da respectiva coordenada e desativamos o campo referente à embarcação.

```
4583
4584            /*Agora vamos adicionar o submarino do jogador*/
4585            /*Analisamos se não existe embarcações já posicionadas na posição onde o jogador informou para posicionar o submarino*/
4586            if (matrizPanels[(int) indice.get( index: 6)][(int) indice.get( index: 7)].getBackground() != Color.GREEN) {
4587
4588                /*Alteramos a cor dos paineis localizados na coordenada informada para verde, representando que uma embarcação foi adicionada*/
4589                matrizPanels[(int) indice.get( index: 6)][(int) indice.get( index: 7)].setBackground( bg: Color.green);
4590
4591                /*Após isso desativamos o campo referente ao submarino, já que esta embarcação já foi adicionada ao tabuleiro*/
4592                jTextField5.setEnabled( enabled: false);
4593            }
```

Após isso verificamos se todos os campos de textos das embarcações estão desativados, ou sejam, se todos as embarcações do jogador já foram posicionadas. Em caso positivo desativamos o botão “INICIAR” e chamamos o método que sorteará as posições das embarcações da IA e as posicionar.

```
4602        /*Verificamos se todos os campos de coordenadas das embarcações já estiverem inativos, ou seja, se todos os barcos do jogador já foram posicionados*/
4603        if (jTextField1.isEnabled() == false && jTextField2.isEnabled() == false && jTextField3.isEnabled() == false && jTextField5.isEnabled() == false) {
4604
4605            /*Desativamos o botão iniciar e chamamos o método que posicionará as embarcações da IA*/
4606            jButton1.setEnabled( b: false);
4607            posicionaIA();
4608        }
4609    }
4610
```

No método posicionaIA declaramos as variáveis que receberão as coordenadas das embarcações da IA e a que receberá o tipo de posicionamento das embarcações (vertical ou horizontal). E sorteamos o tipo de posicionamento do porta aviões, caso 0 for sorteado, será posicionado na horizontal, caso 1 for sorteado, será posicionado na vertical.

```
4611        /*Método que posicionará as embarcações da IA*/
4612        private void posicionaIA() {
4613
4614            /*Declaramos as variáveis que armazenarão os valores coluna, da linha e tipo de posição das embarcações, respectivamente*/
4615            int x, y, p;
4616
4617            /*Declaramos o random*/
4618            Random sorteia = new Random();
4619
4620            /*Sorteamos o tipo de posição para o porta-aviões*/
4621            /*0 é horizontal, 1 é vertical*/
4622            p = sorteia.nextInt( origin: 0,  bound: 2);
```

Caso o porta-aviões for ser posicionado na horizontal, sorteamos as coordenadas de onde posicionaremos o porta-aviões e caso a coluna sorteada faça com que parte da embarcação fique para fora da tabela, a sortearemos novamente enquanto isso ocorre.

```
4624            /*Posicionando o porta-aviões*/
4625            /*Caso o porta-aviões seja colocado na horizontal*/
4626            if (p == 0) {
4627
4628                /*Sorteamos os valores da coluna e da linha da emb
4629                x = sorteia.nextInt( origin: 0,  bound: 10);
4630
4631                y = sorteia.nextInt( origin: 0,  bound: 5);
4632
4633                /*Caso o valor sorteado para a coluna faça com que
4634                while (x > 6) {
4635                    x = sorteia.nextInt( origin: 0,  bound: 10);
4636
4637                }
```

Criamos e atualizamos o valor da variável comprimento para 4, que é o comprimento do porta-aviões. Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direta na horizontal.

```
4639                /*Adicionamos à variavel comprimento o comprimento do porta
4640                int comprimento = 4;
4641
4642                /*Criamos um for que repetirá 4 vezes, correspondendo ao 4
4643                for (int a = 0; a < comprimento; a++) {
4644
4645                    /*Atualizamos o valor da coordenada da matriz de status
4646                    statusmatrizPanelsIA[y][x + a] = 1;
4647                }
```

Caso o porta-aviões não for ser posicionado na horizontal, ou seja, for ser posicionado na vertical, sorteamos as coordenadas de onde posicionaremos o porta-aviões e caso a linha sorteada faça com que parte da embarcação fique para fora da tabela, enquanto isso ocorre a sortearemos novamente. Criamos e atualizamos o valor da variável comprimento para 4, que é o comprimento do porta-aviões.

```
        } else {

            /*Sorteamos os valores da coluna e da linha da embarcação*/
            x = sorteia.nextInt( origin: 0,  bound: 10);

            y = sorteia.nextInt( origin: 0,  bound: 5);

            /*Caso o valor sorteado para a linha faça com que a embarcação fique co
            while (y > 1) {
                y = sorteia.nextInt( origin: 0,  bound: 5);

            }

            /*Adicionamos à variavel comprimento o comprimento do porta-aviões*/
            int comprimento = 4;
```

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
            /*Criamos um for que repetirá 4 vezes, cor
            for (int a = 0; a < comprimento; a++) {

                /*Atualizamos o valor da coordenada da
                statusmatrizPanelsIA[y + a][x] = 1;
            }
        }
```

Agora vamos posicionar a fragata.

Primeiramente sorteamos o tipo de posicionamento da fragata, caso 0 for sorteado, será posicionada na horizontal, caso 1 for sorteado, será posicionada na vertical.

Caso a fragata for ser posicionada na horizontal, sorteamos as coordenadas de onde posicionaremos a fragata e caso a coluna sorteada faça com que parte da embarcação fique para fora da tabela ou ocupe posições onde já haviam embarcações posicionadas anteriormente, a sortearemos novamente enquanto isso ocorre.

```
        /*Sorteamos o tipo de posição para a fragata*/
        /*0 é horizontal, 1 é vertical*/
        p = sorteia.nextInt( origin: 0,  bound: 2);

        /*Posicionando a fragata*/
        /*Caso a fragata seja colocado na horizontal*/
        if (p == 0) {

            /*Sorteamos os valores da coluna e da linha da embarcação*/
            x = sorteia.nextInt( origin: 0,  bound: 10);

            y = sorteia.nextInt( origin: 0,  bound: 5);

            /*Caso o valor sorteado para a coluna faça com que a embarcação fique com partes para fora do tabuleiro ou se em alguma das posiçõe
            while (x > 7 || (statusmatrizPanelsIA[y][x] == 1 || statusmatrizPanelsIA[y][x + 1] == 1 || statusmatrizPanelsIA[y][x + 2] == 1)) {
                x = sorteia.nextInt( origin: 0,  bound: 10);

            }
```

Criamos e atualizamos o valor da variável comprimento para 3, que é o comprimento da fragata. Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direta na horizontal.

```
4692
4693                /*Adicionamos à variavel comprimento o comprimento da fr
4694                int comprimento = 3;
4695
4696                /*Criamos um for que repetirá 3 vezes, correspondendo ao
4697                for (int a = 0; a < comprimento; a++) {
4698
4699                    /*Atualizamos o valor da coordenada da matriz de sta
4700                    statusmatrizPanelsIA[y][x + a] = 1;
4701                }
4702
```

Caso a fragata não for ser posicionada na horizontal, ou seja, for ser posicionada na vertical, sorteamos as coordenadas de onde posicionaremos a fragata e caso a linha sorteada faça com que parte da embarcação fique para fora da tabela ou ocupe posições onde já haviam embarcações posicionadas anteriormente, a sortearemos novamente enquanto isso ocorre. Criamos e atualizamos o valor da variável comprimento para 3, que é o comprimento da fragata.

```
4703            /*Caso a fragata seja colocada na vertical*/
4704        } else {
4705
4706            /*Sorteamos os valores da coluna e da linha da embarcação*/
4707            x = sorteia.nextInt( origin: 0,  bound: 10);
4708
4709            y = sorteia.nextInt( origin: 0,  bound: 5);
4710
4711            /*Caso o valor sorteado para a linha faça com que a embarcação fique com partes para fora do tabuleiro ou se em alguma das posições o
4712            while (y > 2 || (statusmatrizPanelsIA[y][x] == 1 || statusmatrizPanelsIA[y + 1][x] == 1 || statusmatrizPanelsIA[y + 2][x] == 1)) {
4713                y = sorteia.nextInt( origin: 0,  bound: 10);
4714
4715            }
4716
4717            /*Adicionamos à variavel comprimento o comprimento da fragata*/
4718            int comprimento = 3;
4719
```

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
4720                /*Criamos um for que repetirá 3 vezes, corr
4721                for (int a = 0; a < comprimento; a++) {
4722
4723                    /*Atualizamos o valor da coordenada da
4724                    statusmatrizPanelsIA[y + a][x] = 1;
4725                }
4726
4727            }
4728
```

Agora vamos posicionar o cruzador.

Primeiramente sorteamos o tipo de posicionamento do cruzador, caso 0 for sorteado, será posicionado na horizontal, caso 1 for sorteado, será posicionado na vertical.

Caso o cruzador for ser posicionado na horizontal, sorteamos as coordenadas de onde posicionaremos o cruzador e caso a coluna sorteada faça com que parte da embarcação fique para fora da tabela ou ocupe posições onde já haviam embarcações posicionadas anteriormente, a sortearemos novamente enquanto isso ocorre.

```
4729            /*Sorteamos o tipo de posição para o cruzador*/
4730            /*0 é horizontal, 1 é vertical*/
4731            p = sorteia.nextInt( origin: 0,  bound: 2);
4732
4733            /*Posicionando o cruzador*/
4734            /*Caso o cruzador seja colocado na horizontal*/
4735            if (p == 0) {
4736
4737                /*Sorteamos os valores da coluna e da linha da embarcação*/
4738                x = sorteia.nextInt( origin: 0,  bound: 10);
4739
4740                y = sorteia.nextInt( origin: 0,  bound: 5);
4741
4742                /*Caso o valor sorteado para a coluna faça com que a embarcação fique com partes para fora do
4743                while (x > 8 || (statusmatrizPanelsIA[y][x] == 1 || statusmatrizPanelsIA[y][x + 1] == 1)) {
4744                    x = sorteia.nextInt( origin: 0,  bound: 10);
4745
4746                }
4747
```

Criamos e atualizamos o valor da variável comprimento para 2, que é o comprimento do cruzador. Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no segundo “[ ]” da matriz, isso é porque a deslocação ocorrerá da esquerda para a direta na horizontal.

```
4747
4748                /*Adicionamos à variavel comprimento o comprimento do cruza
4749                int comprimento = 2;
4750
4751                /*Criamos um for que repetirá 2 vezes, correspondendo ao 2
4752                for (int a = 0; a < comprimento; a++) {
4753
4754                    /*Atualizamos o valor da coordenada da matriz de status
4755                    statusmatrizPanelsIA[y][x + a] = 1;
4756                }
4757
```

Caso o cruzador não for ser posicionado na horizontal, ou seja, for ser posicionado na vertical, sorteamos as coordenadas de onde posicionaremos o cruzador e caso a linha sorteada faça com que parte da embarcação fique para fora da tabela ou ocupe posições onde já haviam embarcações posicionadas anteriormente, a sortearemos novamente enquanto isso ocorre. Criamos e atualizamos o valor da variável comprimento para 2, que é o comprimento do cruzador.

```
4758                /*Caso o cruzador seja colocado na vertical*/
4759            } else {
4760
4761                /*Sorteamos os valores da coluna e da linha da embarcação*/
4762                x = sorteia.nextInt( origin: 0,  bound: 10);
4763
4764                y = sorteia.nextInt( origin: 0,  bound: 5);
4765
4766                /*Caso o valor sorteado para a linha faça com que a embarcação fique com partes para fora do
4767                while (y > 3 || (statusmatrizPanelsIA[y][x] == 1 || statusmatrizPanelsIA[y + 1][x] == 1)) {
4768                    y = sorteia.nextInt( origin: 0,  bound: 10);
4769
4770                }
4771
4772                /*Adicionamos à variavel comprimento o comprimento do cruzador*/
4773                int comprimento = 2;
```

Criamos um for que repetirá um número de vezes correspondente ao comprimento da embarcação, onde nele atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali. Note que o índice “a” do for está no primeiro “[ ]” da matriz, isso é porque a deslocação ocorrerá de cima para baixo na vertical.

```
4775                /*Criamos um for que repetirá 2 vezes, corresponden
4776                for (int a = 0; a < comprimento; a++) {
4777
4778                    /*Atualizamos o valor da coordenada da matriz d
4779                    statusmatrizPanelsIA[y + a][x] = 1;
4780                }
4781
4782            }
```

E por fim, vamos posicionar o submarino.

Sorteamos as coordenadas de onde posicionaremos o submarino e caso as coordenadas sorteadas façam com que ocupe posições onde já haviam embarcações posicionadas anteriormente, a sortearemos novamente enquanto isso ocorre. Após isso atualizamos o status do painel da respectiva coordenada analisada para 1, representado que agora foi posicionado uma embarcação ali.

Após as embarcações dos dois jogadores terem sido posicionadas, ativamos o botão e o campo de coordenadas do ataque.

```
4783
4784            /*Posicionando o submarino*/
4785            /*Caso o cruzador seja colocado na horizontal*/
4786            x = sorteia.nextInt( origin: 0,  bound: 10);
4787
4788            y = sorteia.nextInt( origin: 0,  bound: 5);
4789
4790            /*Enquanto a posição sorteada já estiver ocupada p
4791            while (statusmatrizPanelsIA[y][x] == 1) {
4792                x = sorteia.nextInt( origin: 0,  bound: 10);
4793                y = sorteia.nextInt( origin: 0,  bound: 5);
4794
4795            }
4796
4797            /*Atualizamos o valor da coordenada da matriz de s
4798            statusmatrizPanelsIA[y][x] = 1;
4799
4800            /*Já que as embarcações do jogador e da IA já fora
4801            jButton2.setEnabled( b: true);
4802            jTextField4.setEnabled( enabled: true);
4803
4804        }
4805
```

Após o jogador ter clicado no botão de ataque, concatenamos a coordenada informada com a String “IA” e atualizamos o valor do vetor tembarco, chamando o método tembarco, que analisa se os dois jogadores ainda têm embarcações não destruídas.

```
5040        /*Método que será executado quando o jogador clicar no botão "DISPARAR"*/
            private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
5042
5043            /*Definimos as sequência de letras "IA", que deverão vir antes das c
5044            String defaulttxt = "IA";
5045
5046            /*Concatenamos a sequência de letras "IA" com a coordenada de ataque
5047            String tiro = defaulttxt.concat( str: jTextField4.getText());
5048
5049            /*Verificamos se o jogador e a IA ainda tem embarcações posicionadas
5050            tembarco = verificabarco(tembarco);
```

No método verificabarco criamos e inicializamos a variável de controle t, usamos dois for's para analisarmos todas as posições da matriz de painéis do jogador, e se um painel a matriz for verde, "setamos" como 1(ainda tem embarcações) no índice do jogador no vetor tembarco e atualizamos a variável de controle.

```
5196        /*Método que verifica se o jogador e a IA inda tem barcos que não foram destruídos
            public Vector verificabarco(Vector tembarco) {
5198
5199            /*Definimos a variável de controle*/
5200            int t = 1;
5201
5202            /*Criamos dois for's que passarão por todas as posições da matriz de painéis d
5203            for (int a = 0; a < 10; a++) {
5204                for (int b = 0; b < 5; b++) {
5205
5206                    /*Caso a cor do painel da coordenada analisada for igual a verde*/
5207                    if (matrizPanels[b][a].getBackground().equals( obj: Color.green)) {
5208
5209                        /*Atualizamos o vetor tembarco, dizendo que o jogador ainda tem em
5210                        tembarco.set( index: 0,  element: 1);
5211
5212                        /*Atualizamos a variável de controle*/
5213                        t++;
5214                    }
5215
5216                }
5217            }
5218
```

Caso não tenha acontecido alterações na variável de controle, "setamos" como 0(não tem mais embarcações) no índice do jogador no vetor tembarco e reiniciamos o valor da variável t, após isso usamos dois for's para analisarmos todas as posições da matriz de painéis do jogador, e se um painel a matriz de painéis da IA for tiver o status igual à 1(tem embarcação posicionada), "setamos" como 1(ainda tem embarcações) no índice da IA no vetor tembarco e atualizamos a variável de controle.

```
5219        /*Se t for igual a 1, ou seja, não ocorreu a atualizaçã
5220        if (t == 1) {
5221            tembarco.set( index: 0,  element: 0);
5222        }
5223
5224        /*Voltamos o t para o valor incial*/
5225        t = 1;
5226
5227        /*Criamos dois for's que passarão por todas as posições
5228        for (int a = 0; a < 10; a++) {
5229            for (int b = 0; b < 5; b++) {
5230
5231                /*Caso o status do painel da coordenada analisa
5232                if (statusmatrizPanelsIA[b][a] == 1) {
5233
5234                    /*Atualizamos o vetor tembarco, dizendo que
5235                    tembarco.set( index: 1,  element: 1);
5236
5237                    /*Atualizamos a variável de controle*/
5238                    t++;
5239                }
5240
5241            }
5242        }
5243
```

Caso não tenha acontecido alterações na variável de controle, "setamos" como 0(não tem mais embarcações) no índice da IA no vetor tembarco e retornamos o vetor tembarco, que possui os valores obtidos pela análise.

```
5243
5244            /*Se t for igual a 1, ou seja, não occ
5245            if (t == 1) {
5246                tembarco.set( index: 1,  element: 0);
5247            }
5248
5249            /*E então, retornamos o vetor tembarco
5250            return tembarco;
5251        }
5252
```

Voltando para o ataque:

Caso ambos os jogadores ainda tenham embarcações não destruídas, chamamos o método que fará o ataque e à IA.

```
5052        /*Se o jogador e a IA ainda tem embarcações posicionadas que não foram destruídas*/
5053        if (tembarco.get( index: 0) == 1 && tembarco.get( index: 1) == 1) {
5054
5055            /*Chamamos o método que fará o ataque à IA*/
5056            consultamatrizdaIA(tiro);
```

No método consultamaatrizdaIA usamos dois for's para analisarmos todas as posições ada matriz de painéis da IA, caso o painel do índice analisado tiver o mesmo nome da coordenada que o jogador informou que acontecesse o ataque e que nessa coordenada ainda haja embarcações não destruídas, "setamos" a cor do background do painel de vermelho para informar que no painel atacado havia uma embarcação e que ela foi atingida. Em seguida atualizamos o placar do jogador e atualizamos o status do painel da posição analisada para 2(tinha embarcação, mas foi atacado) e alteramos a mensagem do status do ataque do jogador, avisando que ocorreu um tiro em navio

```
5117    public void consultamatrizdaIA(String tiro) {
5118        //Modelo dos status do painel da IA: 0 igual à vazio, 1 tem embarcação, 2 tem embarcação que levou tiro e
5119
5120        /*Criamos dois for's que passarão por todas as posições da matriz de painéis da IA*/
5121        for (int a = 0; a < 10; a++) {
5122            for (int b = 0; b < 5; b++) {
5123
5124                /*Caso o painel da coordenada analizada tenha o mesmo nome que o informado pelo jogador e nele ha
5125                if (matrizPanelsIA[b][a].getName().equals( anObject: tiro) && statusmatrizPanelsIA[b][a] == 1) {
5126
5127                    /*Pintamos o painel de vermelho para representar que alí tinha uma emabarcação, mas ela levou
5128                    matrizPanelsIA[b][a].setBackground( bg: Color.red);
5129
5130                    /*Atualizamos o placar do jogador*/
5131                    int placarjogador = Integer.parseInt( s: jLabel48.getText());
5132                    placarjogador++;
5133                    jLabel48.setText( text: Integer.toString( i: placarjogador));
5134
5135                    /*Atualizamos o status do painel*/
5136                    statusmatrizPanelsIA[b][a] = 2;
5137
5138                        /*Atualizamos o status do ataque do jogador*/
5139                        jLabel50.setText( text: "Status do ataque do jogador: Tiro em navio");
```

Caso alguma das condições do if não forem atendidas, analisamos se o painel do índice analisado tiver o mesmo nome da coordenada que o jogador informou que acontecesse o ataque e que nessa coordenada não haja embarcações, seja elas destruídas ou não, ou seja, for água, "setamos" a cor do background do painel de azul para informar que no painel atacado era água. E atualizamos o status do painel da posição analisada para 3(houve tiro na água) e alteramos a mensagem do status do ataque do jogador, avisando que ocorreu um tiro na água.

```
5141                /*Caso o painel da coordenada analizada não tenha o mesmo nome que o informado pelo jogador ou nele não haja uma embarcação não destruída*/
5142                } else {
5143
5144                    /*Caso o painel da coordenada analizada tenha o mesmo nome que o informado pelo jogador e nele seja água*/
5145                    if (matrizPanelsIA[b][a].getName().equals( anObject: tiro) && statusmatrizPanelsIA[b][a] != 1 && matrizPanelsIA[b][a].getName().equals( anObject
5146
5147                        /*Pintamos o painel de azul para representar que ali só tinha água, mas ela levou um tiro*/
5148                        matrizPanelsIA[b][a].setBackground( bg: Color.blue);
5149
5150                        /*Atualizamos o status do painel*/
5151                        statusmatrizPanelsIA[b][a] = 3;
5152
5153                        /*Atualizamos o status do ataque do jogador*/
5154                        jLabel50.setText( text: "Status do ataque do jogador: Tiro na água");
5155                    }
5156                }
5157
5158            }
5159        }
5160
5161    }
5162
```

```
5144    izada tenha o mesmo nome que o informado pelo jogador e nele seja água*/
5145    .equals( anObject: tiro) && statusmatrizPanelsIA[b][a] != 1 && matrizPanelsIA[b][a].getName().equals( anObject: tiro) && statusmatrizPanelsIA[b][a] != 2) {
5146
5147    ra representar que ali só tinha água, mas ela levou um tiro*/
5148    ound( bg: Color.blue);
5149
5150    el*/
```

Voltando para o ataque:

Após o ataque do jogador, analisamos se os dois jogadores ainda têm embarcações não afundadas. Após isso criamos as variáveis que receberão as coordenadas do ataque da IA, criamos o Random que fará o sorteio e sorteamos as coordenadas do ataque da IA e salvamos nas respectivas variáveis.

```
5058                /*Verificamos se o jogador e a IA ainda tem en
5059                tembarco = verificabarco(tembarco);
5060
5061                /*Definimos as variáveis que receberão os valo
5062                int x, y;
5063
5064                /*Declaramos o random*/
5065                Random sorteia = new Random();
5066
5067
5068                /*Sorteamos os valores das coodenadas(linha e
5069                x = sorteia.nextInt( origin: 0,  bound: 10);
5070
5071                y = sorteia.nextInt( origin: 0,  bound: 5);
5072
```

Caso a coordenada sorteada for a de uma embarcação já destruída ou água que já foi atirada, sorteamos novamente as coordenadas do ataque da IA. Após isso chamamos o método que fará o ataque da IA.

```
5074                /*Enquanto o valor das coordenadas sorteadas forem de um painel que já foi atacado anteriormente, mas tinha um barco ou era água, s
5075                while (matrizPanels[y][x].getBackground().equals( obj: Color.red) || matrizPanels[y][x].getBackground().equals( obj: Color.blue)) {
5076                    x = sorteia.nextInt( origin: 0,  bound: 10);
5077
5078                    y = sorteia.nextInt( origin: 0,  bound: 5);
5079                }
5080
5081                /*Chamamos o método que fará o ataque ao jogador*/
5082                consultamatrizdoJogador(tiro, x, y);
```

No método consultamatrizdoJogador:

Caso o painel atacado do jogador seja verde (tenha embarcação não destruída), “setamos” a cor do background do painel de vermelho para informar que no painel atacado havia uma embarcação e que ela foi atingida. Em seguida atualizamos o placar da IA e alteramos a mensagem do status do ataque da IA avisando que ocorreu um tiro em navio.

```
5163        /*Método que fará o ataque ao jogador*/
5164        public void consultamatrizdoJogador(String tiro, int x, int y) {
5165
5166            /*Caso o painel com a coordenada informada seja verde, ou seja, tiver
5167            if (matrizPanels[y][x].getBackground().equals( obj: Color.green)) {
5168
5169                /*Pintamos o painel de vermelho para representar que alí tinha um
5170                matrizPanels[y][x].setBackground( bg: Color.red);
5171
5172                /*Atualizamos o placar da IA*/
5173                int placarIA = Integer.parseInt( s: jLabel47.getText());
5174                placarIA++;
5175                jLabel47.setText( text: Integer.toString( i: placarIA));
5176
5177                /*Atualizamos o status do ataque da IA*/
5178                jLabel46.setText( text: "Status do ataque da IA: Tiro em navio");
5179
```

Caso o painel atacado do jogador seja não verde (não tenha embarcação não destruída), e como definimos anteriormente no código, a IA não pode atacar um mesmo painel duas vezes, então “setamos” a cor do background do painel de azul para informar que no painel atacado era água. Em seguida atualizamos alteramos a mensagem do status do ataque da IA avisando que ocorreu um tiro na água.

```
5180            /*Caso o painel com a coordenada informada não seja verde*/
5181            } else {
5182
5183                /*Caso o painel com a coordenada informada não seja verde*/
5184                if (matrizPanels[y][x].getBackground() != (Color.green)) {
5185
5186                    /*Pintamos o painel de azul para representar que alí só tinha água,
5187                    matrizPanels[y][x].setBackground( bg: Color.blue);
5188
5189                    /*Atualizamos o status do ataque da IA*/
5190                    jLabel46.setText( text: "Status do ataque da IA: Tiro na água");
5191                }
5192            }
5193
5194        }
5195
```

Voltando para o ataque:

Após o ataque da IA, analisamos se os dois jogadores ainda têm embarcações não afundadas.

```
5084                /*Verificamos se o jogador e a IA ainda tem embarcações posicionadas que não foram destruídas*
5085                tembarco = verificabarco(tembarco);
5086
5087            }
5088
```

Caso o jogador não tenha mais embarcações não destruídas e a IA ainda tenha mais embarcações não destruídas, mostramos uma mensagem dizendo que a IA ganhou o jogo. Caso a IA não tenha mais embarcações não destruídas e o jogador ainda tenha mais embarcações não destruídas, mostramos uma mensagem dizendo que o jogador ganhou o jogo. Em ambos os casos, após a mensagem desativamos o botão e o campo de texto do ataque, pois o jogo já acabou.

```
5089            /*Caso somente o jogador não tenha mais embarções(todas foram destruídas), mostram
5090            if (tembarco.get( index: 0) == 0 && tembarco.get( index: 1) == 1) {
5091                JOptionPane.showMessageDialog( parentComponent: null,  message: "A IA ganhou");
5092                jTextField4.setEnabled( enabled: false);
5093                jButton2.setEnabled( b: false);
5094            }
5095
5096            /*Caso somente a IA não tenha mais embarções(todas foram destruídas), mostramos um
5097            if (tembarco.get( index: 0) == 1 && tembarco.get( index: 1) == 0) {
5098                JOptionPane.showMessageDialog( parentComponent: null,  message: "O jogador ganhou");
5099                jTextField4.setEnabled( enabled: false);
5100                jButton2.setEnabled( b: false);
5101            }
5102
```

E caso o jogador e a IA não tenham mais embarcações não destruídas, mostramos uma mensagem dizendo que houve um empate. Após a mensagem desativamos o botão e o campo de texto do ataque, pois o jogo já acabou.

```
5103            /*Caso o jogador e a IA não tenham mais embarções(todas foram destruídas), mostr
5104            if (tembarco.get( index: 0) == 0 && tembarco.get( index: 1) == 0) {
5105                JOptionPane.showMessageDialog( parentComponent: null,  message: "Houve um empate");
5106                jTextField4.setEnabled( enabled: false);
5107                jButton2.setEnabled( b: false);
5108            }
5109
5110        }
```

E então a documentação da jogatina já foi feito, restando apenas falar sobre algumas características que foram adicionadas ao código:

Quando o jogador clica sobre um dos radioButtons dos tipos de posicionamento (vertical ou horizontal), automaticamente o seu respectivo oposto é deselecionado, por exemplo, ao clicar no V1 o H1 é deselecionado e vice-versa, ao clicar no H2 o V2 é deselecionado e vice-versa, etc.

```
4992    private void H1MouseClicked(java.awt.event.MouseEvent evt) {
4993
4994        H1.setSelected( b: true);
4995        V1.setSelected( b: false);
4996
4997    }
4998
4999    private void H1ActionPerformed(java.awt.event.ActionEvent evt) {
5000        // TODO add your handling code here:
5001    }
5002
5003    /*Quando o jogador seleciona o radio button V1, o radio button H1 é deselecionado*/
5004    private void V1MouseClicked(java.awt.event.MouseEvent evt) {
5005
5006        V1.setSelected( b: true);
5007        H1.setSelected( b: false);
5008
5009
5010    }
5011
5012    /*Quando o jogador seleciona o radio button H2, o radio button V2 é deselecionado*/
5013    private void H2MouseClicked(java.awt.event.MouseEvent evt) {
5014        H2.setSelected( b: true);
5015        V2.setSelected( b: false);
5016    }
```

```
5017        /*Quando o jogador seleciona o radio button V2, o radio button H2 é deselecionado*/
            private void V2MouseClicked(java.awt.event.MouseEvent evt) {
5019            V2.setSelected( b: true);
5020            H2.setSelected( b: false);
5021        }
5022
5023        /*Quando o jogador seleciona o radio button H3, o radio button V3 é deselecionado*/
            private void H3MouseClicked(java.awt.event.MouseEvent evt) {
5025            H3.setSelected( b: true);
5026            V3.setSelected( b: false);
5027        }
5028
5029
5030        /*Quando o jogador seleciona o radio button V3, o radio button H3 é deselecionado*/
            private void V3MouseClicked(java.awt.event.MouseEvent evt) {
5032            V3.setSelected( b: true);
5033            H3.setSelected( b: false);
5034        }
5035
            private void V1ActionPerformed(java.awt.event.ActionEvent evt) {
5037            // TODO add your handling code here:
5038        }
5039
```

E as demais linhas de código referem à declaração dos painéis, que ocorre de maneira automática pelo GUI Builder:

```
5286        // Variables declaration - do not modify
5287        private javax.swing.JPanel A1;
5288        private javax.swing.JPanel A10;
5289        private javax.swing.JPanel A2;
5290        private javax.swing.JPanel A3;
5291        private javax.swing.JPanel A4;
5292        private javax.swing.JPanel A5;
5293        private javax.swing.JPanel A6;
5294        private javax.swing.JPanel A7;
```

CONCLUSÃO

Comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação;

O projeto foi interessante de se desenvolver pois é bem interativo e usa diversos recursos que deixa a aplicação bem dinâmica. Algumas dificuldades encontradas foram o uso de novas funções recém-aprendidas, no caso o uso de aplicações gráficas usando GUI.

REFERÊNCIAS

Aulas de programação com o professor Saulo Henrique Cabral Silva – IFMG Campus Ouro Branco