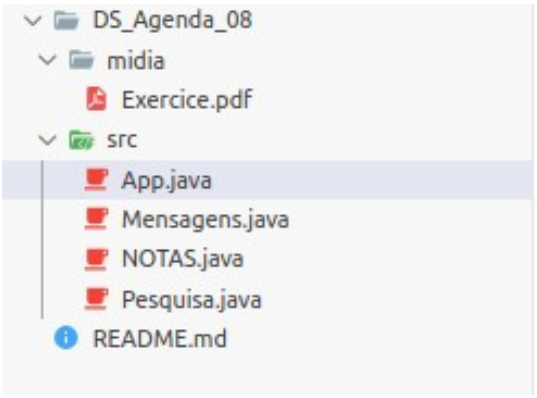
Desenvolvimento de Sistemas
Agenda 08

Matheus Henrique da Silva Mendes

Sumário.

1. Arquivos gerados.



- App: Classe principal
- Pesquisa: classe de funcionalidades da aplicação e que contém toda a regra de negócio.
- NOTAS: classe do tipo ENUM, contendo os valores constantes
- Mensagens: classe para armazenar mensagens de interação com o usuário.

2.Classe App: Classe principal, contendo o metodo main. Usado para iniciar a aplicação.

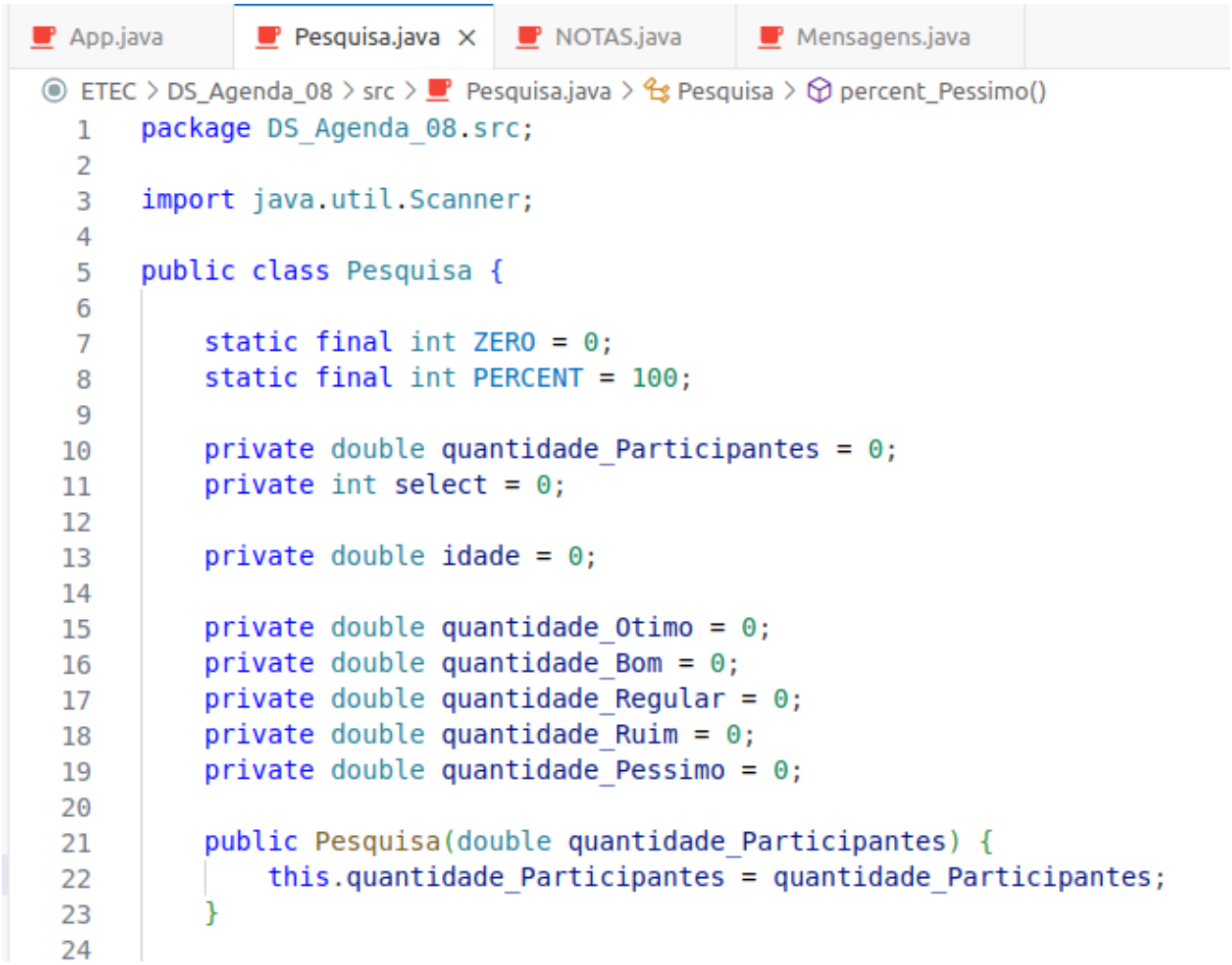
App.java | Pesquisa.java | NOTAS.java | Mensagens.java

ETEC > DS_Agenda_08 > src > App.java > App

```java
package DS_Agenda_08.src;

public class App {
    static final int QUANTIDADADE_PARTICIPANTES = 40;
    Run | Debug
    public static void main(String[] args) {


        Pesquisa pesquisa = new Pesquisa(QUANTIDADADE_PARTICIPANTES);

        pesquisa.start();
        pesquisa.resultado();
    }
}

```

- Foi criado um objeto do tipo Pesquisa chamado pesquisa e atribuido um valor do tipo int em sua criação chamado QUANTIDADE_PARTICIPANTES (constante).
- Em sequencia foi iniciado dois metodos consecutivos do objeto pesquisa: start e resultado.
    - Pesquisa.start() → iniciar os calculos, do algoritimo.
    - Pesquisa.resultado() → retorna os valores que devem ser apresentados ao usuario final.

3. Classe Pesquisa: Regras de negócio



Classe que contém toda lógica da regra de negócios da aplicação. Escolhir fazer o programa com Orientação a objetos para facilitar a organização do código.

Iniciei a criação dessa classe com duas constates do tipo Int apenas para evitar valores numericos soltoos no cóogdiigo.

Em sequencia criei variaveis privadas que representam a quantidade de avalizações, uma para cada tipo de avaliação. Escolhir faze-las como decimais por conta de calculos futuros envolvendo divisão.

No final dessa imagem, dá para notar que foi criado um método construtor onde é obrigado informar uma quantidade de participantes na hora em que um objeto do tipo Pesquisa for criado.

3.1 Metodo Start

```
24
25      public void start(){
26
27          Scanner scan = new Scanner(System.in);
28
29          for (int i = 1; i <= getQuantidade_Participantes(); i++) {
30
31              System.out.println("pessoa Numero: " + i );
32              Mensagens.enter_Idade();
33              set_Idade(scan.nextInt());
34
35              do {
36                  Mensagens.enter_Nota();
37                  Mensagens.select();
38                  Select(scan.nextInt());
39
40
41              } while (getSelect() < ZERO || getSelect() > NOTAS.getSize
42          }
43
44          scan.close();
45      }
```

Metodo responsavel para iniciar os calculos do aaplicativo. Controla a entrada e saída de dados.

For → utilizado para repetir a entrada de dados e armazenalas em suas respectiveis variaveis.

Metodo Select() → recebe um valor do tipo inteiro e faz secagens para saber se é um numero válido ou não.

Do While → verificar se o valor é valido, caso não seja , o ira aparecer uma mensagem de erro e obrigando o usuario entrar com um valor que seja aceito pelo programa.

3.2 Metodo resultado

```
46
47      public void resultado(){
48          Mensagens.print_Total_Otimas(getQuantidade_Otimo());
49          Mensagens.print_Media_Ruim(media_Idade_Quantidade_Ruim());
50          Mensagens.print_Percent_Pessimo(percent_Pessimo());
51      }
52
```

Resposave apenas para retornar os valores finais da aplicação, como o nome sugere, os resultados. Aqui foi utilizado a classe Mensagens para melhorar a organização, mais a frente, essa classe será melhor apresentada.

3.3 Metodo getSelect → Retorna o atributo select.

```
52
53      private double getSelect() {
54          return select;
55      }
56
```

3.4 Metodo Select → vfaz testes de validação do valor do tipo inteiro.

```
56
57      private void Select(int select) {
58          this.select = select;
59
60          if (select == NOTAS.OTIMO.code()) {
61              setQuantidade_Otimo();
62          } else {
63              if (select == NOTAS.BOM.code()) {
64                  setQuantidade_Bom();
65              } else {
66                  if (select == NOTAS.REGULAR.code()) {
67                      setQuantidade_Regular();
68                  } else {
69                      if (select == NOTAS.RUIM.code()) {
70                          setQuantidade_Ruim();
71                      } else {
72                          if (select == NOTAS.PESSIMO.code()) {
73                              setQuantidade_Pessimo();
74                          } else {
75                              System.out.println("------------------------");
76                              System.out.println("Digito invalido!");
77                          }
78                      }
79                  }
80              }
81          }
82      }
83
```

Foi tulizado a estrutura de ddecisão FOR ELSE.
Na validação faço uma comparação com uma variavel do tipo inteiro chamada code. Variavel que pode ser acessada pelo metodo code() da classe do tipo ENUM chamada NOTAS.

Caso o valor atribuido a variavel select seja negada em todas as verificações, no ultimo eles será apresentada uma mensagem de digito invalido para o usuario.

3.5 Metodo Media_Idade_Quantidade_Ruim

```
83
84      private double media_Idade_Quantidade_Ruim() {
85
86          if (getQuantidade_Ruim() != ZERO) {
87              double media = getidade() / getQuantidade_Ruim();
88
89              return media;
90          }
91          return 0.0;
92      }
93
```

Nesse metodo é calculado a media das idades dos participantes que selecionaram a opção RUIM. Precisei colocar uma verificação para evitar resultados zeros,

3.6 Metodo porcentagem Ruim

```
93
94      private double percent_Pessimo() {
95
96          if (getQuantidade_Pessimo() != ZERO) {
97
98              double percent_Pessimo = (getQuantidade_Pessimo() / getQuantidade_Participantes()) * PERCENT;
99              return percent_Pessimo;
100         }
101
102         return 0;
103     }
104
```

Metodo que contém o calculo para a porcentagem de pessoas que escolheram a opção Pessimo. Foi utiliado uma estruturad de decisão IF ELSE para evitar resutados zeros.

3.7 Getters and Setters

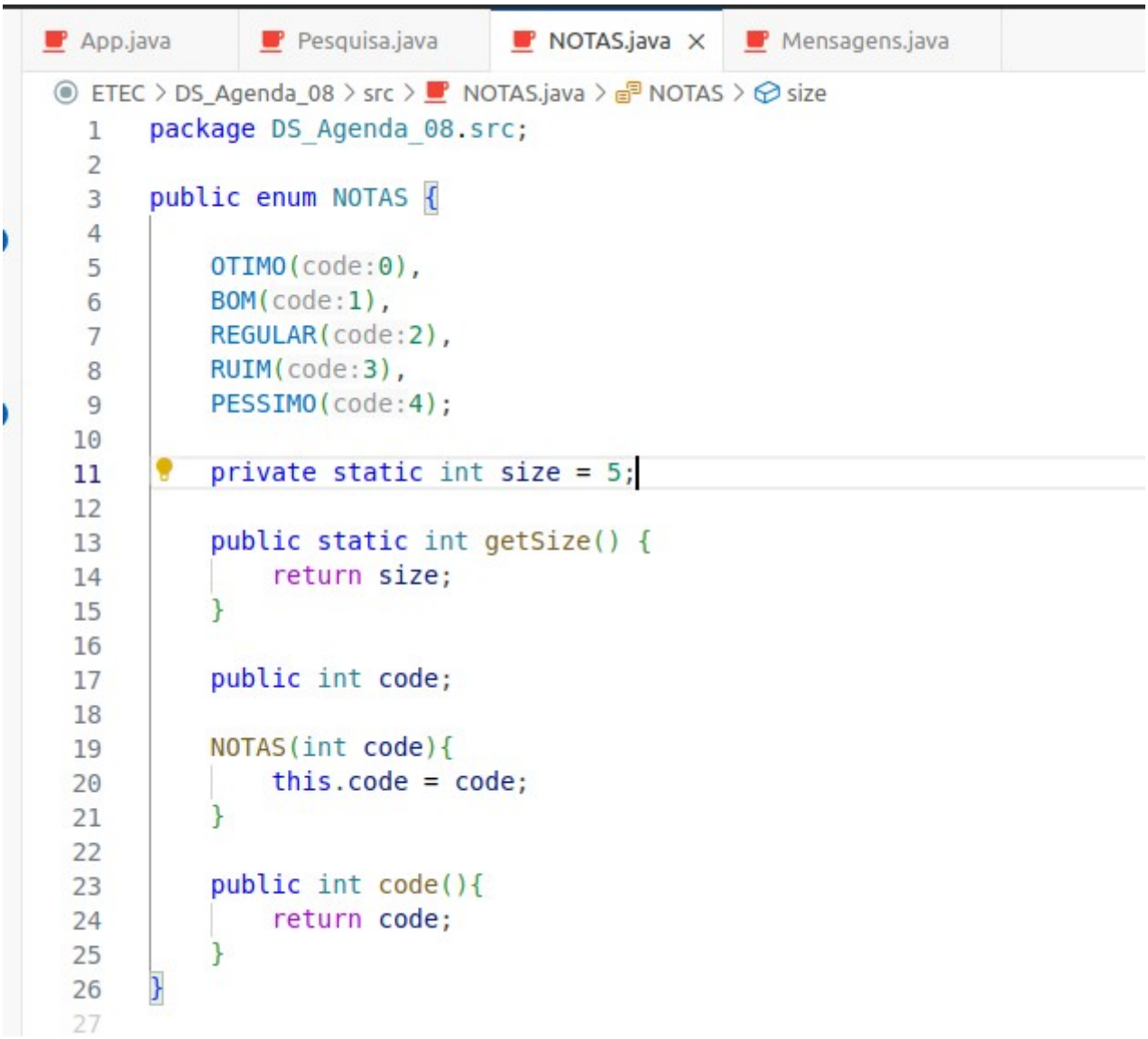
```
105
106
107     // GETTERS AND SETTTERS
108
109     public double getQuantidade_Otimo() {
110         return quantidade_Otimo;
111     }
112
113     private void setQuantidade_Otimo() {
114         quantidade_Otimo++;
115     }
116
117     public double getQuantidade_Bom() {
118         return quantidade_Bom;
119     }
120
121     private void setQuantidade_Bom() {
122         quantidade_Bom++;
123     }
124
125     public double getQuantidade_Regular() {
126         return quantidade_Regular;
127     }
128
```

Metodos para acessar e alterar o valor dos atributos da classe (variaveis no topo do arquivo). Como todos o metodos são privados, não tem sentido de eu ter colocados elas aqui, apenas fiz por objetivos de estufo apenas.

Caso eu não fize-se uso deles eu poderia ter usado o “this.propriedade” ou de forma direta “propriedade”.

Fiz uma par de GET (retornar o valor) e SET (alterar o valor) para cada atributo da classe pesquisa. Não vou colcoar todos aqui pois é desnecessário.

4.classe Notas: enumerados, classe que contem as contantes da regra de negócios.



Classe do tipo ENUM. Enumerados são uma classe que armazenaa um conjunto de constantes que tem o mesmo cenário de uso.

Variavel SIZE → armazena um valor que representa a “quantidade de elementos”. Fiz isso para evitar valores umericos soltos no código.

Metoddo getSIZE → retorna o valor de size.

Variavel CODE → será utiliada para armazenar um valor do tipo inteiro que ira representar umas das possíveis constanstes em AZUL UPPERCASE.

Metodo construtor NOTAS() → Assim que essa classe for chamada e selecionado uma valor constante, será atribuido um dado do tipo inteiro automaticamnte a variavel code;

getCODE → metodo utilizado para acessar a variavel code. (Não podendo alterar).

5. Classe Mensagens: contém metodos de mensagens já pré definidas para interagir com o usuário.

App.java | Pesquisa.java | NOTAS.java | Mensagens.java

ETEC > DS_Agenda_08 > src > Mensagens.java > ...

```java
package DS_Agenda_08.src;

public class Mensagens {

    public static void enter_Idade(){
        System.out.print("--\nEnter a age: ");
    }
    public static void enter_Nota(){
        System.out.println("--\nEnter a note:\n" + Mensagens.notas());
    }

    public static void select(){
        System.out.print("Select: ");
    }

    public static void print_Total_Otimas(double total_Otimas){
        System.out.println("quantidade de respostas ótimo: " + total_Otimas);
    }

    public static void print_Media_Ruim(double media_Ruim){
        System.out.println("média de idade das pessoas que responderam ruim: " + media_Ruim);
    }

    public static void print_Percent_Pessimo(double percent_Pessimo){
        System.out.println("porcentagem de respostas pessimo: " + percent_Pessimo + "%");
    }

    public static String notas(){
        String mensage = String.format("[0]Otimo\n[1]Bom\n[2]Regular\n[3]Ruim\n[4]Pessimo\n");
        return mensage;
    }
}
```

6. TESTE RÁPIDO:

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
Select: 1
pessoa Numero: 40
--
Enter a age: 22
--
Enter a note:
[0]Otimo
[1]Bom
[2]Regular
[3]Ruim
[4]Pessimo

Select: 2
quantidade de respostas ótimo: 3.0
média de idade das pessoas que responderam ruim: 133.33333333333334
porcentagem de respostas pessimo: 22.5%
matheus@lenovo-g4070:~/Documents/ETEC$
```

main* ⊗ 0 ⚠ 0 0 Java: Ready