# ADO - Estruturas de Dados

Daiane Vitória Raso

February 28, 2025

# 1 Situação problema

Uma clínica médica precisa de um sistema para gerenciar suas consultas. O sistema deve permitir que o administrador da clínica adicione novas consultas ao sistema, pesquise consultas existentes pelo nome do paciente ou pelo índice do vetor, altere informações de uma consulta (como data, horário ou especialidade), e remova consultas que foram canceladas ou concluídas.

A Classe Vetor manipula os dados e a Classe Consulta permite a manipulação dos dados dos objetos armazenados no vetor.

## 2 Classe Consulta

```
import java.util.Calendar;
import java.text.SimpleDateFormat;

public class Consulta {
    Calendar calendario = Calendar.getInstance();
    private String nome;
    private int idade;
    private String especialidade;
    private Calendar data;
    private String unidade;
}
```

### 2.1 Construtor

```
public Consulta(String nome, int idade, String especialidade,
    int ano, int mes, int dia, int hora, int minuto, String
    unidade) {
        this.nome = nome;
        this.idade = idade;
        this.especialidade = especialidade;
        this.unidade = unidade;

        this.data = Calendar.getInstance();
        this.data.set(ano, mes, dia, hora, minuto);
    }
```

### 2.2 Getters e Setters

```
    public String getNome() {
        return nome;
    }

    public int getIdade() {
        return idade;
    }

    public String getEspecialidade() {
        return especialidade;
    }

    public void setEspecialidade(String especialidade) {
        this.especialidade = especialidade;
    }

```

```
17      public Calendar getData() {
18          return data;
19      }
20
21      public void setData(Calendar data) {
22          this.data = data;
23      }
24
25      public String getUnidade() {
26          return unidade;
27      }
28
29      public void setUnidade(String unidade) {
30          this.unidade = unidade;
31      }
```

## 2.3 toString

```
1       @Override
2       public String toString() {
3           SimpleDateFormat formatador = new SimpleDateFormat("dd/
                MM/yyyy - HH:mm");
4           String dataFormatada = formatador.format(data.getTime()
                );
5
6           String msg = String.format("-------------------------%n
                " +
7           "Nome do paciente: %s%n" +
8           "Idade: %d%n" +
9           "Especialidade: %s%n" +
10          "Data: %s%n" +
11          "Unidade: %s%n" +
12          "-------------------------%n", nome, idade,
                especialidade, dataFormatada, unidade);
13
14          return msg;
15      }
```

# 3 Classe Vetor

```
public class Vetor {
    private Consulta[] elementos;
    private int tamanho;
    private int quantidade;
}
```

## 3.1 Construtor

```
    public Vetor(int tamanho) {
        this.tamanho = tamanho;
        this.elementos = new Consulta[tamanho];
        this.quantidade = 0;
    }
```

## 3.2 Método para adicionar ao vetor

```
    // m todo para adicionar ao vetor
    public void adicionar(Consulta item) {
        if (quantidade == tamanho) {
            this.aumentaCapacidade();
            this.quantidade++;
        }

        for (int i = 0; i < tamanho; i++) {
            if (elementos[i] == null) {
                elementos[i] = item;
                quantidade++;
                break;
            }
        }
    }

    // m todo sobrescrito para adicionar ao vetor (com indice)
    public void adicionar(int index, Consulta item) throws
        Exception {
        if (index < 0 || index > quantidade) {
            throw new Exception("Indice fora dos limites");
        }

        if (quantidade == tamanho) {
            this.aumentaCapacidade();
        }
        for (int i = quantidade - 1; i >= index; i--) {
            elementos[i + 1] = elementos[i];
```

```
        }
        elementos[index] = item;
        this.quantidade++;
    }

    // m todo para adicionar elemento no in cio do vetor
    public void adicionarInicio(Consulta item) {
        if (quantidade == tamanho) {
            this.aumentaCapacidade();
        }
        for (int i = quantidade - 1; i >= 0; i--) {
            elementos[i + 1] = elementos[i];
        }
        this.elementos[0] = item;
        this.quantidade++;
    }

    // m todo para adicionar elemento no fim do vetor
    public void adicionarUltimo(Consulta item) {
        if (quantidade == tamanho) {
            this.aumentaCapacidade();
        }
        this.elementos[quantidade] = item;
        this.quantidade++;
    }
```

### 3.3 Métodos de busca

```
    // m todo que retorna um elemento baseado no  ndice
    public String busca(int index) throws Exception {
        if (index < 0 || index >= quantidade) {
            throw new Exception("Indice fora dos limites");
        }
        return elementos[index].toString();
    }

    // m todo sobrescrito que retorna o index do elemento
    public int busca(Consulta item) throws Exception {
        for (int i = 0; i < quantidade; i++) {
            if (elementos[i] == item) {
                return i;
            }
        }
        throw new Exception("Elemento nao encontrado");
    }

    // m todo que retorna o primeiro elemento
    public String buscaPrimeiro() throws Exception {
```

```
        if (!isVazio()) {
            return elementos[0].toString();
        }
        throw new Exception("O vetor esta vazio");
    }

    // m todo que retorna o  ltimo  elemento
    public String buscaUltimo() throws Exception {
        if (!isVazio()) {
            return elementos[quantidade - 1].toString();
        }
        throw new Exception("O vetor esta vazio");
    }
```

## 3.4 Métodos de remoção

```
    // m todo que remove um elemento pelo index
    public void remove(int index) throws Exception {
        if (index < 0 || index >= quantidade) {
            throw new Exception("Indice fora dos limites");
        } else {
            for (int i = index; i < quantidade - 1; i++) {
                this.elementos[i] = this.elementos[i + 1];
            }
            this.elementos[quantidade - 1] = null;
            this.quantidade--;
        }
    }

    // m todo sobrescrito que remove um elemento pelo objeto
    public void remove(Consulta item) throws Exception {
        if (this.contem(item)) {
            int pos = -1;

            for (int i = 0; i < quantidade; i++) {
                if (elementos[i] == item) {
                    pos = i;
                    break;
                }
            }

            if (pos != -1) {
                for (int i = pos; i < quantidade - 1; i++) {
                    elementos[i] = elementos[i + 1];
                }
                elementos[quantidade - 1] = null;
                quantidade--;
            }
```

```
33          } else {
34              throw new Exception("Esse item nao existe no vetor"
                    );
35          }
36      }
```

### 3.5 Método para alterar elemento

```
1       public boolean alterar(int index, Consulta item) {
2           if (!this.isVazio()) {
3               if (index > this.quantidade || index < 0) {
4                   if (!this.contem(item)) {
5                       return false;
6                   }
7                   elementos[index] = item;
8                   return true;
9               }
10              return false;
11          }
12          return false;
13      }
```

### 3.6 Método para aumentar capacidade

Utilizado nos casos em que o usuário faz adição de elemento no vetor quando o seu tamanho já não é mais compatível com a capacidade.

```
1       private void aumentaCapacidade() {
2           if (quantidade == elementos.length) {
3               Consulta[] elementosNovos = new Consulta[elementos.
                    length * 2];
4               for (int i = 0; i < quantidade; i++) {
5                   elementosNovos[i] = elementos[i];
6               }
7               this.elementos = elementosNovos;
8               this.tamanho = elementosNovos.length;
9           }
10      }
```

### 3.7 Método para limpar o vetor

```
1       public void limpar() {
2           for (int i = 0; i < quantidade; i++) {
3               elementos[i] = null;
4           }
5           this.quantidade = 0;
```

```
    }
```

### 3.8 Método para verificar o tamanho do vetor

```
    public int tamanho() {
        return this.tamanho;
    }
```

### 3.9 Método para verificar a quantidade de elementos

```
    public int quantidadeAtual() {
        return this.quantidade;
    }
```

### 3.10 Método para verificar se o vetor está vazio

```
    public boolean isVazio() {
        if (quantidade == 0) {
            return true;
        }
        return false;
    }
```

### 3.11 Método que verifica se o vetor contém o objeto

```
    public boolean contem(Consulta item) {
        for (int i = 0; i < quantidade; i++) {
            if (elementos[i] == item) {
                return true;
            }
        }
        return false;
    }
```

### 3.12 toString

```
    @Override
    public String toString() {
        StringBuilder string = new StringBuilder();
        string.append("[ ");

        for (int i = 0; i < quantidade - 1; i++) {
```

```
7               string.append(elementos[i]);
8               string.append(", ");
9           }
10
11          if (quantidade > 0) {
12              string.append(elementos[quantidade - 1]);
13          }
14
15          string.append(" ]");
16
17          return string.toString();
18      }
```

# 4 Classe Main

Para instanciar as supracitadas classes Consulta e Vetor, e realizar testes.

```
    import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        try {
            // criador o vetor das consultas
            Vetor vetor = new Vetor(3);

            // instanciando as Consultas
            Consulta c1, c2, c3, c4, c5;
            c1 = new Consulta("P a d m  Amigdala", 25, "
                Dermatologia", 2025, Calendar.MARCH, 12, 12, 30,
                 "Votuporanga");
            c2 = new Consulta("Anakin Skywalker", 22, "
                Psiquiatria", 2025, Calendar.MARCH, 20, 9, 0, "
                M a r l i a");
            c3 = new Consulta("Yoda Mandalorian", 800, "
                Psicologia", 2025, Calendar.APRIL, 2, 16, 45, "
                N a  e s  Unidas");
            c4 = new Consulta("Lando Calrissian", 41, "
                Fisioterapia", 2025, Calendar.FEBRUARY, 26, 7,
                00, "Interlagos");
            c5 = new Consulta("Han Solo", 35, "Ortopedia",
                2025, Calendar.MARCH, 1, 15, 30, "Lapa");

            // adicionando consultas ao vetor
            vetor.adicionar(c1);
            vetor.adicionar(c2);
            vetor.adicionar(c3);
            vetor.adicionar(c4);
            vetor.adicionar(c5);

            // imprimindo o vetor
            System.out.println("Vetor de consultas:");
            System.out.println(vetor);

            // buscando uma consulta pelo  ndice
            System.out.println("\nConsulta no  ndice  1:");
            System.out.println(vetor.busca(1));

            // buscando uma consulta pelo objeto
            System.out.println("\nBuscando  ndice  da consulta
                do Yoda:");
            System.out.println(vetor.busca(c3));

            // removendo uma consulta pelo  ndice
```

```
37              System.out.println("\nRemovendo a consulta no
                    ndice  0:");
38              vetor.remove(0); // remove a consulta de Padm
39              System.out.println(vetor);
40
41              // removendo uma consulta pelo objeto
42              System.out.println("\nRemovendo a consulta do
                    Anakin:");
43              vetor.remove(c2);
44              System.out.println(vetor);
45
46              // adicionando mais uma consulta ao in cio
47              System.out.println("\nAdicionando uma nova consulta
                    no in cio:");
48              vetor.adicionarInicio(new Consulta("Obi-Wan Kenobi"
                    , 75, "Geriatria", 2025, Calendar.MARCH, 8, 10,
                    0, "Ca apava"));
49              System.out.println(vetor);
50
51              // buscando o primeiro e  ltimo  elemento
52              System.out.println("\nPrimeira Consulta: \n" +
                    vetor.buscaPrimeiro());
53              System.out.println(" ltima  Consulta: \n" + vetor.
                    buscaUltimo());
54
55              // limpando o vetor
56              vetor.limpar();
57              System.out.println("\nVetor ap s limpeza:");
58              System.out.println(vetor);
59
60          } catch (Exception e) {
61              System.out.println(e.getMessage());
62          }
63
64      }
65  }
```