



INSTITUTO FEDERAL DE MINAS GERAIS

CAMPUS OURO BRANCO

**TÉCNICO INTEGRADO EM INFORMÁTICA**

### **NOME DOS AUTORES**

- Lucas Rodrigues Moreira
- Gabriel Guimarães Barbosa

## **TÍTULO**

### **Documentação Do Trabalho Prático 4 - Batalha de Cartas Colecionáveis**

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**

**Agosto de 2023**

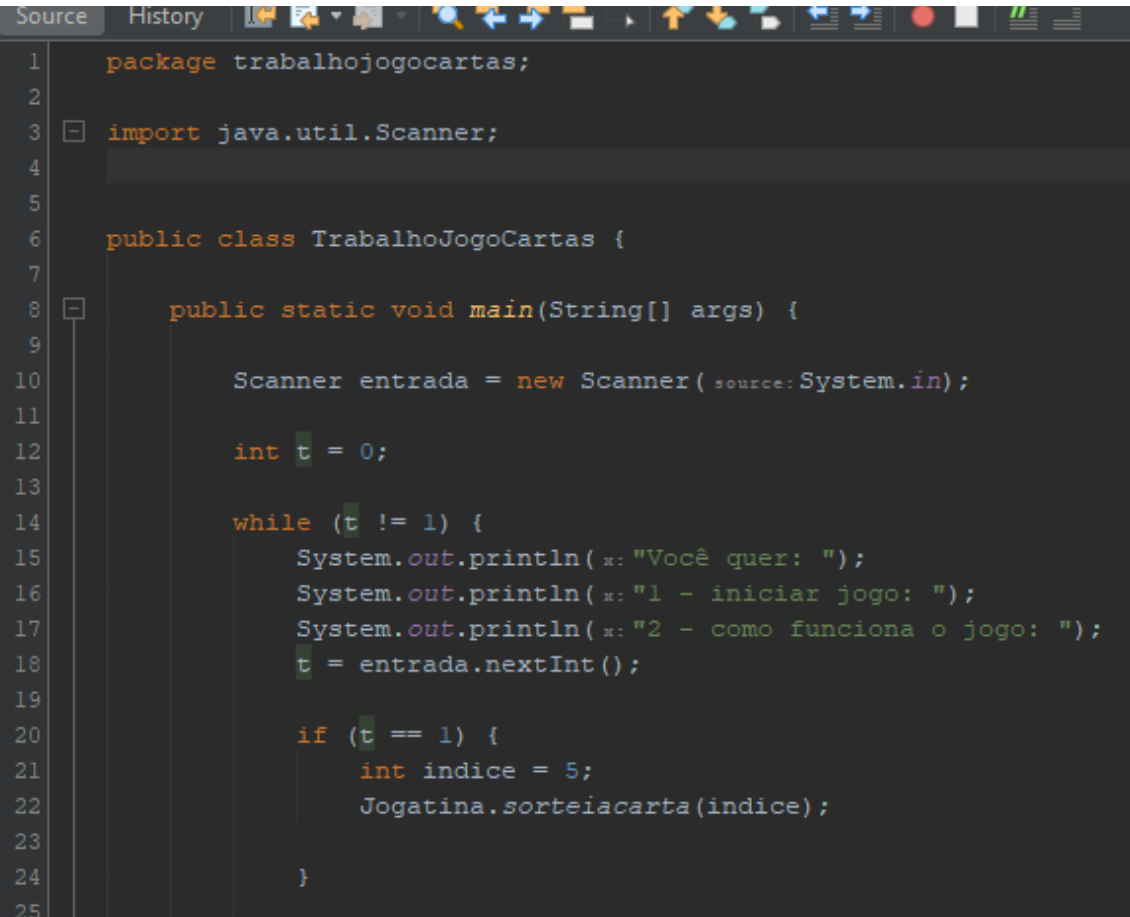
## INTRODUÇÃO

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

O programa tem a função de simular, em forma de texto interativo, simular um jogo de cartas colecionáveis semelhante aos conhecidos Magic e Yu-Gi-Oh. O jogo consiste em utilizar estratégias para posicionar monstros no tabuleiro, equipá-los com cartas de equipamento e realizar ataques contra o oponente.

Durante uma rodada, os jogadores podem escolher entre as seguintes ações: Posicionar um novo monstro no tabuleiro; equipar um monstro com uma carta de equipamento; descartar uma carta da mão; realizar um ataque contra o oponente; alterar o estado de um monstro (ataque/defesa). Leia na próxima seção algumas informações que auxiliam na solução do problema.

## DESENVOLVIMENTO



```
1 package trabalhojogocartas;
2
3 import java.util.Scanner;
4
5
6 public class TrabalhoJogoCartas {
7
8     public static void main(String[] args) {
9
10         Scanner entrada = new Scanner(System.in);
11
12         int t = 0;
13
14         while (t != 1) {
15             System.out.println("Você quer: ");
16             System.out.println("1 - iniciar jogo: ");
17             System.out.println("2 - como funciona o jogo: ");
18             t = entrada.nextInt();
19
20             if (t == 1) {
21                 int indice = 5;
22                 Jogatina.sorteiacarta(indice);
23
24             }
25
26         }
```

Declaramos a variável `t` do tipo `int` inicializada com 0, que será responsável por armazenar o número informado a seguir pelo usuário.

Criamos um laço de repetição “while” para repetir as opções e salvar o valor de `t` enquanto `t` for diferente de 1, ou seja, enquanto o jogo não for inicializado, as opções são repetidas. Se o usuário informar 1, a variável `indice` do tipo `int` é

inicializada com 5, esta variável representa as 5 primeiras cartas que serão sorteadas que formarão o baralho inicial dos 2 jogadores. E o método `sorteiacarta` da classe `Jogatina` é invocada, levando o valor da variável `indice` como parâmetro.

```

25
26         if (t == 2) {
27             instrucoes();
28         }
29
30         if (t != 1 && t != 2) {
31             System.out.println("índice não disponível");
32         }
33     }
34 }
35
36

```

Caso o usuário informar 2, o método `instrucoes` é invocado e as instruções do jogo são mostradas, caso o usuário digite algum valor diferente de 1 ou 2, uma mensagem aparece informando para o usuário que a opção que ele escreveu não é válida.

Caso o usuário informar 2 as seguintes informações são mostradas para o usuário:

```

36
37 public static void instrucoes() {
38
39     System.out.println("Distribuição inicial de cartas: Cada jogador receberá aleatoriamente 5 cartas da coleção para formar sua mão inicial.");
40     System.out.println("");
41     System.out.println("Posicionamento de monstros: A cada rodada os jogadores podem selecionar entre os monstros disponíveis nas cartas na mão e "
42         + "posicionar o monstro no tabuleiro. Os jogadores podem manter posicionados em seu tabuleiro no máximo 5 monstros. \n"
43         + "Caso o limite de 5 monstros seja alcançado, o jogador não poderá posicionar mais monstros até que algum seja removido/destruído. "
44         + "Ao posicionar um monstro no tabuleiro, o usuário pode escolher se o mesmo será inserido em estado de ataque ou defesa. O valor atual do "
45         + "monstro depende exclusivamente do estado ao qual está no momento.\n"
46         + "\n"
47         + "a. Ex: Um Monstro com 2500 de ataque e 1700 de defesa. Se o mesmo está em estado de ataque vamos considerar o valor de 2500. Se o monstro "
48         + "está em estado de defesa vamos considerar o valor de 1700.\n"
49         + "b. Um monstro pode realizar ataques apenas se estiver em estado de ataque");
50
51     System.out.println("");
52     System.out.println("Limite de cartas na mão: Cada jogador pode ter no máximo 10 cartas em sua mão. Caso esse limite seja ultrapassado, o jogador "
53         + "deve descartar cartas suficientes para não exceder o limite.");
54     System.out.println("");
55
56     System.out.println("Realização de rodadas: O jogo é realizado em rodadas. A cada rodada, os jogadores recebem uma nova carta em suas mãos e "
57         + "escolhem suas ações.");
58     System.out.println("");
59
60     System.out.println("Ataques: Durante um ataque, o jogador deve escolher qual monstro do oponente deseja atacar. Se o monstro do oponente estiver "
61         + "na posição de ataque, o oponente perde pontos relativos à diferença de ataque entre os monstros. Se o monstro do oponente estiver na "
62         + "posição de defesa e a defesa for menor que o ataque do jogador, o monstro é destruído e o oponente não perde\n"
63         + "pontos de ataque. Caso a defesa seja maior que o ataque, o jogador atual (atacante) perde pontos.\n"
64         + "a. Quando um monstro realiza um ataque, o usuário ficará impedido de alterar o estado do mesmo na rodada corrente. Podendo alterar "
65         + "o estado apenas na sua próxima rodada.");
66     System.out.println("");
67     System.out.println("Restrição de ataque: Cada monstro posicionado no tabuleiro só pode atacar uma vez a cada rodada.");
68     System.out.println("Cada jogador deve iniciar o jogo com 10000 pontos. Caso a pontuação do jogador seja igual a 0 (zero), o oponente será o vencedor."
69         + " Caso o baralho não tenha mais cartas, o jogador com a maior pontuação será considerado o vencedor.");
70     System.out.println("");
71     System.out.println("Caso o usuário não tenha posicionado nenhum monstro no tabuleiro, seu adversário poderá infringir dano direto aos seus pontos.\n"
72         + "a. Os ataques podem ser realizados apenas a partir da segunda rodada.");
73
74     System.out.println("Quando o jogo solicitar que o usuário digite o índice de alguma carta, o padrão de resposta que usuário deverá usar será, "
75         + "por exemplo, em um conjunto de cartas fenix flamejante, arqueiro mágico, dragão do trovão, caso o usuário queira selecionar a carta "
76         + "fenix flamejante ele deverá digitar 1 como índice, 2 para a carta arqueiro mágico, etc.");
77     System.out.println("");
78 }
79

```

Caso o usuário informar 1, o método `sorteiacarta` da classe `Jogatina` é invocada, levando o valor da variável `indice` como parâmetro:

Fazemos os imports necessários para o desenvolvimento do código e dentro da classe `Jogatina` declaramos a variável `meuBanco` do tipo “FakeBD”, que será usado para acessar a classe `FakeBD`; a variável `rodada`, que registrará a rodada na qual o jogo está; a variável `controle`, que registrará se a pontuação inicial dos jogadores já foi salva, as variáveis

pontosjogador1 e pontosjogador2 que serão usadas para salvar as pontuações dos dois jogadores e o vetor dinâmico pontuacao que será usado para salvar as pontuações dos dois jogadores em um só vetor.

```

1  package trabalhojogocartas;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.util.Scanner;
9  import java.util.Vector;
10
11  public class Jogatina {
12
13      static FakeBD meuBanco = new FakeBD();
14      static int rodada = 1;
15      static int controle = 0;
16      static int pontosjogador1 = 0;
17      static int pontosjogador2 = 0;
18      static Vector<Integer> pontuacao = new Vector<>();
19

```

Após isso, o método sorteiacarta é executado:

Os vetores dinâmicos responsáveis por armazenar as cartas das mãos dos 2 jogadores são declarados, sendo esses vetores declarados como sendo do tipo “cartas”.

Esse tipo de variável “cartas” é criado usando conceitos da programação Orientada a Objetos, essa criação ocorre na classe “cartas”:

```

50  public static void sorteiacarta(int indice) {
51
52      Vector<cartas> maojogador1 = new Vector<>();
53      Vector<cartas> maojogador2 = new Vector<>();
54

```

São declaradas as variáveis que armazenarão os dados das cartas e em seguida é criado o construtor da classe.

```

Source History
1 package trabalhojogocartas;
2
3 public class cartas {
4
5     private String nome;
6     private String descricao;
7     private int ataque;
8     private int defesa;
9     private String tipo;
10
11     public cartas(String nome, String descricao, int ataque, int defesa, String tipo) {
12         this.nome = nome;
13         this.descricao = descricao;
14         this.ataque = ataque;
15         this.defesa = defesa;
16         this.tipo = tipo;
17     }
18

```

Os get's das variáveis são declarados:

```

17
18
19     public String getNome() {
20         return nome;
21     }
22
23     public String getDescricao() {
24         return descricao;
25     }
26
27     public int getAtaque() {
28         return ataque;
29     }
30
31     public int getDefesa() {
32         return defesa;
33     }
34
35     public String getTipo() {
36         return tipo;
37     }
38

```

Os set's das variáveis são declarados:

```

37     }
38
39     public void setNome(String nome) {
40         this.nome = nome;
41     }
42
43     public void setDescricao(String descricao) {
44         this.descricao = descricao;
45     }
46
47     public void setAtaque(int ataque) {
48         this.ataque = ataque;
49     }
50
51     public void setDefesa(int defesa) {
52         this.defesa = defesa;
53     }
54
55     public void setTipo(String tipo) {
56         this.tipo = tipo;
57     }
58

```

E declaramos o método `toString` para que os nomes das cartas sejam transformados de código para nomes que o usuário possa compreender.

```

58
59     @Override
60     public String toString() {
61         return nome;
62     }
63
64 }
65

```

Voltando a explicar a classe Jogatina:

```

53
54 File arquivoCartas = new File(pathname: "C:\\Users\\SYSTEM N.P\\Downloads\\cartas.csv");
55
56 if (arquivoCartas.exists()) {
57
58     try {
59         FileReader marcaleitura = new FileReader(file: arquivoCartas);
60
61         BufferedReader buffleitura = new BufferedReader(in: marcaleitura);
62
63         String linha;
64
65
66         do {
67             linha = buffleitura.readLine();
68
69             if (linha != null) {
70
71                 String dadoslinha[] = linha.split(regex: ";");
72
73                 int ataque = Integer.parseInt(dadoslinha[2]);
74                 int defesa = Integer.parseInt(dadoslinha[3]);
75
76                 cartas cartas = new cartas(dadoslinha[0], dadoslinha[1], ataque, defesa, dadoslinha[4]);
77
78                 meuBanco.inserecartasorteadas(cartasorteadas: cartas);
79
80             }
81
82         } while (linha != null);
83

```

Criamos a variável do tipo File chamada arquivoCartas e indicamos o link de acesso do arquivo .csv das cartas, fazemos a verificação de se o arquivo indicado pelo link existe, se sim a execução do programa continua normalmente, caso o arquivo exista, usamos um try, criamos um FileReader chamado marcaleitura, que recebe como parâmetro as informações do arquivo .csv que indicamos na variável arquivoCartas. Criamos a variável do tipo String linha, que receberá as informações das linhas da planilha de cartas.

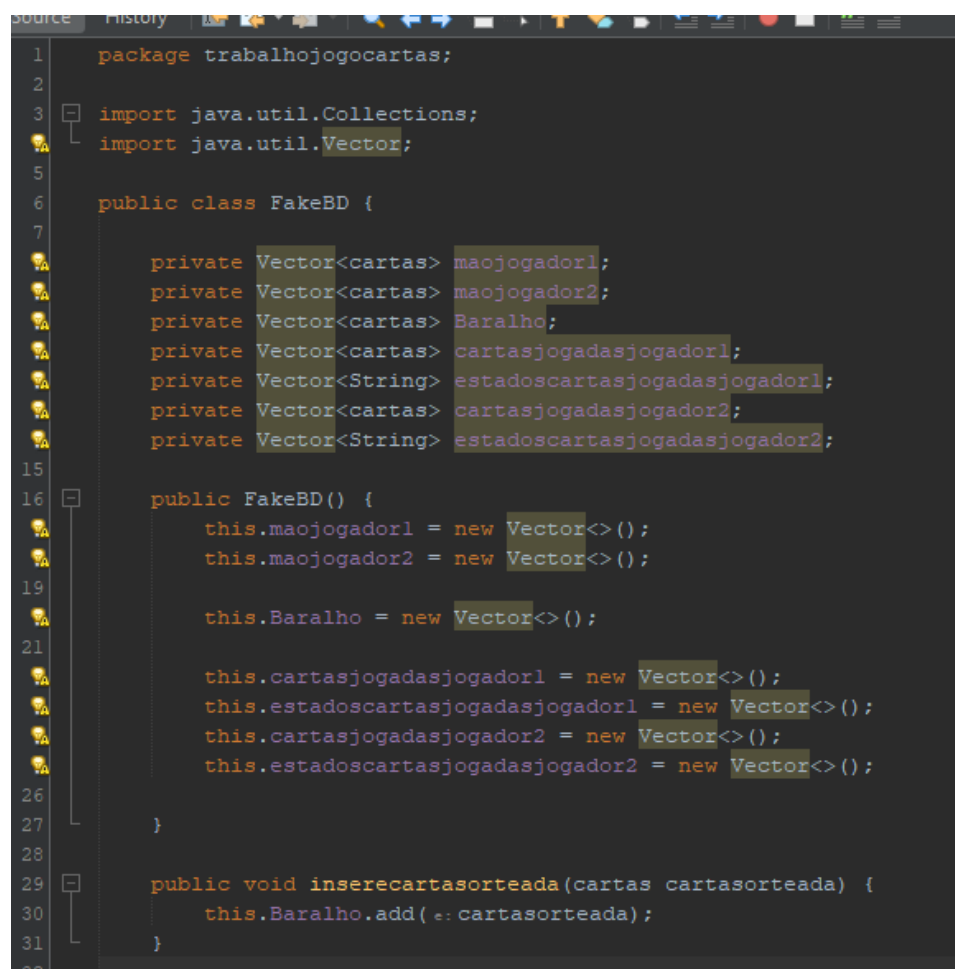
Após isso usamos um do while que repetirá enquanto a linha lida tiver informações, ou seja, a variável não tiver valor nulo, após isso usamos o comando readLine do buffer e salvamos na variável linha, se a linha lida for diferente de nula,



ou seja, tiver informações, usamos o comando split, usando “;” como ponto de divisão e salvamos as informações no vetor dadoslinha.

Após isso, usamos o parseInt nos índices da planilha onde ficam os valores do ataque e da defesa, respectivamente, para transformar os valores de ataque e defesa em números inteiros e salvamos nas variáveis do tipo inteiro “ataque” e “defesa”, respectivamente.

Criamos a variável cartas do tipo Cartas e salvamos nela as informações obtidas na leitura da linha, em seguida invocamos o método inserenovacarta da classe FakeBD(por meio do “meuBanco” que declaramos anteriormente), levando a variável cartas como parâmetro.



```

1  package trabalhojogocartas;
2
3  import java.util.Collections;
4  import java.util.Vector;
5
6  public class FakeBD {
7
8      private Vector<cartas> maojogador1;
9      private Vector<cartas> maojogador2;
10     private Vector<cartas> Baralho;
11     private Vector<cartas> cartasjogadasjogador1;
12     private Vector<String> estadoscartasjogadasjogador1;
13     private Vector<cartas> cartasjogadasjogador2;
14     private Vector<String> estadoscartasjogadasjogador2;
15
16     public FakeBD() {
17         this.maojogador1 = new Vector<>();
18         this.maojogador2 = new Vector<>();
19
20         this.Baralho = new Vector<>();
21
22         this.cartasjogadasjogador1 = new Vector<>();
23         this.estadoscartasjogadasjogador1 = new Vector<>();
24         this.cartasjogadasjogador2 = new Vector<>();
25         this.estadoscartasjogadasjogador2 = new Vector<>();
26     }
27
28     public void inserenovacarta(cartas cartasorteada) {
29         this.Baralho.add(e: cartasorteada);
30     }
31
32

```

Primeiramente, vamos explicar os componentes iniciais da classe FakeBD e depois o método que foi invocado:

Primeiramente temos os imports que serão necessários na classe, depois definimos os vetores dinâmicos onde serão salvas as cartas das mãos dos dois jogadores, o vetor das cartas do baralho de onde os jogadores vão receber as cartas,

os vetores onde serão salvas as cartas que os jogadores jogaram/colocaram no campo e os vetores onde serão salvos os estados das cartas jogadas pelos jogadores (ataque ou defesa). Em seguida criamos os vetores mencionados acima.

Agora podemos falar do método que foi invocado:

No método `inserecartasorteada` adicionamos a carta que foi passada como parâmetro é adicionada no vetor `Baralho`.

Voltando para a classe `Jogatina`:

```
82         } while (linha != null);
83
84         meuBanco.removelinhaembranco();
85
```

Depois de todas as linhas do arquivo `.csv` ser lidas, invocamos o método `removelinhaembranco` da classe `FakeBD`:

Como podemos observar, no arquivo `.csv` disponibilizado tem uma linha em branco, então o método `removelinhaembranco` terá a função de apagar essa linha em branco que foi lida, pois se não for apagada, pode acontecer o caso em que um jogador recebe uma carta “vazia”, deixando-o em desvantagem em relação ao outro jogador.

36	Escudo dos Ancestrais	Um escudo ancestral que aumenta a defesa do monstro equipado.	0	1000	equipamento
37			0	0	equipamento
38	Espada Lendária	Uma espada lendária que aumenta o ataque do monstro equipado.	500	0	monstro

```
32
33 public void removelinhaembranco() {
34
35     this.Baralho.remove(index: 36);
36
37 }
38
```

Voltando para a classe `Jogatina`:

Definimos a variável `jogador`, inicializamos ela com 1 e vamos montar as mãos dos jogadores, começamos pela mão do jogador 1, chamando o método `embaralhar` da classe `FakeBD`, levando as variáveis `jogador` e `indice` como parâmetros, após isso atualizamos a variável `jogador` para 2 e formamos a mão do jogador 2, também chamando o método `embaralhar` da classe `FakeBD`, levando as variáveis `jogador` e `indice` como parâmetros:

```
85
86     int jogador = 1;
87
88     maojogador1 = meuBanco.embaralhar(jogador, indice);
89
90     jogador = 2;
91
92     maojogador2 = meuBanco.embaralhar(jogador, indice);
93
```

O método `embaralhar` retorna um vetor dinâmico do formado por valores do tipo `cartas`, caso o jogador seja igual à 1, usamos o comando `Collections.shuffle` para embaralhar o vetor `Baralho`, usamos um `for` em que o `i` vai de 0 até `indice-1` (por causa do `<`), e como foi definido na classe `TrabalhoJogoCartas` que `indice=5`, são sorteadas as 5 cartas da mão inicial do jogador 1, a cada repetição a variável `cartasorteada` do tipo `cartas` recebe a carta do baralho do índice `i`, adicionamos a carta salva na variável `cartasorteada` à mão do jogador 1 e após isso removemos a carta do baralho, pois

ela foi adicionada à mão do jogador 1. Após isso o método retorna a mão do jogador 1. Caso o jogador seja igual à 2, usamos os mesmos comandos usados no caso do jogador 1, só que adicionamos as cartas sorteadas à mão do jogador 2 e depois da execução do for o método retorna a mão do jogador 2. Caso o jogador não for diferente de 1 ou 2, o método retorna nulo

```

39 public Vector<cartas> embaralhar(int jogador, int indice) {
40
41     if (jogador == 1) {
42
43         Collections.shuffle(list:Baralho);
44         for (int i = 0; i < indice; i++) {
45             cartas cartassorteada = Baralho.get(index:i);
46
47             maojogador1.add(*:cartassorteada);
48             Baralho.remove(index:i);
49         }
50
51         return maojogador1;
52     }
53
54     if (jogador == 2) {
55         Collections.shuffle(list:Baralho);
56         for (int i = 0; i < indice; i++) {
57             cartas cartassorteada = Baralho.get(index:i);
58
59             maojogador2.add(*:cartassorteada);
60             Baralho.remove(index:i);
61         }
62         return maojogador2;
63     }
64     return null;
65 }
66

```

Voltando para a classe Jogatina:

```

92 maojogador12 = meubanco.embaralhar(jogador, indice);
93
94     } catch (FileNotFoundException ex) {
95         System.err.println(*:"O arquivo nao existe no caminho especificado");
96
97     } catch (IOException ex) {
98         System.out.println(*:"Problema na leitura dos dados do arquivo");
99     }
100
101     } else {
102
103         System.err.println(*:"O arquivo nao existe no caminho especificado");
104     }
105
106     jogo(maojogador1, maojogador2);
107
108 }
109

```

Aqui temos os catch's do try que usamos na linha 58, caso o link do arquivo esteja errado (FileNotFoundException ex) é mostrado uma mensagem para o usuário, caso tenha algum outro erro (IOException ex) também é mostrado uma

mensagem para o usuário. Em seguida temos o else do if da linha 56, ou seja, caso o arquivo não exista no link indicado, é mostrado uma mensagem para o usuário.

Após isso invocamos o método jogo, levando as mãos dos jogadores como parâmetro.

```

109
110 public static void jogo(Vector<cartas> maojogador1, Vector<cartas> maojogador2) {
111     Scanner entrada = new Scanner(System.in);
112
113     System.out.println("Mãos iniciais já sorteadas e entregues aos jogadores");
114
115     if (pontossjogador1 == 0 && pontossjogador2 == 0) {
116         pontossjogador1 = 10000;
117         pontossjogador2 = 10000;
118     }
119
120     if (controle == 0) {
121         pontuacao.add(10000);
122         pontuacao.add(10000);
123         controle = 1;
124     }
125
126     Vector<cartas> cartasjogadasjogador1 = new Vector<>();
127     Vector<String> estadoscartasjogadasjogador1 = new Vector<>();
128     Vector<cartas> cartasjogadasjogador2 = new Vector<>();
129     Vector<String> estadoscartasjogadasjogador2 = new Vector<>();
130     Vector<Integer> situacaocartasjogadasjogador1 = new Vector<>();
131     Vector<Integer> situacaocartasjogadasjogador2 = new Vector<>();
132
133
134     int x = -1;
135     int jogador = 1;
136     Vector<Integer> colocouequipamento = new Vector<>();
137     colocouequipamento.add(0);
138     colocouequipamento.add(0);
139

```

No método jogo declaramos o Scanner para que o usuário possa informar dados, depois disso é mostrado para os usuários uma mensagem avisando que as mãos iniciais dos jogadores já foram entregues para os jogadores, após isso verificamos se a pontuação dos jogadores não foi atualizada, se sim, atualizamos a pontuação dos jogadores com os 10000 pontos iniciais dos jogadores, após isso verificamos se a variável controle ainda não foi atualizada, se sim adicionamos a pontuação inicial dos jogadores ao vetor pontuacao e atualizamos a variável controle.

Após isso criamos os vetores dinâmicos onde ficarão salvos as cartas jogadas pelos jogadores, os estados das cartas jogadas pelos jogadores (ataque ou defesa) e as situações das cartas jogadas pelos jogadores (se a carta já atacou ou não em uma rodada, 0-não atacou, 1-atacou). Criamos a variável x, que representará a opção escolhida pelos jogadores, criamos a variável que armazena o número do jogador. Criamos o vetor dinâmico colocouequipamento, que será

responsável em armazenar se o jogador já colocou um equipamento em uma rodada, e adicionamos 2 zeros representando os 2 jogadores (0-não colocou equipamento, 1-colocou equipamento).

```

139
140     while (pontossjogador1 > 0 && pontossjogador2 > 0) {
141
142         if (x != 4) {
143             while (x != 4) {
144                 if (x == 4) {
145
146                     break;
147                 }
148
149                 if (pontossjogador1 > 0 && pontossjogador2 > 0) {
150                     System.out.println(x: "");
151                     System.out.println("Rodada: " + rodada);
152
153                     System.out.println(x: "Pontuação:");
154                     System.out.println("Jogador 1: " + pontuacao.get(index:0));
155                     System.out.println("Jogador 2: " + pontuacao.get(index:1));
156                     System.out.println(x: "");
157
158                     System.out.println("Jogador " + jogador + ", você quer:");
159                     System.out.println(x: "-1-ver cartas nas mãos: ");
160                     System.out.println(x: "0-Descartar uma carta da mão: ");
161                     System.out.println(x: "1-posicionar monstro:");
162                     System.out.println(x: "2-adicionar equipamento:");
163                     System.out.println(x: "3-mostrar area de cartas:");
164                     System.out.println(x: "4-passar turno:");
165                     System.out.println(x: "5-ver detalhes sobre as cartas nas mãos: ");
166

```

Enquanto as pontuações dos jogadores forem maiores que 0, ou seja ninguém perder, o jogo repete/continua. Se x for diferente de 4, ou seja, o jogador não passe a vez, entramos em outro laço de repetição while que enquanto x for diferente

de 4, repete. Se x for igual à 4, ou seja, o jogador informar 4 (passar a vez), usamos o break para finalizar o laço de repetição da linha 143.

Se as pontuações dos jogadores forem maiores que 0, o menu com as pontuações dos jogadores é mostrado acessando os índices dos jogadores 1 e 2 (nas posições 0 e 1 do vetor pontuação, respectivamente), após isso é mostrado o menu de opções de interações.

```

166
167
168         if (jogador == 1) {
169             if (cartasjogadasjogador1.isEmpty() == false) {
170                 System.out.println(x: "6-alterar estado de cartas:");
171             }
172         }
173
174         if (jogador == 2) {
175             if (cartasjogadasjogador2.isEmpty() == false) {
176                 System.out.println(x: "6-alterar estado de cartas:");
177             }
178         }
179
180
181         if (jogador == 1 && rodada > 2) {
182             if (cartasjogadasjogador1.isEmpty() == false) {
183
184                 System.out.println(x: "7-atacar:");
185             }
186         }
187
188         if (jogador == 2 && rodada > 2) {
189             if (cartasjogadasjogador2.isEmpty() == false) {
190
191                 System.out.println(x: "7-atacar:");
192             }
193         }

```

Caso o jogador (1 ou 2) tenha alguma carta do tipo monstro posicionada (o vetor de cartas jogada pelo usuário não está vazio) a opção de alterar o estado da(s) carta(s) posicionada(s) é mostrada para o(s) usuário(s) que cumpre(m) as

condições. E caso o jogador (1 ou 2) tenha alguma carta do tipo monstro posicionada e caso já houve duas rodadas no jogo, a opção de atacar o oponente aparece para o(s) usuário(s) que cumpre(m) as condições.

```

195         x = entrada.nextInt();
196
197         if (x < -1 || x > 7) {
198             System.out.println(x: "opção selecionada não disponível");
199         }
200

```

O jogador da rodada informa a opção desejada e a variável x recebe a escolha. Se a opção informada pelo usuário não corresponder com as opções disponíveis, é mostrada uma mensagem para o jogador informando que a opção selecionada não é uma opção disponível.

```

226
227         if (x == -1) {
228
229             mostramaos(maojogador1, maojogador2, jogador);
230
231         }

```

**Caso o jogador escolha ver as cartas da sua mão:** o método mostramaos é invocado, levando as mãos dos jogadores e o número do jogador como parâmetros.

```

19
20 public static void mostramaos(Vector<cartas> maojogador1, Vector<cartas> maojogador2, int jogador) {
21     if (jogador == 1) {
22         System.out.println(x: "Mão do jogador 1: ");
23
24         for (int i = 0; i < maojogador1.size(); i++) {
25             System.out.print(obj: maojogador1.get(index: i));
26
27             if (i < maojogador1.size() - 1) {
28                 System.out.print(s: ", ");
29             }
30         }
31         System.out.println(x: "");
32     }
33
34
35     if (jogador == 2) {
36         System.out.println(x: "Mão do jogador 2:");
37         for (int i = 0; i < maojogador2.size(); i++) {
38             System.out.print(obj: maojogador2.get(index: i));
39
40             if (i < maojogador2.size() - 1) {
41                 System.out.print(s: ", ");
42             }
43         }
44         System.out.println(x: "");
45     }
46 }
47
48

```

É mostrada uma mensagem falando de qual jogador é a mão que está sendo mostrada, após isso é usado um for em que o índice vai de 0 até a quantidade de cartas presentes na mão do jogador que escolheu a opção-1 (por causa do <), a cada repetição o nome de uma das cartas da mão do jogador que escolheu a opção é mostrada e em seguida é mostrada uma vírgula para separar os nomes das cartas (exceto no último nome), e isso se repete até todas as cartas da mão do jogador



que escolheu a opção sejam mostradas. E por fim é usado um `System.out.println("");` para dar uma quebra de linha para deixar o texto mais organizado.

```

200
201     if (x == 0) {
202         System.out.println(x: "Digite a carta que você quer descartar");
203         int cartadescartada = entrada.nextInt();
204         cartadescartada--;
205
206         if (jogador == 1 && cartadescartada >= 0 && cartadescartada < maojogador1.size()) {
207             System.out.println("O jogador 1 descartou a carta " + maojogador1.get(index: cartadescartada));
208
209             meuBaralho.inserircartasorteada( cartasorteada: maojogador1.get(index: cartadescartada));
210             maojogador1.remove(index: cartadescartada);
211
212         } else {
213             if ((jogador == 1 && cartadescartada < 0) || (jogador == 1 && cartadescartada >= maojogador1.size())) {
214                 System.out.println(x: "Índice não disponível");
215             }
216         }
217
218         if (jogador == 2 && cartadescartada >= 0 && cartadescartada < maojogador2.size()) {
219             System.out.println("O jogador 2 descartou a carta " + maojogador2.get(index: cartadescartada));
220
221             meuBaralho.inserircartasorteada( cartasorteada: maojogador2.get(index: cartadescartada));
222             maojogador2.remove(index: cartadescartada);
223
224         } else {
225             if ((jogador == 2 && cartadescartada < 0) || (jogador == 2 && cartadescartada >= maojogador2.size())) {
226                 System.out.println(x: "Índice não disponível");
227             }
228         }
229     }
230
231 }

```

### Caso o jogador escolha descartar uma carta da mão:

É mostrada uma mensagem pedindo para o jogador informar a carta que ele quer destacar e o valor informado é salvo na variável `cartadescartada` e o valor de `cartadescartada` é subtraído em 1, já que os índices de vetores começam em 0. Se o jogador for igual à 1 e o índice informado for válido (for igual ao índice de uma carta presente à mão do jogador 1) é mostrada uma mensagem avisando o nome da carta que o jogador 1 descartou, a carta é retornada para o baralho e a carta é removida da mão do jogador 1. Caso o índice não for válido, uma mensagem é mostrada para o usuário.

Se o jogador for igual à 2 e o índice informado for válido (for igual ao índice de uma carta presente à mão do jogador 2) é mostrada uma mensagem avisando o nome da carta que o jogador 2 descartou, a carta é retornada para o baralho e a carta é removida da mão do jogador 2. Caso o índice não for válido, uma mensagem é mostrada para o usuário.

### Caso o jogador escolha posicionar uma carta do tipo monstro:

```

Source History
386     if (x == 1) {
387         String estado = null;
388         int q = 0;
389
390         if (jogador == 1 && cartasjogadasjogador1.size() < 5) {
391
392             System.out.println(x: "Digite a posição da carta na mão");
393
394             q = entrada.nextInt();
395             q--;
396
397             if (q >= 0 && q <= maojogador1.size() - 1 && maojogador1.get(index: q).getTipo().equalsIgnoreCase(anotherString: "monstro")) {
398                 System.out.println(x: "Você quer colocar a carta no modo ataque ou defesa?");
399                 estado = entrada.next();
400
401                 if (estado.equalsIgnoreCase(anotherString: "ataque") || estado.equalsIgnoreCase(anotherString: "defesa")) {
402                     cartasjogadasjogador1 = meuBanco.adicionaraocampo(q, jogador, maojogador1);
403

```

Declaramos a variável estado do tipo String, que armazenará temporariamente o estado (ataque ou defesa) da carta selecionada, declaramos a variável q, que será usada para armazenar a i índice da carta informado pelo usuário. Se o jogador for igual à 1 e o jogador não tiver 5 monstros posicionados, o posicionamento continua, aí uma mensagem é mostrada para o usuário solicitando que ele informe a posição da carta na sua mão e o valor informado é salvo na variável q e o valor de q é subtraído em 1, já que os índices de vetores começam em 0. Se o valor informado for válido e o índice for de uma carta do tipo “monstro” é mostrado uma mensagem perguntando o estado da carta e a informação informada é salva na variável estado, se o estado informado for válido (ataque ou defesa) a carta informada é adicionada ao vetor de cartas jogadas do jogador 1 por meio do método adicionaraocampo da classe FakeBD.

```

67
68     public Vector<cartas> adicionaraocampo(int q, int jogador, Vector<cartas> maojogador1) {
69         if (jogador == 1) {
70
71             cartasjogadasjogador1.add(e: maojogador1.get(index: q));
72
73             return cartasjogadasjogador1;
74         }
75
76         if (jogador == 2) {
77             cartasjogadasjogador2.add(e: maojogador1.get(index: q));
78
79             return cartasjogadasjogador2;
80         }
81
82         return null;
83     }
84

```

O método adicionaraocampo retorna um vetor dinâmico do formado por valores do tipo cartas, se o jogador for igual à 1, no vetor cartasjogadasjogador1 é adicionado a carta informada pelo jogador 1 e o vetor cartasjogadasjogador1 é

retornado, se o jogador for igual à 2, no vetor `cartasjogadasjogador2` é adicionado a carta informada pelo jogador 2 e o vetor `cartasjogadasjogador2` é retornado, e caso o jogador seja diferente de 1 e de 2 é retornado nulo.

Voltando para a classe Jogatina:

O vetor dos estados das cartas jogadas pelo jogador 1 é atualizado com o estado da carta informada, por meio do método `adicionaestado` da classe `FakeBD`:

```
402         cartasjogadasjogador1 = meuBanco.adicionaCartaCampo(q, jogador, maojogador1);
403
404         estadocartasjogadasjogador1 = meuBanco.adicionaestado(estado, jogador);
405
```

O método `adicionaestado` retorna um vetor dinâmico do formado por valores do tipo `String`, se o jogador for igual à 1, no vetor `estadocartasjogadasjogador1` é adicionado o estado da carta informada pelo jogador 1 e o vetor `estadocartasjogadasjogador1` é retornado, se o jogador for igual à 2, no vetor `estadocartasjogadasjogador2` é adicionado

o estado da carta informada pelo jogador 2 e o vetor `estadocartasjogadasjogador2` é retornado, e caso o jogador seja diferente de 1 e de 2 é retornado nulo.

```

84 public Vector<String> adicionarestado(String estado, int jogador) {
86     if (jogador == 1) {
87
88         estadocartasjogadasjogador1.add(e: estado);
89
90         return estadocartasjogadasjogador1;
91     }
92
93     if (jogador == 2) {
94         estadocartasjogadasjogador2.add(e: estado);
95
96         return estadocartasjogadasjogador2;
97     }
98     return null;
99 }
100

```

Voltando para a classe Jogatina:

```

404 estadocartasjogadasjogador1 = maoBanco.adicionarestado(estado, jogador);
405
406 System.out.println("O jogador 1 colocou a carta " + maojogador1.get(index:q) + " na posição de " + estado);
407 maojogador1.remove(index:q);
408
409 if (estadocartasjogadasjogador1.get(cartasjogadasjogador1.size() - 1).equalsIgnoreCase(anotherString: "ataque")) {
410     situacaocartasjogadasjogador1.add(e: 0);
411 }
412
413 if (estadocartasjogadasjogador1.get(cartasjogadasjogador1.size() - 1).equalsIgnoreCase(anotherString: "defesa")) {
414     situacaocartasjogadasjogador1.add(e: 1);
415 }
416 } else {
417     System.out.println(x: "O modo escolhido não é válido");
418 }
419
420 } else {
421
422     if (q < 0 || q > maojogador1.size() - 1) {
423         System.out.println(x: "O índice escolhido não é válido");
424     } else {
425         if (maojogador1.get(index:q).getTipo().equalsIgnoreCase(anotherString: "equipamento")) {
426             System.out.println(x: "A carta informada não é uma carta do tipo monstro");
427         }
428     }
429 }

```

É mostrado para o usuário que o monstro informado foi adicionado e essa carta é removida da mão do jogador. Se o

estado da carta for de ataque, no vetor `situacaocartasjogadasjogador1` é adicionado o valor 0 (pode atacar) e se o estado da carta for de defesa, no vetor `situacaocartasjogadasjogador1` é adicionado o valor 1 (não pode atacar).

Caso o modo escolhido não seja nem de ataque nem de defesa é mostrado para o usuário uma mensagem o avisando que o modo escolhido é inválido. Se o índice escolhido para a carta for inválido é mostrado uma mensagem para o usuário e se a carta escolhida for do tipo equipamento é o programa avisa pra ele que a carta informada não é do tipo monstro.

```

429     }
430 } else {
431     if (jogador == 1 && cartasjogadasjogador1.size() >= 5) {
432         System.out.println("Não foi possível posicionar o monstro selecionado pois há 5 monstros posicionados pelo jogador " + jogador);
433     }
434 }
435 }
436 }
437

```

Se o jogador 1 já tiver colocado 5 cartas do tipo monstro na área de cartas, uma mensagem o informando disso é mostrada.

```

438 if (jogador == 2 && cartasjogadasjogador2.size() < 5) {
439
440     System.out.println("Digite a posição da carta na mão");
441
442     q = entrada.nextInt();
443     q--;
444     if (q >= 0 && q <= maojogador2.size() - 1 && maojogador2.get(index:q).getTipo().equalsIgnoreCase("monstro")) {
445
446         System.out.println("Você quer colocar a carta no modo ataque ou defesa?");
447         estado = entrada.next();
448         if (estado.equalsIgnoreCase("ataque") || estado.equalsIgnoreCase("defesa")) {
449             cartasjogadasjogador2 = meuBanco.adicionaraocampo(q, jogador, maojogador1:maojogador2);
450
451             estadoscartasjogadasjogador2 = meuBanco.adicionarestado(estado, jogador);
452             System.out.println("O jogador 2 colocou a carta " + maojogador2.get(index:q) + " na posição de " + estado);
453             maojogador2.remove(index:q);
454
455             if (estadoscartasjogadasjogador2.get(cartasjogadasjogador2.size() - 1).equalsIgnoreCase("ataque")) {
456                 situacaocartasjogadasjogador2.add(e:0);
457             }
458
459             if (estadoscartasjogadasjogador2.get(cartasjogadasjogador2.size() - 1).equalsIgnoreCase("defesa")) {
460                 situacaocartasjogadasjogador2.add(e:1);
461             }
462
463         } else {
464             System.out.println("O modo escolhido não é válido");
465         }
466     }
467 }
468

```

Seguindo a mesma lógica usada no caso o jogador seja 1, caso o jogador seja igual à 2 e o jogador não tiver 5 monstros posicionados, o posicionamento continua, aí uma mensagem é mostrada para o usuário solicitando que ele informe a posição da carta na sua mão e o valor informado é salvo na variável `q` e o valor de `q` é subtraído em 1, já que os índices de vetores começam em 0. Se o valor informado for válido e o índice for de uma carta do tipo “monstro” é mostrado uma mensagem perguntando o estado da carta e a informação informada é salva na variável `estado`, se o estado informado for válido (ataque ou defesa) a carta informada é adicionada ao vetor de cartas jogadas do jogador 2 por meio do método `adicionaaocampo` da classe `FakeBD`.

O vetor dos estados das cartas jogadas pelo jogador 2 é atualizado com o estado da carta informada, por meio do método `adicionaestado` da classe `FakeBD`. É mostrado para o usuário que o monstro informado foi adicionado e essa carta é

removida da mão do jogador. Se o estado da carta for de ataque, no vetor `situacaocartasjogadasjogador2` é adicionado o valor 0 (pode atacar) e se o estado da carta for de defesa, no vetor `situacaocartasjogadasjogador2` é adicionado o valor 1 (não pode atacar).

Caso o modo escolhido não seja nem de ataque nem de defesa é mostrado para o usuário uma mensagem o avisando que o modo escolhido é inválido.

```

465     }
466     } else {
467
468         if (q < 0 || q > maojogador2.size() - 1) {
469             System.out.println(x: "O indice escolhido não é válido");
470         } else {
471             if (maojogador2.get(index: q).getTipo().equalsIgnoreCase("equipamento")) {
472                 System.out.println(x: "A carta informada não é uma carta do tipo monstro");
473             }
474         }
475     }
476
477     } else {
478
479         if (jogador == 2 && cartasjogadasjogador2.size() >= 5) {
480             System.out.println("Não foi possível posicionar o monstro selecionado pois há 5 monstros posicionados pelo jogador " + jogador);
481         }
482     }
483 }
484 }
485

```

Se o índice escolhido para a carta for inválido é mostrado uma mensagem para o usuário e se a carta escolhida for do tipo equipamento é o programa avisa pra ele que a carta informada não é do tipo monstro. Se o jogador 2 já tiver colocado 5 cartas do tipo monstro na área de cartas, uma mensagem o informando disso é mostrada.

### Caso o jogador escolha adicionar equipamento à uma carta:

```

486     if (x == 2) {
487
488         if (jogador == 1) {
489             if (cartasjogadasjogador1.isEmpty() == false && colocouequipamento.get(index: 0) == 0) {
490                 System.out.println(x: "Qual a posicao do equipamento na mao?");
491                 int posicaooequi = entrada.nextInt();
492                 posicaooequi--;
493

```

Se o jogador for igual à 1, tiver alguma carta do tipo monstro adicionada ao campo de cartas e o jogador ainda não adicionou nenhuma carta do tipo equipamento na rodada, o programa pergunta o índice do equipamento para o jogador

e salva o valor informado na variável `posicao EQUI`, e o valor de `posicao EQUI` é subtraído em 1, já que os índices de vetores começam em 0.

```

494     if (posicao EQUI >= 0 && posicao EQUI <= maoJogador1.size() - 1 && maoJogador1.get(index:posicao EQUI).getTipo().equalsIgnoreCase("equipamento"))
495
496         System.out.println("Em qual carta você quer adicionar o equipamento?");
497         int EQUI = entrada.nextInt();
498         EQUI--;
499
500         if (EQUI >= 0 && EQUI <= cartasJogadasJogador1.size() - 1) {
501             int novoAtaque = cartasJogadasJogador1.get(index:EQUI).getAtaque() + maoJogador1.get(index:posicao EQUI).getAtaque();
502             int novaDefesa = cartasJogadasJogador1.get(index:EQUI).getDefesa() + maoJogador1.get(index:posicao EQUI).getDefesa();
503
504             cartasJogadasJogador1.get(index:EQUI).setAtaque(ataque:novoAtaque);
505             cartasJogadasJogador1.get(index:EQUI).setDefesa(defesa:novaDefesa);
506             System.out.println("O equipamento " + maoJogador1.get(index:posicao EQUI) + " foi adicionado a carta " + cartasJogadasJogador1.get(index:EQUI));
507             System.out.println("");
508             maoJogador1.remove(index:posicao EQUI);
509
510             coloqueEquipamento.set(index:0, element:1);
511
512         } else {
513             System.out.println("O índice escolhido não é válido");
514         }
515     } else {
516         if (posicao EQUI < 0 || posicao EQUI > maoJogador1.size() - 1) {
517             System.out.println("O índice escolhido não é válido");
518         } else {
519             if (maoJogador1.get(index:posicao EQUI).getTipo().equalsIgnoreCase("monstro")) {
520                 System.out.println("A carta informada não é uma carta do tipo equipamento");
521             }
522         }
523     }
524 }

```

Se índice informado for válido e a carta do índice informado for uma carta do tipo equipamento, é perguntado para o jogador a carta do campo de cartas à qual ele quer adicionar o equipamento e o valor informado é salvo na variável `EQUI` e o valor de `EQUI` é subtraído em 1, já que os índices de vetores começam em 0, se índice informado também for válido criamos as variáveis `novoAtaque` e `novaDefesa`, que receberão, respectivamente, os valores do ataque e defesa da carta selecionada do campo de cartas e do equipamento selecionado pelo jogador. Em seguida usamos o comando `set` para atualizar o ataque e a defesa da carta selecionada do vetor de cartas jogadas pelo jogador.

Em seguida removemos a carta do tipo equipamento da mão do jogador e atualizamos o vetor `coloqueEquipamento` de 0 para 1 no índice 0.

Se o índice informado para a carta que vai receber o equipamento for inválido, é mostrado uma mensagem para o jogador.

Se o índice informado do equipamento for inválido também é mostrado uma mensagem para o jogador

e se o índice carta do tipo equipamento indicada pelo jogador for uma carta do tipo monstro, é mostrado uma mensagem informando para o jogador que a carta indicada deveria ser do tipo equipamento.

```

524
525         } else {
526             if (colocouequipamento.get(index:0) == 1) {
527                 System.out.println(x: "O jogador 1 já usou uma carta do tipo equipamento nessa rodada");
528             }
529         }
530     }
531 }

```

Se o jogador já tiver colocado uma carta equipamento na rodada, o programa mostra uma mensagem para o jogador avisando que ele não poderá usar mais cartas do tipo equipamento nessa rodada.

```

531         } else {
532             if (cartasjogadasjogador2.isEmpty() == false && colocouequipamento.get(index:1) == 0) {
533                 System.out.println(x: "Qual a posicao do equipamento na mao?");
534                 int posicaooequi = entrada.nextInt();
535                 posicaooequi--;
536             }

```

Seguindo a mesma lógica usada no caso o jogador seja 1, caso o jogador não seja igual à 1, ou seja, seja igual à 2 e ele ainda não usou uma carta do tipo equipamento nessa rodada, o programa mostra uma mensagem solicitando ao jogador que informe o índice do equipamento que ele deseja usar e salva o valor informado na variável `posicaooequi` e o valor de `posicaooequi` é subtraído em 1, já que os índices de vetores começam em 0.

```

537         if (posicaooequi >= 0 && posicaooequi <= maojogador2.size() - 1 && maojogador2.get(index:posicaooequi).getTipo().equalsIgnoreCase("equipamento")) {
538             System.out.println(x: "Em qual carta você quer adicionar o equipamento?");
539             int equi = entrada.nextInt();
540             equi--;
541             if (equi >= 0 && equi <= cartasjogadasjogador2.size() - 1) {
542                 int novoataque = cartasjogadasjogador2.get(index:equi).getAtaque() + maojogador2.get(index:posicaooequi).getAtaque();
543                 int novadefesa = cartasjogadasjogador2.get(index:equi).getDefesa() + maojogador2.get(index:posicaooequi).getDefesa();
544                 cartasjogadasjogador2.get(index:equi).setAtaque(ataque: novoataque);
545                 cartasjogadasjogador2.get(index:equi).setDefesa(defesa: novadefesa);
546                 System.out.println("O equipamento " + maojogador2.get(index:posicaooequi) + " foi adicionado a carta " + cartasjogadasjogador2.get(index:equi));
547                 System.out.println(x: "");
548                 maojogador2.remove(index:posicaooequi);
549                 colocouequipamento.set(index:1, element:1);
550             } else {
551                 System.out.println(x: "O indice escolhido não é válido");
552             }
553         } else {
554             if (posicaooequi < 0 || posicaooequi > maojogador2.size() - 1) {
555                 System.out.println(x: "O indice escolhido não é válido");
556             } else {
557                 if (maojogador2.get(index:posicaooequi).getTipo().equalsIgnoreCase("monstro")) {
558                     System.out.println(x: "A carta informada não é uma carta do tipo equipamento");
559                 }
560             }
561         }
562     }
563 }
564 }

```

Se índice informado for válido e a carta do índice informado for uma carta do tipo equipamento, é perguntado para o jogador a carta do campo de cartas à qual ele quer adicionar o equipamento e o valor informado é salvo na variável `equi`



e o valor de `equi` é subtraído em 1, já que os índices de vetores começam em 0, se índice informado também for válido criamos as variáveis `novoataque` e `novadefesa`, que receberão, respectivamente, os valores do ataque e defesa da carta selecionada do campo de cartas e do equipamento selecionado pelo jogador. Em seguida usamos o comando `set` para atualizar o ataque e a defesa da carta selecionada do vetor de cartas jogadas pelo jogador 2.

Em seguida removemos a carta do tipo equipamento da mão do jogador 2 e atualizamos o vetor `colocoequipamento` de 0 para 1 no índice 1.

Se o índice informado para a carta que vai receber o equipamento for inválido, é mostrado uma mensagem para o jogador 2. Se o índice informado do equipamento for inválido também é mostrado uma mensagem para o jogador

e se o índice carta do tipo equipamento indicada pelo jogador for uma carta do tipo monstro, é mostrado uma mensagem informando para o jogador que a carta indicada deveria ser do tipo equipamento.

```

564         }
565     } else {
566         if (coloqueequipamento.get(index:1) == 1) {
567             System.out.println("O jogador 2 já usou uma carta do tipo equipamento nessa rodada");
568         }
569     }
570 }
571
572 }
573
574 }

```

Se o jogador 2 já tiver colocado uma carta equipamento na rodada, o programa mostra uma mensagem para o jogador avisando que ele não poderá usar mais cartas do tipo equipamento nessa rodada.

### Caso o jogador escolha mostrar a área de cartas:

Seguindo o modelo de saída nome, ataque, defesa e estado, as informações das cartas dos dois jogadores são mostradas usando um for para cada jogador, onde à cada repetição acessa os índices das cartas jogadas pelos dois jogadores.

```

574
575     if (x == 3) {
576         System.out.println("Modelo de saída: nome, ataque, defesa, estado");
577         System.out.println("Jogador 1:");
578
579         for (int i = 0; i < cartasjogadasjogador1.size(); i++) {
580             System.out.print(cartasjogadasjogador1.get(index:i).getNome() + "---" + cartasjogadasjogador1.get(index:i).getAtaque() + "---" + cartasjogadasjogador1.get(index:i).getDefesa() + "---" + cartasjogadasjogador1.get(index:i).getEstado());
581             System.out.println("");
582         }
583
584         System.out.println("Jogador 2:");
585         for (int i = 0; i < cartasjogadasjogador2.size(); i++) {
586             System.out.print(cartasjogadasjogador2.get(index:i).getNome() + "---" + cartasjogadasjogador2.get(index:i).getAtaque() + "---" + cartasjogadasjogador2.get(index:i).getDefesa() + "---" + cartasjogadasjogador2.get(index:i).getEstado());
587             System.out.println("");
588         }
589     }
590 }
591
592 }
593
594 }

```

```

578
579     .get(index:i).getAtaque() + "---" + cartasjogadasjogador1.get(index:i).getDefesa() + "---" + estadoscartasjogadasjogador1.get(index:i));
580
581
582
583
584
585
586     .get(index:i).getAtaque() + "---" + cartasjogadasjogador2.get(index:i).getDefesa() + "---" + estadoscartasjogadasjogador2.get(index:i));
587
588
589
590

```

### Caso o jogador escolha passar a vez:

Se o jogador for igual à 1, usamos o set para transformar em 0 o valor do índice 0 do vetor que salva se os jogadores usaram uma carta do tipo equipamento, usamos um for que vai de 0 até o tamanho do vetor que armazena se as cartas do jogador 1 podem atacar e “resetamos” os valores com base nos seus estados, se o estado for de ataque “setamos” como 0, se o estado for de defesa “setamos” como 1.

Se o jogador for igual à 2, usamos o set para transformar em 0 o valor do índice 1 do vetor que salva se os jogadores usaram uma carta do tipo equipamento, usamos um for que vai de 0 até o tamanho do vetor que armazena se as cartas do jogador 2 podem atacar e “resetamos” os valores com base nos seus estados, se o estado for de ataque “setamos” como 0, se o estado for de defesa “setamos” como 1.

E o número da rodada aumenta em 1.

```

338         if (x == 4) {
339             if (jogador == 1) {
340                 colocouequipamento.set(index:0, element:0);
341
342                 for (int i = 0; i < situacaocartasjogadasjogador1.size(); i++) {
343                     if (estadoscartasjogadasjogador1.get(index:i).equalsIgnoreCase("ataque")) {
344                         situacaocartasjogadasjogador1.set(index:i, element:0);
345                     }
346
347                     if (estadoscartasjogadasjogador1.get(index:i).equalsIgnoreCase("defesa")) {
348                         situacaocartasjogadasjogador1.set(index:i, element:1);
349                     }
350                 }
351             }
352
353             if (jogador == 2) {
354                 colocouequipamento.set(index:1, element:0);
355
356                 for (int i = 0; i < situacaocartasjogadasjogador2.size(); i++) {
357                     if (estadoscartasjogadasjogador2.get(index:i).equalsIgnoreCase("ataque")) {
358                         situacaocartasjogadasjogador2.set(index:i, element:0);
359                     }
360
361                     if (estadoscartasjogadasjogador2.get(index:i).equalsIgnoreCase("defesa")) {
362                         situacaocartasjogadasjogador2.set(index:i, element:1);
363                     }
364                 }
365             }
366
367             rodada++;
368         }

```

Criamos o vetor dinâmico vetorcartasorteada do tipo cartas, que receberá a carta que será sorteada para os jogadores quando eles passam seu turno, já que à cada turno os jogadores recebem uma carta. Caso o jogador seja igual à 1 e na mão do jogador 1 tenha menos que 10 cartas invocamos o método “refil” da classe FakeBD e adicionamos a carta sorteada à mão do jogador 1. Caso o jogador seja igual à 2 e na mão do jogador 2 tenha menos que 10 cartas invocamos

o método “refil” da classe FakeBD e adicionamos a carta sorteada à mão do jogador 2. Depois limpamos o vetor `vetorcartasorteada` e usamos o `break` para finalizar o laço de repetição.

```

368      Vector<cartas> vetorcartasorteada = new Vector<>();
370
371      if (jogador == 1 && maojogador1.size() <= 9) {
372          vetorcartasorteada = meuBanco.refil(maojogador1, maojogador2, jogador, pontosjogador1, pontosjogador2);
373          maojogador1.add(e: vetorcartasorteada.get(index:0));
374      }
375
376      if (jogador == 2 && maojogador2.size() <= 9) {
377          vetorcartasorteada = meuBanco.refil(maojogador1, maojogador2, jogador, pontosjogador1, pontosjogador2);
378          maojogador2.add(e: vetorcartasorteada.get(index:0));
379      }
380
381      vetorcartasorteada.clear();
382
383      break;
384  }
385

```

Método “refil” da classe FakeBD:

Criamos o vetor dinâmico `vetorcartasorteada`, se o vetor não estiver vazio, usamos o comando `.clear` para limpá-lo.

Se o jogador é igual à 1, embaralhamos o baralho, sorteamos uma carta e adicionamos ela ao vetor `vetorcartasorteada` e removemos ela do baralho.

Se o baralho estiver vazio, mostramos a pontuação dos jogadores, se o jogador 1 tiver a maior pontuação, o jogador 1 ganha e é mostrado uma mensagem de vitória, se o jogador 2 tiver a maior pontuação, o jogador 2 ganha e é mostrado uma mensagem de vitória.

```

100  public Vector<cartas> refil(Vector<cartas> maojogador1, Vector<cartas> maojogador2, int jogador, int pontosjogador1, int pontosjogador2) {
102
103      Vector<cartas> vetorcartasorteada = new Vector<>();
104
105      if (vetorcartasorteada.isEmpty() == false) {
106          vetorcartasorteada.clear();
107      }
108
109
110      if (jogador == 1) {
111
112          Collections.shuffle(list: Baralho);
113          for (int i = 0; i < 1; i++) {
114              vetorcartasorteada.add(e: Baralho.get(index:i));
115
116              Baralho.remove(index:i);
117
118              if (Baralho.size() == 0) {
119                  System.out.println("Pontuação:");
120                  System.out.println("Jogador 1: " + pontosjogador1);
121                  System.out.println("Jogador 2: " + pontosjogador2);
122                  System.out.println(" ");
123
124                  if (pontosjogador1 > pontosjogador2) {
125                      System.out.println("As cartas do baralho acabaram e o jogador 1 tem maior pontuação, logo o jogador 1 ganhou");
126                  }
127                  if (pontosjogador1 < pontosjogador2) {
128                      System.out.println("As cartas do baralho acabaram e o jogador 2 tem maior pontuação, logo o jogador 2 ganhou");
129                  }

```

Se os dois jogadores tiverem a mesma pontuação, ocorre um empate e uma mensagem de empate é mostrada. E por fim a execução do programa é finalizada.

```
129     }
130     if (pontosjogador1 == pontosjogador2) {
131         System.out.println("As cartas do baralho acabaram e os jogadores possuem a mesma pontuação, logo houve um empate");
132     }
133     System.exit(status: 0);
134 }
135 }
136 }
137 }
138 }
```

Se o jogador é igual à 2, embaralhamos o baralho, sorteamos uma carta e adicionamos ela ao vetor `vetorcartasorteada` e removemos ela do baralho.

Se o baralho estiver vazio, mostramos a pontuação dos jogadores, se o jogador 1 tiver a maior pontuação, o jogador 1 ganha e é mostrado uma mensagem de vitória, se o jogador 2 tiver a maior pontuação, o jogador 2 ganha e é mostrado

uma mensagem de vitória. Se os dois jogadores tiverem a mesma pontuação, ocorre um empate e uma mensagem de empate é mostrada. E por fim a execução do programa é finalizada. E o método retorna o vetor `vetorcartasorteada`.

```

140     if (jogador == 2) {
141         Collections.shuffle(lista:Baralho);
142         for (int i = 0; i < 1; i++) {
143             vetorcartasorteada.add(e:Baralho.get(index:i));
144
145             Baralho.remove(index:i);
146
147             if (Baralho.size() == 0) {
148                 if (pontosjogador1 > pontosjogador2) {
149                     System.out.println("As cartas do baralho acabaram e o jogador 1 tem maior pontuação, logo o jogador 1 ganhou");
150                 }
151                 if (pontosjogador1 < pontosjogador2) {
152                     System.out.println("As cartas do baralho acabaram e o jogador 2 tem maior pontuação, logo o jogador 2 ganhou");
153                 }
154                 if (pontosjogador1 == pontosjogador2) {
155                     System.out.println("As cartas do baralho acabaram e os jogadores possuem a mesma pontuação, logo houve um empate");
156                 }
157                 System.exit(status:0);
158             }
159         }
160     }
161
162     return vetorcartasorteada;
163 }
164
165
166
167

```

Voltando para a classe Jogatina:

Depois da execução do `break` quando o jogador escolhe passar o turno, o número do jogador é trocado, se o jogador era um, passa para o 2 e vice-versa, e o `x` é “resetado” para o valor no qual ele é inicializado.

```

595         if (jogador == 1) {
596             jogador = 2;
597         } else {
598             if (jogador == 2) {
599                 jogador = 1;
600             }
601
602         }
603
604         x = -1;
605
606     }
607
608 }
609
610 }
611

```

**Caso o jogador escolha ver detalhes sobre as cartas na mão:**

Se o jogador escolher ver detalhes sobre as cartas na mão, o método `descricaomao` é invocado, levando as mãos dos jogadores e o número do jogador como parâmetros.

```

236         }
237         if (x == 5) {
238             descricaomao(maojogador1, maojogador2, jogador);
239         }
240     }

```

Se o jogador for igual à 1, mostramos um menu com os nomes, descrições, ataque, defesa e o tipo das cartas que estão na mão do jogador 1.

```

667     public static void descricaomao(Vector<cartas> maojogador1, Vector<cartas> maojogador2, int jogador) {
668
669         if (jogador == 1) {
670             System.out.println(x: "Cartas da mão do jogador 1");
671             System.out.println(x: "-----");
672             for (int i = 0; i < maojogador1.size(); i++) {
673                 System.out.println("nome: " + maojogador1.get(index:i).getNome());
674
675                 System.out.println("Descricao: " + maojogador1.get(index:i).getDescricao());
676
677                 System.out.println("ataque: " + maojogador1.get(index:i).getAtaque());
678
679                 System.out.println("defesa: " + maojogador1.get(index:i).getDefesa());
680
681                 System.out.println("tipo: " + maojogador1.get(index:i).getTipo());
682
683                 System.out.println(x: "");
684             }
685         }
686     }
687 }
688

```

Se o jogador for igual à 2, mostramos um menu com os nomes, descrições, ataque, defesa e o tipo das cartas que estão na mão do jogador 2.

```

688
689         if (jogador == 2) {
690             System.out.println(x: "Cartas da mão do jogador 2");
691             System.out.println(x: "-----");
692             for (int i = 0; i < maojogador2.size(); i++) {
693                 System.out.println("nome: " + maojogador2.get(index:i).getNome());
694
695                 System.out.println("Descricao: " + maojogador2.get(index:i).getDescricao());
696
697                 System.out.println("ataque: " + maojogador2.get(index:i).getAtaque());
698
699                 System.out.println("defesa: " + maojogador2.get(index:i).getDefesa());
700
701                 System.out.println("tipo: " + maojogador2.get(index:i).getTipo());
702
703                 System.out.println(x: "");
704             }
705         }
706     }
707 }
708
709

```

**Caso o jogador escolha alterar o estado de cartas:**

Se o jogador for igual à 1 e ele tiver alguma carta no campo de cartas, alteramos o estado da carta desejada pelo usuário a partir do método `alteraestado` e atualizamos o vetor `estadoscartasjogador1`.

```

241     if (x == 6) {
242
243         if (jogador == 1 && cartasjogadasjogador1.isEmpty() == false) {
244
245             estadoscartasjogadasjogador1 = alteraestado(jogador, cartasjogadasjogador1, estadoscartasjogadasjogador1, situacaocartasjogadasjogador1, cartasjoga
246
247
248
249
244     jogador1, estadoscartasjogadasjogador1, situacaocartasjogadasjogador1, cartasjogadasjogador2, estadoscartasjogadasjogador2, situacaocartasjogadasjogador2);
245
246

```

- Método `alteraestado`:

No método é mostrado uma mensagem para o usuário pedindo para ele informar o índice da carta o qual ele quer mudar o estado e o valor indicado é salvo na variável `indice` e o valor de `indice` é subtraído em 1, já que os índices de vetores começam em 0.

Se o jogador for igual à 1 e se o índice informado for válido, se a carta estiver no modo de ataque e ela já atacou nessa rodada é mostrada uma mensagem para o jogador informando-o que não é possível fazer a mudança. Se a carta estiver no estado de ataque, é mudado para defesa e é mostrado uma mensagem avisando que a troca ocorreu, e a situação da carta é “setada” como 1 (não pode atacar), já que cartas no modo de defesa não podem atacar. Se a carta estiver no estado



de defesa, é mudado para ataque e é mostrado uma mensagem avisando que a troca ocorreu, e a situação da carta é “setada” como 0 (pode atacar), já que cartas no modo de ataque podem atacar.

Se o índice informado não for válido, uma mensagem é mostrada para o usuário o informando disso.

E o método retorna o vetor `estadocartasjogadasjogador1`

```

611 public static Vector<String> alteraestado(int jogador, Vector<cartas> cartasjogadasjogador1, Vector<String> estadocartasjogadasjogador1, Vector<Integer> si
613 Scanner entrada = new Scanner(System.in);
614
615 System.out.println("Digite a carta que você deseja alterar o estado: ");
616 int indice = entrada.nextInt();
617 indice--;
618
619 if (jogador == 1) {
620     if (indice >= 0 && indice <= situacaocartasjogadasjogador1.size() - 1) {
621         if (situacaocartasjogadasjogador1.get(indice) == 1 && estadocartasjogadasjogador1.get(indice).equalsIgnoreCase("ataque"))
622             System.out.println("Não é possível alterar o estado da carta selecionada, pois quando um monstro realiza um ataque, o usuário ficará impe
623         } else {
624             if (estadocartasjogadasjogador1.get(indice).equalsIgnoreCase("ataque")) {
625                 estadocartasjogadasjogador1.set(indice, "defesa");
626
627                 System.out.println("O estado da carta " + cartasjogadasjogador1.get(indice) + " foi mudada do modo de ataque para o de defesa");
628                 situacaocartasjogadasjogador1.set(indice, 1);
629             } else {
630                 estadocartasjogadasjogador1.set(indice, "ataque");
631                 System.out.println("O estado da carta " + cartasjogadasjogador1.get(indice) + " foi mudada do modo de defesa para o de ataque");
632                 situacaocartasjogadasjogador1.set(indice, 0);
633             }
634         }
635     } else {
636         System.out.println("O índice escolhido não é válido");
637     }
638     return estadocartasjogadasjogador1;
639 }

```

```

613 :aocartasjogadasjogador1, Vector<cartas> cartasjogadasjogador2, Vector<String> estadocartasjogadasjogador2, Vector<Integer> situacaocartasjogadasjogador2) {
614
615
616
617
618
619
620
621
622 de alterar o estado do mesmo na rodada corrente. Podendo alterar o estado apenas na sua próxima rodada. ");
623
624
625
626
627
628
629
630
631

```

Se o jogador for igual à 2 e se o índice informado for válido, se a carta estiver no modo de ataque e ela já atacou nessa rodada é mostrada uma mensagem para o jogador informando-o que não é possível fazer a mudança. Se a carta estiver no estado de ataque, é mudado para defesa e é mostrado uma mensagem avisando que a troca ocorreu, e a situação da carta é “setada” como 1 (não pode atacar), já que cartas no modo de defesa não podem atacar. Se a carta estiver no estado de defesa, é mudado para ataque e é mostrado uma mensagem avisando que a troca ocorreu, e a situação da carta é

“setada” como 0 (pode atacar), já que cartas no modo de ataque podem atacar. Se o índice informado não for válido, uma mensagem é mostrada para o usuário o informando disso. E o método retorna o vetor `estadocartasjogadasjogador2`. Caso jogador seja diferente de 1 e de 2, o método retorna nulo.

```

640     if (jogador == 2) {
641         if (indice >= 0 && indice <= situacaocartasjogadasjogador2.size() - 1) {
642             if (situacaocartasjogadasjogador2.get(indice) == 1 && estadocartasjogadasjogador2.get(indice).equalsIgnoreCase("ataque")) {
643                 System.out.println("Não é possível alterar o estado da carta selecionada, pois quando um monstro realiza um ataque, o usuário ficará imp");
644             } else {
645                 if (estadocartasjogadasjogador2.get(indice).equalsIgnoreCase("ataque")) {
646                     estadocartasjogadasjogador2.set(indice, element: "defesa");
647
648                     System.out.println("O estado da carta " + cartasjogadasjogador2.get(indice) + " foi mudada do modo de ataque para o de defesa");
649
650                     situacaocartasjogadasjogador2.set(indice, element: 1);
651                 } else {
652                     estadocartasjogadasjogador2.set(indice, element: "ataque");
653                     System.out.println("O estado da carta " + cartasjogadasjogador2.get(indice) + " foi mudada do modo de defesa para o de ataque");
654                     situacaocartasjogadasjogador2.set(indice, element: 0);
655                 }
656             }
657         } else {
658             System.out.println("O índice escolhido não é válido");
659         }
660         return estadocartasjogadasjogador2;
661     }
662     return null;
663 }
664
665
666
667

```

### Caso o jogador escolha atacar o jogador adversário:

Se o jogador escolher atacar o jogador adversário e a rodada for um número maior que 2, se o jogador for igual à 1 e o jogador 1 ainda tem cartas que podem atacar (tem cartas com situação igual à 0) e invocamos o método ataque da classe Ataque e o resultado é salvo no vetor pontuação:

```

264     if (x == 7 && rodada > 2) {
265         if (jogador == 1) {
266             if (situacaocartasjogadasjogador1.contains(0) == true) {
267                 pontuacao = Ataque.ataque(pontuacao, pontosjogador1, pontosjogador2, jogador, cartasjogadasjogador1, cartasjogadasjogador2, estadocartasjogadasjogador1, cartasjogadasjogador2, estadocartasjogadasjogador1, estadocartasjogadasjogador2, situacaocartasjogadasjogador1, situacaocartasjogadasjogador2);
268             }
269         }
270     }
271 }
272

```

### Método ataque da classe Ataque:

Na classe Ataque temos os imports que serão usados.

Caso o jogador seja igual à 1, declaramos a variável `r`, que receberá o índice que o jogador monstro que ele irá usar no ataque e a variável `foieliminado`, que registrará se a carta que foi usada no ataque foi destruída no ataque. Enquanto o jogador 1 ainda tiver cartas que podem atacar, as variáveis `pontosjogador1` e `pontosjogador2` recebem seus respectivos valores vindos do vetor `pontuacao`, mostramos uma mensagem para o jogador 1 pedindo que informe a carta que quer usar no ataque e o programa salva o valor informado na variável `r` e o valor de `r` é subtraído em 1, já que os índices de

vetores começam em 0. Se o índice informado for válido, pegamos a carta informada pelo usuário e salvamos na variável `monstroescolhido` e se ela for uma carta que está no estado de ataque e pode atacar naquele turno...

```

1 package trabalhojogocartas;
2
3 import java.util.Scanner;
4 import java.util.Vector;
5
6 public class Ataque {
7
8     public static Vector ataque(Vector<Integer> pontuacao, int pontosjogador1, int pontosjogador2, int jogador, Vector<cartas> cartasjogadasjogador1, Vector<cartas> cartasjogadasjogador2) {
9         Scanner entrada = new Scanner(System.in);
10
11         if (jogador == 1) {
12             Integer r = 0;
13             Integer foieliminado = null;
14
15             while (situacaocartasjogadasjogador1.contains(0) == true) {
16
17                 pontosjogador1 = pontuacao.get(index:0);
18                 pontosjogador2 = pontuacao.get(index:1);
19
20                 System.out.println("Digite o monstro que você deseja usar:");
21
22                 r = entrada.nextInt();
23
24                 r--;
25                 if (r >= 0 && r <= situacaocartasjogadasjogador1.size() - 1 && situacaocartasjogadasjogador1.get(index:r) == 0) {
26
27                     cartas monstroescolhido = cartasjogadasjogador1.get(index:r);
28
29                     if (estadoscartasjogadasjogador1.get(index:r).equalsIgnoreCase("ataque")) {
30
31                         if (situacaocartasjogadasjogador1.get(index:r) == 0) {

```

Se o jogador 2 não tiver posicionado nenhuma carta do tipo monstro, o jogador 2 perde os pontos relativos aos pontos de ataque da carta selecionada pelo jogador 1, e o vetor `situacaocartasjogadasjogador1` é atualizado no índice `r` de 0 para 1, indicando que a carta no índice `r` não pode mais atacar no mesmo turno.

```

31 if (situacaocartasjogadasjogador1.get(index:r) == 0) {
32
33     if (cartasjogadasjogador2.isEmpty() == true) {
34
35         if (estadoscartasjogadasjogador1.get(index:r).equalsIgnoreCase("ataque")) {
36
37             pontosjogador2 = pontosjogador2 - monstroescolhido.getAtaque();
38             System.out.println("O jogador 1 atacou com a carta " + monstroescolhido + " e já que o jogador 2 não havia posicionado nenhuma carta, o jogador 2 perde pontos.");
39         }
40         situacaocartasjogadasjogador1.set(index:r, element:1);
41     }
42 }

```

Se o jogador 2 posicionou cartas do tipo monstro, e a carta informada pelo jogador 1 pode atacar, é mostrada uma mensagem perguntando ao jogador 1 qual carta inimiga ele quer atacar, e o valor informado é salvo na variável `g` e o valor de `g` é subtraído em 1, já que os índices de vetores começam em 0. Se o índice informado for válido, inicializamos a variável `foieliminado`.

Se a carta atacada estiver no estado de ataque, é mostrada uma mensagem sobre o ataque, se o ataque do jogador 1 for maior que o ataque do jogador 2 uma mensagem é mostrada que fala sobre o resultado do ataque, criamos a variável

```

43 if (cartasjogadasjogador2.isEmpty() == false) {
44
45     if (situacaocartasjogadasjogador1.get(index:r) == 0) {
46         System.out.println("Digite o monstro inimigo que você deseja atacar:");
47         int g = entrada.nextInt();
48         g--;
49
50         if (g >= 0 && g <= estadoscartasjogadasjogador2.size() - 1) {
51             foieliminado = 0;
52
53             if (estadoscartasjogadasjogador2.get(index:g).equalsIgnoreCase(anotherString:"ataque")) {
54                 System.out.println("O jogador 1 usou a carta " + cartasjogadasjogador1.get(index:r) + " para atacar a carta " + cartasjogadasjogador2.get(index:g));
55
56                 if (cartasjogadasjogador2.get(index:g).getAtaque() < cartasjogadasjogador1.get(index:r).getAtaque()) {
57                     System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 tinha menor valor de ataque que a carta " + cartasjogadasjogador1.get(index:r));
58
59                     int resultado = cartasjogadasjogador1.get(index:r).getAtaque() - cartasjogadasjogador2.get(index:g).getAtaque();
60
61                     cartasjogadasjogador1.get(index:r).setAtaque(ataque:resultado);
62
63                     cartasjogadasjogador2.remove(o: cartasjogadasjogador2.get(index:g));
64                     situacaocartasjogadasjogador2.remove(o: situacaocartasjogadasjogador2.get(index:g));
65                     estadoscartasjogadasjogador2.remove(o: estadoscartasjogadasjogador2.get(index:g));
66
67                     pontosjogador2 = pontosjogador2 - resultado;
68                     System.out.println("E o jogador 2 tomou " + resultado + " de dano");
69
70                     cartasjogadasjogador2.remove(index:g);
71                     estadoscartasjogadasjogador2.remove(index:g);
72                     situacaocartasjogadasjogador2.remove(index:g);
73
74                     System.out.println("A carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 foi destruída.");
75
76                     cartasjogadasjogador2.add(index:g, cartasjogadasjogador2.get(index:g));
77                     estadoscartasjogadasjogador2.add(index:g, estadoscartasjogadasjogador2.get(index:g));
78                     situacaocartasjogadasjogador2.add(index:g, situacaocartasjogadasjogador2.get(index:g));
79
80                     pontosjogador2 = pontosjogador2 + resultado;
81                     System.out.println("O jogador 1 ganhou " + resultado + " pontos.");
82
83                     cartasjogadasjogador1.setAtaque(ataque:pontosjogador1);
84                     cartasjogadasjogador2.setAtaque(ataque:pontosjogador2);
85
86                     System.out.println("Fim do jogo!");
87                 } else {
88                     System.out.println("Carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 não foi atacada.");
89
90                     cartasjogadasjogador2.setAtaque(ataque:pontosjogador2);
91                     pontosjogador2 = pontosjogador2 + resultado;
92                     System.out.println("O jogador 1 ganhou " + resultado + " pontos.");
93
94                     cartasjogadasjogador1.setAtaque(ataque:pontosjogador1);
95                     cartasjogadasjogador2.setAtaque(ataque:pontosjogador2);
96
97                     System.out.println("Fim do jogo!");
98                 }
99             } else {
100                 System.out.println("Estado da carta " + estadoscartasjogadasjogador2.get(index:g) + " não é válido.");
101             }
102         } else {
103             System.out.println("Índice da carta " + index + " fora dos limites da lista de cartas jogadas pelo jogador 2.");
104         }
105     } else {
106         System.out.println("Situação da carta " + situacaocartasjogadasjogador1.get(index:r) + " não é válida.");
107     }
108 }

```

Caso o ataque do jogador 1 for igual aos pontos de ataque da carta do inimigo é mostrado uma mensagem sobre o ataque, a carta do jogador 1 que atacou é removida do vetor de cartas jogadas do jogador 1, de situações das cartas do jogador 1 e de estados das cartas do jogador 1. A variável foi eliminado é atualizada, e a carta do jogador 2 que foi atacada é

removida do vetor de cartas jogadas do jogador 2, de situações das cartas do jogador 2, e de estados das cartas do jogador 2.

```

69     } else {
70         if (cartasjogadasjogador2.get(index:g).getAtaque() == cartasjogadasjogador1.get(index:r).getAtaque()) {
71             System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 tinha o mesmo valor de ataque que a carta " + cartasjogad
72
73             cartasjogadasjogador1.remove(0: cartasjogadasjogador1.get(index:r));
74             situacaocartasjogadasjogador1.remove(0: situacaocartasjogadasjogador1.get(index:r));
75             estadoscartasjogadasjogador1.remove(0: estadoscartasjogadasjogador1.get(index:r));
76
77             foieliminado = 1;
78
79             cartasjogadasjogador2.remove(0: cartasjogadasjogador2.get(index:g));
80             situacaocartasjogadasjogador2.remove(0: situacaocartasjogadasjogador2.get(index:g));
81             estadoscartasjogadasjogador2.remove(0: estadoscartasjogadasjogador2.get(index:g));
82
83             mo valor de ataque que a carta " + cartasjogadasjogador1.get(index:r) + " do jogador 1, as duas cartas foram destruidas e nenhum jogador sofreu danos");
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Se o ataque do jogador 1 for menor que o ataque do jogador 2 uma mensagem é mostrada que fala sobre o resultado do ataque, criamos a variável resultado, e a diferença entre os pontos de ataque é salva nela. A variável foi eliminado é atualizada, “setamos” o novo ataque (como resultado do ataque) da carta atacada do jogador 2 como o valor salvo na variável resultado, a carta destruída do jogador 1 é removida da área de cartas, do vetor de estados e de situação das

cartas do jogador 1 e o jogador 1 perde pontos relativos ao valor salvo na variável resultado e uma mensagem é mostrada, informando quantos pontos o jogador 1 perdeu.

```

82     } else {
83
84         if (cartasjogadasjogador2.get(index:g).getAtaque() > cartasjogadasjogador1.get(index:r).getAtaque()) {
85             System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 tinha maior valor de ataque que a carta " + cartasjog
86             int resultado = cartasjogadasjogador2.get(index:g).getAtaque() - cartasjogadasjogador1.get(index:r).getAtaque();
87
88             cartasjogadasjogador2.get(index:g).setAtaque(ataque: resultado);
89
90             cartasjogadasjogador1.remove(o: cartasjogadasjogador1.get(index:r));
91             situacaocartasjogadasjogador1.remove(o: situacaocartasjogadasjogador1.get(index:r));
92             estadoscartasjogadasjogador1.remove(o: estadoscartasjogadasjogador1.get(index:r));
93
94             foieliminado = 1;
95             pontosjogador1 = pontosjogador1 - resultado;
96             System.out.println("E o jogador 1 tomou " + resultado + " de dano");
97         }
98     }
99
100

```

```

83
84
85     maior valor de ataque que a carta " + cartasjogadasjogador1.get(index:r) + " a carta " + cartasjogadasjogador1.get(index:r) + " do jogador 1 foi destruída");
86     ).getAtaque();
87
88
89
90
91
92
93
94
95

```

Caso a carta atacada do jogador 2 esteja no modo de defesa, uma mensagem falando sobre qual carta foi usada pelo jogador 1 para atacar e qual carta do jogador 2 foi atacada. Se o ataque e da carta usada pelo jogador 1 for diferente da defesa da carta atacada do jogador 2 (caso fossem iguais um defenderia o ataque do outro e nenhum sofreria danos), se o a defesa da carta atacada do jogador 2 for menor que o ataque da carta usada pelo jogador 1, é mostrada uma mensagem explicando a situação citada acima, criamos a variável resultado, que recebe a diferença entre o ataque da carta usada pelo jogador 1 e a defesa da carta atacada do jogador 2. A carta destruída do jogador 2 é removida da área de cartas, do

vetor de estados e de situação das cartas do jogador 2 e o jogador 2 perde pontos relativos ao valor salvo na variável resultado e uma mensagem é mostrada, informando quantos pontos o jogador 2 perdeu.

```

101     else {
102         if (estadoscartasjogadasjogador2.get(index:g).equalsIgnoreCase("defesa")) {
103             System.out.println("O jogador 1 usou a carta " + cartasjogadasjogador1.get(index:r) + " para atacar a carta " + cartasjogadasjogador2.get(index:g) + " d
104
105             if (cartasjogadasjogador2.get(index:g).getDefesa() != cartasjogadasjogador1.get(index:r).getAtaque()) {
106
107                 if (cartasjogadasjogador2.get(index:g).getDefesa() < cartasjogadasjogador1.get(index:r).getAtaque()) {
108
109                     System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 tinha valor de defesa menor que o valor de ataque qu
110
111                     int resultado = cartasjogadasjogador1.get(index:r).getAtaque() - cartasjogadasjogador2.get(index:g).getDefesa();
112
113                     cartasjogadasjogador2.remove(o: cartasjogadasjogador2.get(index:g));
114                     situacaocartasjogadasjogador2.remove(o: situacaocartasjogadasjogador2.get(index:g));
115                     estadoscartasjogadasjogador2.remove(o: estadoscartasjogadasjogador2.get(index:g));
116
117                     pontosjogador2 = pontosjogador2 - resultado;
118                     System.out.println("E o jogador 2 tomou " + resultado + " de dano");
119
120
121
122
123         x:g) + " do jogador 2, que estava no modo de defesa");
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Se o a defesa da carta atacada do jogador 2 não for menor que o ataque da carta usada pelo jogador 1, ou seja, se o a defesa da carta atacada do jogador 2 for maior que o ataque da carta usada pelo jogador 1, criamos a variável resultado, que recebe a diferença da defesa da carta atacada do jogador 2 e o ataque da carta selecionada pelo jogador 1, “setamos” a nova defesa (como resultado do ataque) da carta atacada do jogador 2 como o valor salvo na variável resultado, é mostrado uma mensagem explicando a situação citada acima e o jogador 1 perde pontos relativo ao valor salvo na variável resultado. Caso o índice informado pelo jogador 1 relacionado à carta do jogador 2 que será atacada for inválido, é mostrada uma mensagem informando isso para o jogador. Caso a carta escolhida pelo jogador 1 para atacar esteja no

modo de defesa, é mostrado uma mensagem para o jogador explicando para o jogador que não é possível atacar com uma carta no modo de defesa e o vetor pontuacao é retornado.

```

119
120         } else {
121
122             int resultado = cartasjogadasjogador2.get(index:g).getDefesa() - cartasjogadasjogador1.get(index:r).getAtaque();
123
124             cartasjogadasjogador2.get(index:g).setDefesa(defesa: resultado);
125
126             System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:g) + " do jogador 2 tinha valor de defesa maior q
127             pontosjogador1 = pontosjogador1 - resultado;
128
129         }
130     }
131 }
132
133
134 } else {
135
136     System.out.println("O índice escolhido não é válido");
137
138 }
139
140 }
141
142 } else {
143
144     if (estadoscartasjogadasjogador1.get(index:r).equalsIgnoreCase("defesa")) {
145         System.out.println("Não é possível atacar com uma carta na posição de defesa");
146         return pontuacao;
147     }
148 }

```

```

121
122     setAtaque();
123
124
125
126     or de defesa maior que o valor de ataque que a carta " + cartasjogadasjogador1.get(index:r) + " do jogador 1, ao jogador 1 tomou " + resultado + " de dano");
127
128
129
130
131

```

Se a carta do jogador 1 que atacou não foi destruída, sua situação é atualizada, indicando que ela já atacou nessa rodada, o vetor pontuacao é limpo e atualizado com as novas pontuações e o vetor pontuacao é retornado.

```

150
151         if (foieliminado != null) {
152             if (foieliminado != 1) {
153                 situacaocartasjogadasjogador1.set(index:r, element:1);
154             }
155         } else {
156         }
157         pontuacao.clear();
158
159         if (pontuacao.isEmpty()) {
160             pontuacao.add(e: pontosjogador1);
161             pontuacao.add(e: pontosjogador2);
162         }
163
164         return pontuacao;

```

Se o índice informado relacionado com a carta do jogador 1 que vai realizar o ataque é inválido, uma mensagem é mostrada para o jogador informando isso. Se o índice for válido e está no modo de ataque, ou seja, a situação da carta selecionada pelo jogador 1 é 1, ou seja, já atacou nessa rodada, então é mostrada uma mensagem para o jogador



explicando o porquê de ele não poder atacar usando essa carta nessa rodada, e se o índice for válido e está no modo de defesa, é mostrada uma mensagem explicando para o usuário que não é possível atacar usando uma carta no modo de defesa. E por fim, é retornado o vetor pontuacao.

```

165         } else {
166
167             if (r < 0 || r > situacaocartasjogadasjogador1.size() - 1) {
168                 System.out.println("O índice escolhido não é válido");
169             } else {
170
171                 if (estadocartasjogadasjogador1.get(index:r).equalsIgnoreCase("ataque")) {
172
173                     System.out.println("A carta escolhida pelo jogador " + jogador + " já atacou nesse turno e só poderá atacar na próxima rodada, escolha outra opção ou passe sua vez");
174                 }
175                 if (estadocartasjogadasjogador1.get(index:r).equalsIgnoreCase("defesa")) {
176                     System.out.println("Não é possível atacar com uma carta na posição de defesa");
177                 }
178             }
179         }
180         return pontuacao;
181     }
182 }
183
184 }
185
186 }
187

```

```

166         tuacaocartasjogadasjogador1.size() - 1) {
167             :ln("O índice escolhido não é válido");
168
169
170
171             asjogadasjogador1.get(index:r).equalsIgnoreCase("ataque")) {
172
173                 >println("A carta escolhida pelo jogador " + jogador + " já atacou nesse turno e só poderá atacar na próxima rodada, escolha outra opção ou passe sua vez");
174
175             asjogadasjogador1.get(index:r).equalsIgnoreCase("defesa")) {
176                 >println("Não é possível atacar com uma carta na posição de defesa");
177
178
179
180
181
182

```

Agora, caso o jogador seja igual à 2, seguimos uma lógica semelhante ao caso ataque do jogador 1:

Declaramos a variável `r`, que receberá o índice que o jogador monstro que ele irá usar no ataque e a variável `foieliminado`, que registrará se a carta que foi usada no ataque foi destruída no ataque. Enquanto o jogador 2 ainda tiver cartas que podem atacar, as variáveis `pontosjogador1` e `pontosjogador2` recebem seus respectivos valores vindos do vetor `pontuacao`, mostramos uma mensagem para o jogador 2 pedindo que informe a carta que quer usar no ataque e o programa salva o valor informado na variável `r` e o valor de `r` é subtraído em 1, já que os índices de vetores começam

em 0. Se o índice informado for válido, pegamos a carta informada pelo usuário e salvamos na variável `monstroescolhido` e se ela for uma carta que está no estado de ataque e pode atacar naquele turno.

Se o jogador 1 não posicionou nenhum monstro no campo de cartas:

```

187
188     if (jogador == 2) {
189         Integer r = 0;
190         Integer foieliminada = null;
191
192         while (situacaocartasjogadasjogador2.contains(0) == true) {
193
194             pontosjogador1 = pontuacao.get(index:0);
195             pontosjogador2 = pontuacao.get(index:1);
196
197             System.out.println("Digite o monstro que você deseja usar:");
198
199             r = entrada.nextInt();
200
201             r--;
202             if (r >= 0 && r <= situacaocartasjogadasjogador2.size() - 1 && situacaocartasjogadasjogador2.get(index:r) == 0) {
203
204                 cartasmonstroescolhido = cartasjogadasjogador2.get(index:r);
205
206                 if (estadoscartasjogadasjogador2.get(index:r).equalsIgnoreCase("ataque")) {
207
208                     if (situacaocartasjogadasjogador2.get(index:r) == 0) {
209
210                         if (cartasjogadasjogador1.isEmpty() == true) {

```

Se o jogador 1 não tiver posicionado nenhuma carta do tipo monstro, o jogador 1 perde os pontos relativos aos pontos de ataque da carta selecionada pelo jogador 2, e o vetor `situacaocartasjogadasjogador2` é atualizado no índice `r` de 0 para 1, indicando que a carta no índice `r` não pode mais atacar no mesmo turno. E uma mensagem é mostrada explicando como ocorreu o ataque.

Se o jogador 1 posicionou cartas do tipo monstro, e a carta informada pelo jogador 2 pode atacar, é mostrada uma mensagem perguntando ao jogador 2 qual carta inimiga ele quer atacar, e o valor informado é salvo na variável `g` e o valor de `g` é subtraído em 1, já que os índices de vetores começam em 0. Se o índice informado for válido, inicializamos a variável `foieliminada`.

Se a carta atacada estiver no estado de ataque, é mostrada uma mensagem sobre o ataque, se o ataque do jogador 1 for maior que o ataque do jogador 2 uma mensagem é mostrada que fala sobre o resultado do ataque, criamos a variável `resultado`, e a diferença entre os pontos de ataque é salva nela, “setamos” o novo ataque (como resultado do ataque) da

carta selecionada pelo jogador 2 como o valor salvo na variável resultado, a carta destruída do jogador 1 é removida da área de cartas e de situação das cartas do jogador 2.

```

210     if (cartasjogadasjogador1.isEmpty() == true) {
211
212         if (estadoscartasjogadasjogador2.get(index:r).equalsIgnoreCase("ataque")) {
213
214             pontosjogador1 = pontosjogador1 - monstroescolhido.getAtaque();
215             System.out.println("O jogador 2 atacou com a carta " + monstroescolhido + " e já que o jogador 1 não havia posicio
216         }
217         situacaocartasjogadasjogador2.set(index:r, element:l);
218     }
219
220     if (cartasjogadasjogador1.isEmpty() == false) {
221
222         if (situacaocartasjogadasjogador2.get(index:r) == 0) {
223             System.out.println("Digite o monstro inimigo que você deseja atacar:");
224             int g = entrada.nextInt();
225             g--;
226
227             if (g >= 0 && g <= estadoscartasjogadasjogador1.size() - 1) {
228                 foieliminado = 0;
229
230                 if (estadoscartasjogadasjogador1.get(index:g).equalsIgnoreCase("ataque")) {
231                     System.out.println("O jogador 2 usou a carta " + cartasjogadasjogador2.get(index:r) + " para atacar a carta
232
233                     if (cartasjogadasjogador1.get(index:g).getAtaque() < cartasjogadasjogador2.get(index:r).getAtaque()) {
234                         System.out.println("Já que a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1 tinha menor
235
236                         int resultado = cartasjogadasjogador2.get(index:r).getAtaque() - cartasjogadasjogador1.get(index:g).getA
237
238                         cartasjogadasjogador2.get(index:r).setAtaque(ataque: resultado);
239
240                         cartasjogadasjogador1.remove(o: cartasjogadasjogador1.get(index:g));
241                         situacaocartasjogadasjogador1.remove(o: situacaocartasjogadasjogador1.get(index:g));
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

E também é removida do vetor de estados e de situação das cartas do jogador 1 e o jogador 1 perde pontos relativos ao valor salvo na variável resultado e uma mensagem é mostrada, informando quantos pontos o jogador 1 perdeu.

Caso o ataque do jogador 2 for igual aos pontos de ataque da carta do inimigo é mostrado uma mensagem sobre o ataque, a carta do jogador 2 que atacou é removida do vetor de cartas jogadas do jogador 2, de situações das cartas do jogador 2 e de estados das cartas do jogador 2. A variável foieliminado é atualizada, e a carta do jogador 1 que foi atacada é

removida do vetor de cartas jogadas do jogador 2, de situações das cartas do jogador 1, e de estados das cartas do jogador 1.

```

242     estadoscartasjogadasjogador1.remove(o: estadoscartasjogadasjogador1.get(index:g));
243
244     pontosjogador1 = pontosjogador1 - resultado;
245     System.out.println("E o jogador 1 tomou " + resultado + " de dano");
246 } else {
247     if (cartasjogadasjogador1.get(index:g).getAtaque() == cartasjogadasjogador2.get(index:r).getAtaque()) {
248         System.out.println("Já que a carta " + cartasjogadasjogador2.get(index:r) + " do jogador 2 tinha o mesmo valor d
249         cartasjogadasjogador2.remove(o: cartasjogadasjogador2.get(index:r));
250         situacaocartasjogadasjogador2.remove(o: situacaocartasjogadasjogador2.get(index:r));
251         estadoscartasjogadasjogador2.remove(o: estadoscartasjogadasjogador2.get(index:r));
252
253         foieliminada = 1;
254
255         cartasjogadasjogador1.remove(o: cartasjogadasjogador1.get(index:g));
256         situacaocartasjogadasjogador1.remove(o: situacaocartasjogadasjogador1.get(index:g));
257         estadoscartasjogadasjogador1.remove(o: estadoscartasjogadasjogador1.get(index:g));
258
259     }
260     {
261         System.out.println("o mesmo valor de ataque que a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1, as duas cartas foram destruídas e nenhum jogador sofreu danos");
262     }
263 }
264

```

Se o ataque do jogador 2 for menor que o ataque do jogador 1 uma mensagem é mostrada que fala sobre o resultado do ataque, criamos a variável resultado, e a diferença entre os pontos de ataque é salva nela. A variável foieliminada é atualizada, “setamos” o novo ataque (como resultado do ataque) da carta atacada do jogador 1 como o valor salvo na variável resultado, a carta destruída do jogador 2 é removida da área de cartas, do vetor de estados e de situação das

cartas do jogador 2 e o jogador 2 perde pontos relativos ao valor salvo na variável resultado e uma mensagem é mostrada, informando quantos pontos o jogador 2 perdeu.

```

257     estadoscartasjogadasjogador1.remove(o: estadoscartasjogadasjogador1.get(index:g));
258 } else {
259
260     if (cartasjogadasjogador1.get(index:g).getAtaque() > cartasjogadasjogador2.get(index:r).getAtaque()) {
261         System.out.println("Já que a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1 tinha maior valor de ataque que a carta " + cart
262
263         int resultado = cartasjogadasjogador1.get(index:g).getAtaque() - cartasjogadasjogador2.get(index:r).getAtaque();
264
265         cartasjogadasjogador1.get(index:g).setAtaque(ataque: resultado);
266
267         cartasjogadasjogador2.remove(o: cartasjogadasjogador2.get(index:r));
268         situacaocartasjogadasjogador2.remove(o: situacaocartasjogadasjogador2.get(index:r));
269         estadoscartasjogadasjogador2.remove(o: estadoscartasjogadasjogador2.get(index:r));
270
271         foieliminado = 1;
272         pontosjogador2 = pontosjogador2 - resultado;
273         System.out.println("E o jogador 2 tomou " + resultado + " de dano");
274     }
275 }
276
277
278 {
279     maior valor de ataque que a carta " + cartasjogadasjogador2.get(index:r) + " a carta " + cartasjogadasjogador2.get(index:r) + " do jogador 2 foi destruída");
280
281     r).getAtaque();
282
283
284
285
286

```

Caso a carta atacada do jogador 1 esteja no modo de defesa, uma mensagem falando sobre qual carta foi usada pelo jogador 2 para atacar e qual carta do jogador 1 foi atacada. Se o ataque e da carta usada pelo jogador 2 for diferente da defesa da carta atacada do jogador 1 (caso fossem iguais um defenderia o ataque do outro e nenhum sofreria danos), se o a defesa da carta atacada do jogador 1 for menor que o ataque da carta usada pelo jogador 2, é mostrada uma mensagem explicando a situação citada acima, criamos a variável resultado, que recebe a diferença entre o ataque da carta usada pelo jogador 2 e a defesa da carta atacada do jogador 1. A carta destruída do jogador 1 é removida da área de cartas, do vetor de estados e de situação das cartas do jogador 1 e o jogador 1 perde pontos relativos ao valor salvo na variável resultado e uma mensagem é mostrada, informando quantos pontos o jogador 1 perdeu.

Se o a defesa da carta atacada do jogador 1 não for menor que o ataque da carta usada pelo jogador 2, ou seja, se o a defesa da carta atacada do jogador 1 for maior que o ataque da carta usada pelo jogador 2, criamos a variável resultado, que recebe a diferença da defesa da carta atacada do jogador 1 e o ataque da carta selecionada pelo jogador 2, “setamos” a nova defesa (como resultado do ataque) da carta atacada do jogador 1 como o valor salvo na variável resultado, é

mostrado uma mensagem explicando a situação citada acima e o jogador 2 perde pontos relativo ao valor salvo na variável resultado.

```

277     } else {
278         if (estadoscartasjogadasjogador1.get(index:g).equalsIgnoreCase(anotherString: "defesa")) {
279             System.out.println("O jogador 2 usou a carta " + cartasjogadasjogador2.get(index:r) + " para atacar a carta " + cartasjogadasjogador1.get(index:g)
280
281             if (cartasjogadasjogador1.get(index:g).getDefesa() != cartasjogadasjogador2.get(index:r).getAtaque()) {
282                 if (cartasjogadasjogador1.get(index:g).getDefesa() < cartasjogadasjogador2.get(index:r).getAtaque()) {
283
284                     System.out.println("Já que a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1 tinha valor de defesa menor que o valor de at
285
286                     int resultado = cartasjogadasjogador2.get(index:r).getAtaque() - cartasjogadasjogador1.get(index:g).getDefesa();
287
288                     cartasjogadasjogador1.remove(o: cartasjogadasjogador1.get(index:g));
289                     situacaocartasjogadasjogador1.remove(o: situacaocartasjogadasjogador1.get(index:g));
290                     estadoscartasjogadasjogador1.remove(o: estadoscartasjogadasjogador1.get(index:g));
291
292                     pontosjogador1 = pontosjogador1 - resultado;
293                     System.out.println("E o jogador 1 tomou " + resultado + " de dano");
294
295                 } else {
296                     int resultado = cartasjogadasjogador1.get(index:g).getDefesa() - cartasjogadasjogador2.get(index:r).getAtaque();
297
298                     cartasjogadasjogador1.get(index:g).setDefesa(defesa: resultado);
299
300                     System.out.println("Já que a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1 tinha valor de defesa maior que o valor de at
301                     pontosjogador2 = pontosjogador2 - resultado;
302
303                 }
304             }
305         }
306     }

```

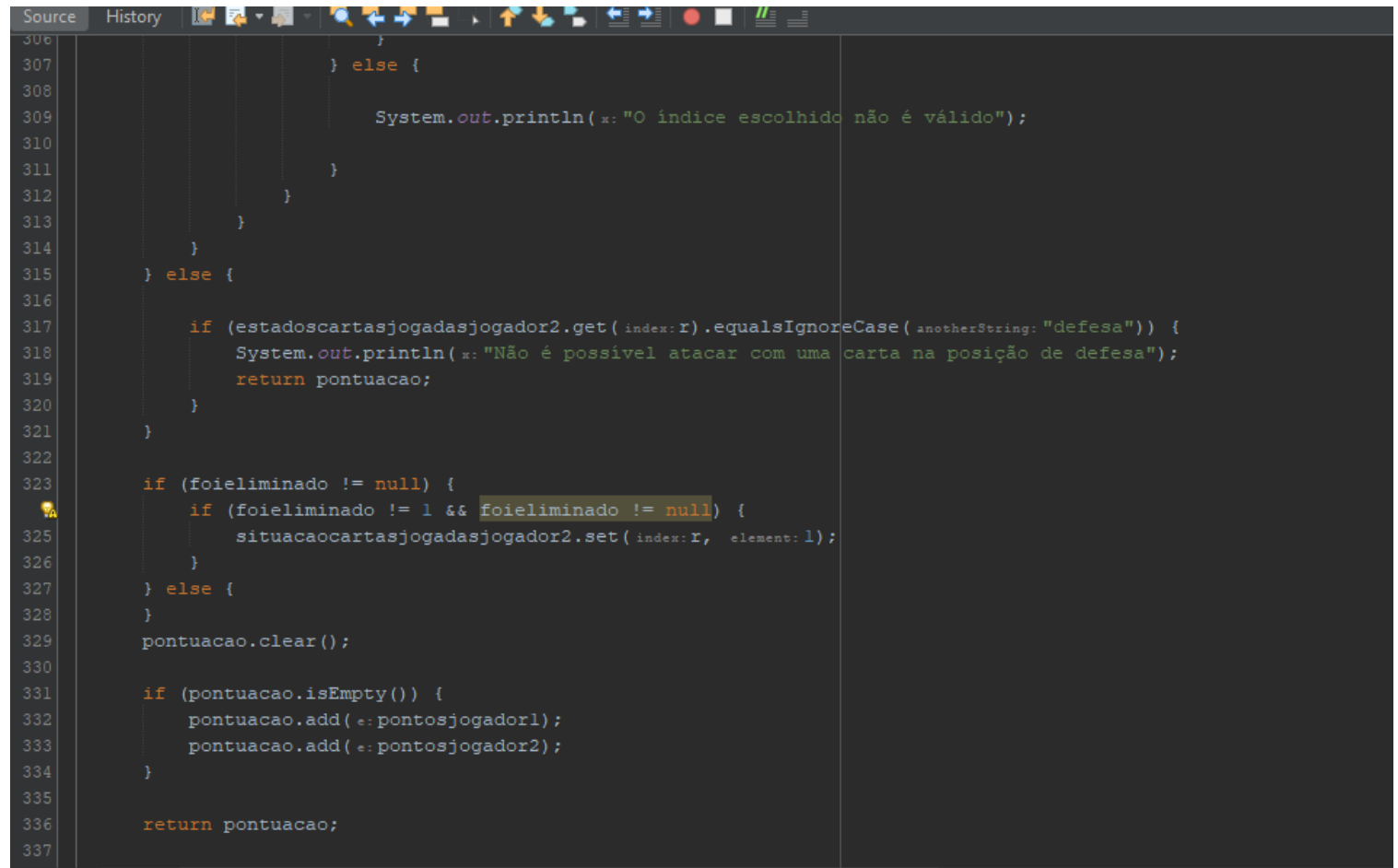
```

278     index:g) + " do jogador 1, que estava no modo de defesa");
279
280
281
282
283
284     : ataque que a carta " + cartasjogadasjogador2.get(index:r) + " do jogador 2, a carta " + cartasjogadasjogador1.get(index:g) + " do jogador 1 foi destruída");
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300     : ataque que a carta " + cartasjogadasjogador2.get(index:r) + " do jogador 2, ao jogador 2 tomou " + resultado + " de dano");
301
302

```

Caso o índice informado pelo jogador 2 relacionado à carta do jogador 1 que será atacada for inválido, é mostrada uma mensagem informando isso para o jogador. Caso a carta escolhida pelo jogador 2 para atacar esteja no modo de defesa, é mostrado uma mensagem para o jogador explicando para o jogador que não é possível atacar com uma carta no modo de defesa e o vetor pontuacao é retornado. Se a carta do jogador 2 que atacou não foi destruída, sua situação é atualizada,

indicando que ela já atacou nessa rodada, o vetor pontuacao é limpo e atualizado com as novas pontuações e o vetor pontuacao é retornado.



```

306     }
307     } else {
308
309         System.out.println(x: "O índice escolhido não é válido");
310
311     }
312 }
313 }
314 }
315 } else {
316
317     if (estadoscartasjogadasjogador2.get(index:r).equalsIgnoreCase("defesa")) {
318         System.out.println(x: "Não é possível atacar com uma carta na posição de defesa");
319         return pontuacao;
320     }
321 }
322
323 if (foieliminado != null) {
324     if (foieliminado != 1 && foieliminado != null) {
325         situacaocartasjogadasjogador2.set(index:r, element:1);
326     }
327 } else {
328 }
329 pontuacao.clear();
330
331 if (pontuacao.isEmpty()) {
332     pontuacao.add(e: pontosjogador1);
333     pontuacao.add(e: pontosjogador2);
334 }
335
336 return pontuacao;
337

```

Se o índice informado relacionado com a carta do jogador 2 que vai realizar o ataque é inválido, uma mensagem é mostrada para o jogador informando isso. Se o índice for válido e está no modo de ataque, ou seja, a situação da carta selecionada pelo jogador 2 é 1, ou seja, já atacou nessa rodada, então é mostrada uma mensagem para o jogador explicando o porquê de ele não poder atacar usando essa carta nessa rodada, e se o índice for válido e está no modo de

defesa, é mostrada uma mensagem explicando para o usuário que não é possível atacar usando uma carta no modo de defesa. E por fim, é retornado o vetor pontuacao.

```

338     } else {
339         if (r < 0 || r > situacaocartasjogadasjogador2.size() - 1) {
340             System.out.println("O índice escolhido não é válido");
341         } else {
342
343             if (estadoscartasjogadasjogador2.get(index:r).equalsIgnoreCase("ataque")) {
344
345                 System.out.println("A carta escolhida pelo jogador " + jogador + " já atacou nesse turno e só poderá atacar na próxima rodada, escolha outra opção ou passe sua vez");
346             }
347             if (estadoscartasjogadasjogador2.get(index:r).equalsIgnoreCase("defesa")) {
348                 System.out.println("Não é possível atacar com uma carta na posição de defesa");
349             }
350         }
351     }
352     return pontuacao;
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 return pontuacao;
361 }
362 }
363 }
364 }

```

```

338     situacaocartasjogadasjogador2.size() - 1) {
339         ln("O índice escolhido não é válido");
340     }
341 }
342 }
343 sjogadasjogador2.get(index:r).equalsIgnoreCase("ataque")) {
344
345     rintln("A carta escolhida pelo jogador " + jogador + " já atacou nesse turno e só poderá atacar na próxima rodada, escolha outra opção ou passe sua vez");
346 }
347 sjogadasjogador2.get(index:r).equalsIgnoreCase("defesa")) {
348     rintln("Não é possível atacar com uma carta na posição de defesa");
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }

```

Voltamos para a classe Jogatina:

Se a pontuação retornada de algum dos jogadores for menor igual à 0, um menu é mostrado com as pontuações dos dois jogadores, uma mensagem que fala qual foi o jogador vencedor é mostrada e a execução do programa é finalizada. Se o jogador 1 não tiver cartas na área de cartas e tentar atacar, uma mensagem falando que a opção né está disponível é



mostrada para o jogador, senão, uma mensagem é mostrada para o jogador avisando que ele não possui cartas na área de cartas que podem atacar naquela rodada.

```

272         if (pontuacao.get(index:0) <= 0 || pontuacao.get(index:1) <= 0) {
273             System.out.println("\n");
274             System.out.println("\nPontuação:");
275             System.out.println("Jogador 1: " + pontuacao.get(index:0));
276             System.out.println("Jogador 2: " + pontuacao.get(index:1));
277             System.out.println("\n");
278
279             if (pontuacao.get(index:1) <= 0) {
280                 System.out.println("\nO jogador 1 ganhou");
281             }
282             if (pontuacao.get(index:0) <= 0) {
283                 System.out.println("\nO jogador 2 ganhou");
284             }
285
286             System.exit(status:0);
287         }
288
289     } else {
290         if (cartasjogadasjogador1.isEmpty() == true) {
291             System.out.println("\nOpção indisponível");
292         } else {
293             System.out.println("\nO jogador " + jogador + " não possui cartas que podem atacar");
294         }
295     }
296 }
297

```

Se o jogador for igual à 2 e o jogador 2 ainda tem cartas que podem atacar (tem cartas com situação igual à 0) e invocamos o método ataque da classe Ataque e o resultado é salvo no vetor pontuação. Se a pontuação retornada de algum dos jogadores for menor igual à 0, um menu é mostrado com as pontuações dos dois jogadores, uma mensagem que fala qual foi o jogador vencedor é mostrada e a execução do programa é finalizada. Se o jogador 2 não tiver cartas na área de cartas e tentar atacar, uma mensagem falando que a opção não está disponível é mostrada para o jogador, senão, uma

mensagem é mostrada para o jogador avisando que ele não possui cartas na área de cartas que podem atacar naquela rodada.

```

299     if (jogador == 2) {
300         if (situacaocartasjogadasjogador2.contains(c: 0) == true) {
301
302             pontuacao = Ataque.ataque(pontuacao, pontosjogador1, pontosjogador2, jogador, cartasjogadasjogador1, cartasjogadasjogador2, estadoscartasjogadasjogador2);
303
304             if (pontuacao.get(index: 0) <= 0 || pontuacao.get(index: 1) <= 0) {
305                 System.out.println(x: "");
306                 System.out.println(x: "Pontuação:");
307                 System.out.println("Jogador 1: " + pontuacao.get(index: 0));
308                 System.out.println("Jogador 2: " + pontuacao.get(index: 1));
309                 System.out.println(x: "");
310
311                 if (pontuacao.get(index: 1) <= 0) {
312                     System.out.println(x: "O jogador 1 ganhou");
313                 }
314                 if (pontuacao.get(index: 0) <= 0) {
315                     System.out.println(x: "O jogador 2 ganhou");
316                 }
317
318                 System.exit(status: 0);
319             }
320
321         } else {
322             if (cartasjogadasjogador2.isEmpty() == true) {
323                 System.out.println(x: "Opção indisponível");
324             } else {
325                 System.out.println("O jogador " + jogador + " não possui cartas que podem atacar");
326             }
327         }

```

Se os jogadores tentarem atacar em rodadas antes da 3º rodada, uma mensagem é mostrada para os usuários falando que a opção não está disponível.

```

330
331         } else {
332
333             if (x == 7 && rodada <= 2) {
334                 System.out.println(x: "opção indisponível");
335             }
336         }
337

```

## 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 que além de ser um código extenso, usa recursos recém-aprendidos por nós.

**REFERÊNCIAS**

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

[https://yugioh.fandom.com/pt-br/wiki/Manual\\_Oficial\\_de\\_Regras](https://yugioh.fandom.com/pt-br/wiki/Manual_Oficial_de_Regras)