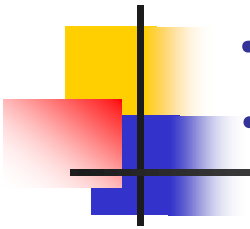
UNIVERSIDADE CATÓLICA DE PERNAMBUCO

Tratamento de Exceções

Universidade Católica de Pernambuco
Ciência da Computação

Prof. Márcio Bueno
poonoite@marciobueno.com

Fonte: Material da Profª Karina Oliveira



Introdução

- Exceções
  - São eventos que ocorrem durante a execução de um programa e quebram o fluxo normal de execução das instruções.
  - Indicam a ocorrência de erros ou condições excepcionais no programa.

Sem Tratamento de Exceções

- Exemplo 1:

Erro em tempo de execução!

```
public class Divide {
     public static void main (String args[]) {
          int dividendo = Integer.parseInt(args[0]);
          int divisor = Integer.parseInt(args[1]);
          divide(dividendo, divisor);
     }

     public static void divide(int dividendo, int divisor) {
        System.out.println("Divisão = " + (dividendo/divisor));
     }
}
```

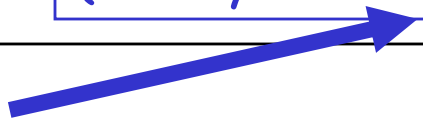
ArithmeticException lançada!

Exception in thread "main" java.lang.ArithmeticException: by zero

Sem Tratamento de Exceções

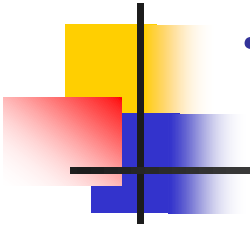
- Exemplo 2:

Erro em tempo de execução!
(ArrayIndexOutOfBoundsException)



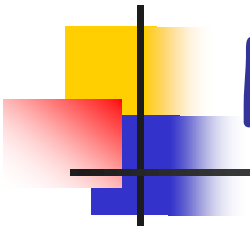
```
public class Divide {
    public static void main (String args[]) {
        int dividendo = Integer.parseInt(args[0]);
        int divisor = Integer.parseInt(args[1]);
        divide(dividendo, divisor);
    }

    public static void divide(int dividendo, int divisor) {
        System.out.println("Divisão = " + (dividendo/divisor));
    }
}
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException



Tipos de Exceções

- Erros aritméticos;
- Estouro de limite de array;
- Entrada de dados inválidos;
- Erros na manipulação de arquivos;
- Erros na comunicação com bancos de dados;
- Falhas de comunicação entre programas distribuídos;
- Entre outros.



Palavras Reservadas

- Em Java:
  - **`try, catch`** e **`finally`**
    - Define um bloco de tratamento de exceção.
  - **`throws`**
    - Declara que um método pode lançar uma exceção ou mais exceções.
  - **`throw`**
    - Lança uma exceção.

Tratamento de Exceções

- Usando `try`, `catch` e `finally`
  - Define um bloco de tratamento de exceção.

```
try {
      ...
} catch (Excecao1 e1) {
      ...
} catch (Excecao2 e2) {
      ...
} finally {
      ...
}
```

Tratamento de Exceções

- Usando `try` - `catch` - Exemplo 1:

```
public class Divide {
     public static void main (String args[]) {
          int dividendo = Integer.parseInt(args[0]);
          int divisor = Integer.parseInt(args[1]);
          try {
             divide(dividendo, divisor);
          } catch (ArithmeticException e) {
             System.out.println("Não pode dividir por zero!");
          }
     }
     public static void divide(int dividendo, int divisor) {
          System.out.println("Divisão = " + (dividendo/divisor));
     }
}
```

Tratamento de Exceções

- Usando `try` - `catch` - Exemplo 2:

```
public class Divide {
    public static void main (String args[]) {
        try {
            int dividendo = Integer.parseInt(args[0]);
            int divisor = Integer.parseInt(args[1]);
            divide(dividendo, divisor);
        } catch (ArithmeticException e1) {
            System.out.println("Não pode dividir por zero!");
        } catch (ArrayIndexOutOfBoundsException e2) {
            System.out.println("Favor Informar dois números!");
        }
    }
    public static void divide(int dividendo, int divisor) {
        System.out.println("Divisão = " + (dividendo/divisor));
    }
}
```

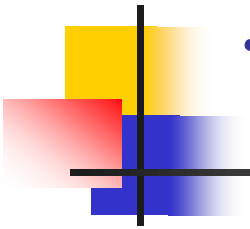
Usando `try - catch` - Exemplo 3:

```
import javax.swing.JOptionPane;
public class Divide {
    public static void main (String args[]) {
        int dividendo, divisor;
        while(true){
            try {
                dividendo = Integer.parseInt(JOptionPane.showInputDialog
                ("Digite o dividendo: "));
                divisor = Integer.parseInt(JOptionPane.showInputDialog
                ("Digite o divisor: "));
                divide(dividendo, divisor);
                break;
            } catch (ArithmeticException e1) {
                JOptionPane.showMessageDialog(null,
                "Não pode dividir por zero!");
            } catch (NumberFormatException e2) {
                JOptionPane.showMessageDialog(null,
                "Favor informar apenas números inteiros!");
            }
        }
    }
    /* Método divide aqui */
}
```

Tratamento de Exceções

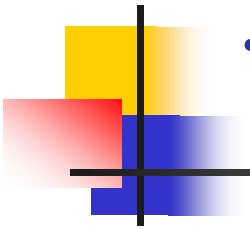
- Usando `try` - `catch` - Exemplo 4:

```
public class OperacoesMatematicas {
 public static void main (String args[ ]) {
  System.out.println("Início do Programa.");
  try {
   int a= Integer.parseInt(JOptionPane.showInputDialog("a:"));
   int b= Integer.parseInt(JOptionPane.showInputDialog("b:"));
   System.out.println("Divisão = " + (a / b));
   System.out.println("Multiplicação = " + (a * b));
   System.out.println("Soma = " + (a + b));
   System.out.println("Subtração = " + (a - b));
  } catch (ArithmeticException e1) {
   System.out.println("Não pode dividir por zero!");
  } catch (NumberFormatException e2) {
   System.out.println("Digite um número válido!");
  }
  System.out.println("Fim do Programa.");
 }
}
```



Tratamento de Exceções

- Usando `throws`
  - Declara que um método pode lançar uma ou mais exceções.
  - Um método Java pode lançar uma exceção se encontrar uma situação com a qual ele não possa lidar;
  - Um método deve informar ao compilador os parâmetros que ele recebe, o valor que ele retorna e também o que pode acontecer de errado usando `throws`.



Tratamento de Exceções

- Usando throws
  - Sintaxe:

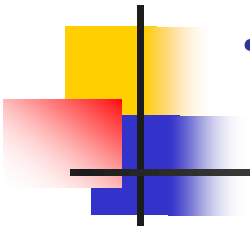
```
public void metodo( ) throws Excecao {
        ...
}
```

```
public void metodo( ) throws Excecao1, Excecao2 {
        ...
}
```

Tratamento de Exceções

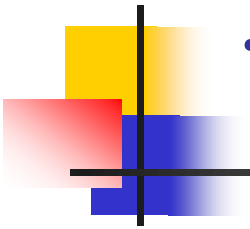
- Usando throws – Exemplo1:

```
import java.io.FileWriter;

public class GravaArquivo {
      public static void grava(String texto) {
         FileWriter fw = new FileWriter("teste.txt");
         fw.write(texto);
         fw.close( );
      }
}
```



Tratamento de Exceções

- Usando throws – Exemplo 1 (Cont.):
  - O trecho de código anterior irá causar um erro de compilação.
  - O compilador exige que se declare a exceção IOException na cláusula throws do método ou que a mesma seja tratada dentro do método!



Tratamento de Exceções

- Usando throws – Exemplo 1 (Cont.):

```
import java.io.FileWriter;

public class GravaArquivo {
    public static void grava(String texto) throws IOException {
        FileWriter fw = new FileWriter("teste.txt");
        fw.write(texto);
        fw.close( );
    }
}
```

Tratamento de Exceções

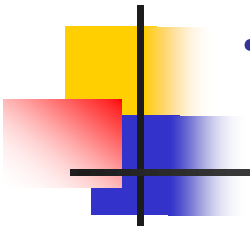
- Usando throws – Exemplo 1 (Cont.):

```
import java.io.FileWriter;
import java.io.IOException;
public class GravaArquivo {
   public static void main(String args[]) {
      try {
         grava(args[0]);
      } catch (IOException e) {
         System.out.println("Erro ao gravar arquivo!");
         System.out.println(e); // Imprime detalhadamente a Exceção.
         e.printStackTrace( ); // Imprime detalhadamente a Exceção.
      }
   }
}
```

Mais alguma exceção deveria ser tratada nesse código?

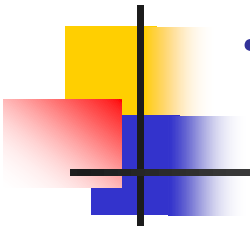
Usando throws – Exemplo 1 (Cont.):

```
import java.io.FileWriter;
import java.io.IOException;
public class GravaArquivo {
    public static void grava(String texto) throws IOException{
        FileWriter fw = new FileWriter("teste.txt");
        fw.write(texto);
        fw.close( );
    }
    public static void main(String args[]) {
        try {
           grava(args[0]);
        }catch(IOException e) {
           System.out.println("Erro ao gravar arquivo!");
           System.out.println(e); // Imprime detalhadamente a Exceção.
           e.printStackTrace( ); // Imprime detalhadamente a Exceção.
        }catch(ArrayIndexOutOfBoundsException e){
           System.out.println("Você deve digitar algo para gravar no arquivo!");
           System.out.println(e); // Imprime detalhadamente a Exceção.
           e.printStackTrace( ); // Imprime detalhadamente a Exceção.
        }
   }
}
```



Tratamento de Exceções

- Usando throws – Exemplo 1 (Cont.):
  - No código anterior, caso não haja argumentos:
    - Você deve digitar algo para gravar no arquivo!
    - java.lang.ArrayIndexOutOfBoundsException: 0
    - java.lang.ArrayIndexOutOfBoundsException: 0
      at GravaArquivo.main(GravaArquivo.java:11)



Tratamento de Exceções

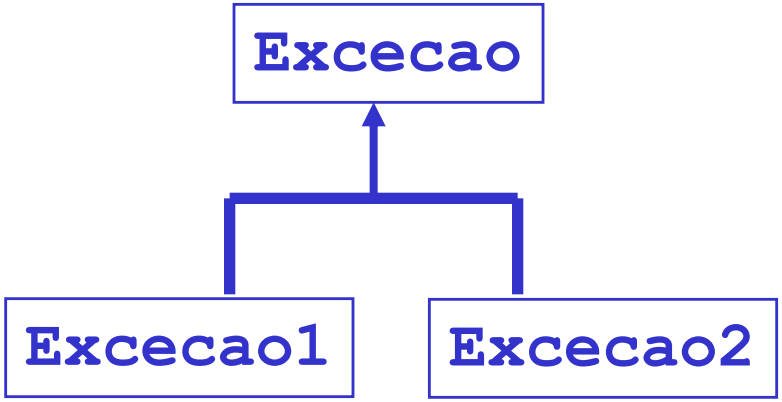
- Usando `finally`
  - O trecho de código colocado dentro da cláusula *finally* sempre será executado, independente do código que pode lançar exceção executar com sucesso ou com erro.

Usando `finally` - Exemplo:

```
import java.io.FileWriter;
import java.io.IOException;
public class GravaArquivo2 {
   public static void main(String args[ ]) {
      FileWriter fw = null;
      try {
        fw = new FileWriter("teste.txt");
        fw.write(args[0]);
      } catch (IOException e) {
            System.out.println("Erro ao gravar arquivo!");
            System.out.println(e); // Imprime detalhes da Exceção.
      } finally {
           try{
               fw.close();
            }catch(IOException e){
               System.out.println("Erro ao fechar arquivo!");
               System.out.println(e); // Imprime detalhes da Exceção.
            }
      }
}
```

Lançando e tratando Exceções pela Superclasse

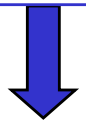
- Exemplo 1: Excecao1 e Excecao2 são subclasses de Excecao!



Lançando e tratando Exceções pela Superclasse

- Exemplo 1 (Cont.):
  - As exceções declaradas com throws podem ser superclasses das exceções que são realmente lançadas:

```
public void metodo throws Excecao1, Excecao2 {
        ...
}
```



```
public void metodo throws Excecao {
        ...
}
```

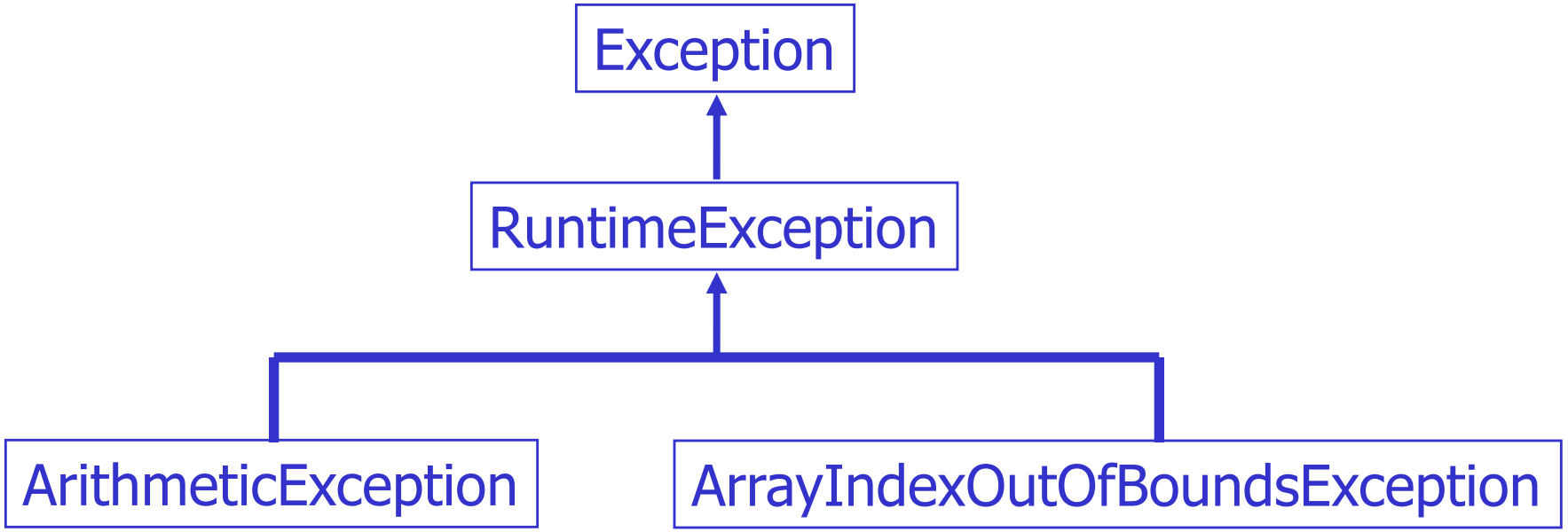
Lançando e tratando Exceções pela Superclasse

- Exemplo 2: Tratando pelas subclasses

```
public class Divide {
     public static void main (String args[]) {
          try {
              int dividendo = Integer.parseInt(args[0]);
              int divisor = Integer.parseInt(args[1]);
              divide(dividendo, divisor);
          } catch (ArithmeticException e1) {
              System.out.println("Não pode dividir por zero!");
          } catch (ArrayIndexOutOfBoundsException e2) {
              System.out.println("Favor Informar dois números!");
          }
     }
     public static void divide(int dividendo, int divisor) {
          System.out.println("Divisão = " + (dividendo/divisor));
     }
}
```

Lançando e tratando Exceções pela Superclasse

- Exemplo 2: Tratando pela superclasse
  - **`ArithmeticException`** e **`ArrayIndexOutOfBoundsException`** são subclasses de **`Exception`**!

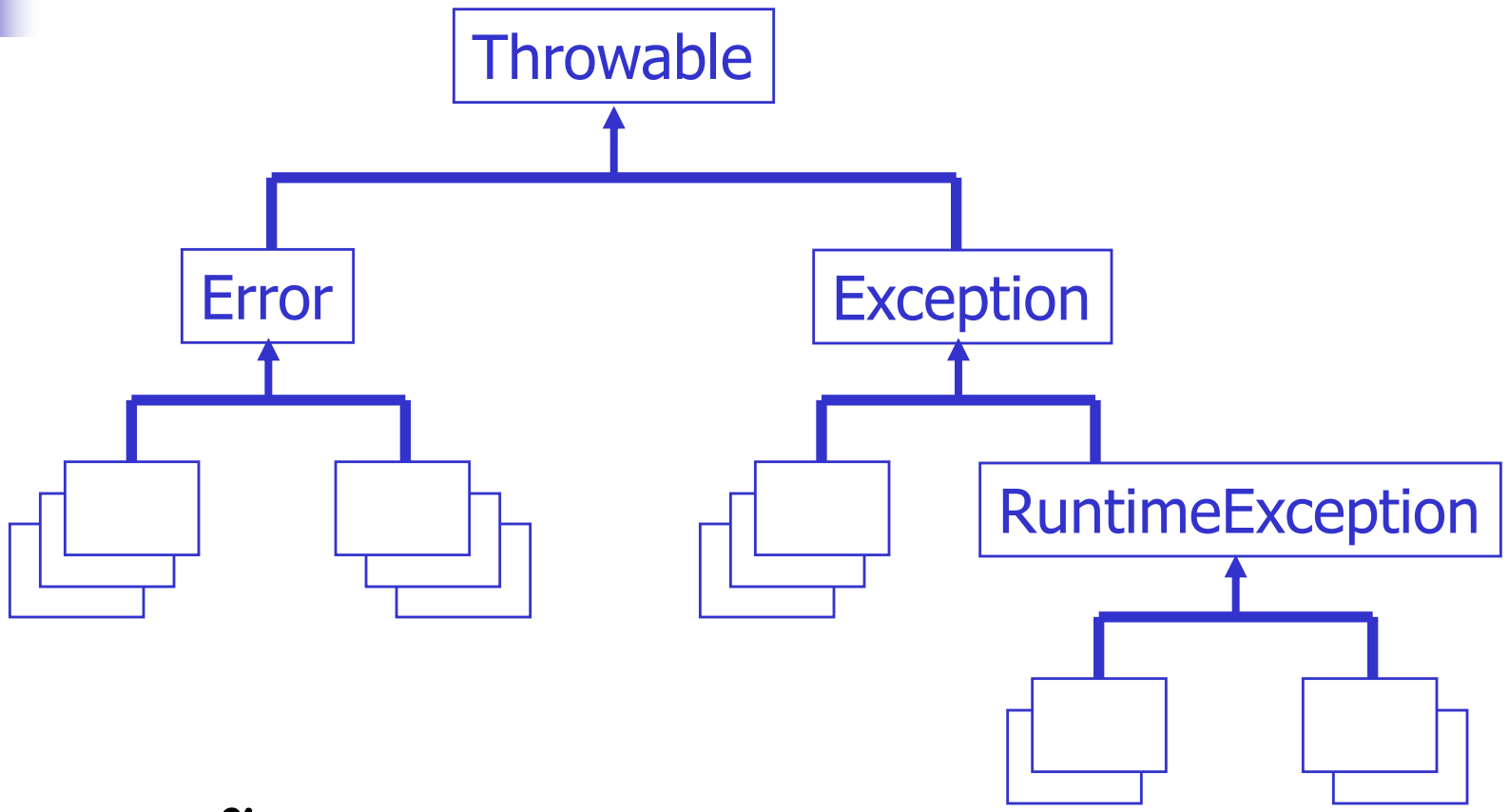


Lançando e tratando Exceções pela Superclasse

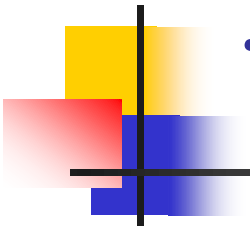
- Exemplo 2: Tratando pela superclasse

```
public class Divide {
    public static void main (String args[]) {
        try {
            int dividendo = Integer.parseInt(args[0]);
            int divisor = Integer.parseInt(args[1]);
            divide(dividendo, divisor);
        } catch (Exception e) {
            System.out.println("Erro ao executar programa!");
        }
    }
    public static void divide(int dividendo, int divisor) {
        System.out.println("Divisão = " + (dividendo/divisor));
    }
}
```

Hierarquia de Exceções Java



OBSERVAÇÃO: O compilador não exige que se declare ou trate exceções de qualquer subclasse de Error ou de RuntimeException.



Tratamento de Exceções

- Usando throw
  - Palavra reservada utilizada para lançar uma exceção.
    - Exemplo 1:

```
// Instanciando e lançando o objeto Exception
throw new Exception("Mensagem de ERRO!");
```

    - Exemplo 2:

```
// Instanciação do objeto Exception
Exception e = new Exception("Mensagem de ERRO!");
// Lançando a exceção
throw e;
```

Definindo suas Próprias Exceções

- Um programa pode ter um problema que não esteja descrito adequadamente em nenhuma das classes de exceções
- Criar sua própria exceção como uma subclasse da classe `Exception`
- Exemplo:

```
public class ExcecaoTextoInvalido extends Exception {
        public ExcecaoTextoInvalido(String mensagem) {
                super(mensagem);
        }
}
```

Definindo suas Próprias Exceções

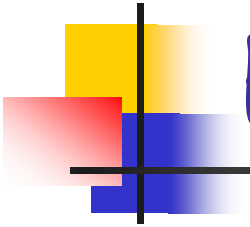
- Exemplo (cont.):

```
public static void grava(String texto) throws IOException, ExcecaoTextoInvalido {
        FileWriter fw = null;
        if (texto == null || texto.trim( ).equals("")) {
                throw new ExcecaoTextoInvalido("Texto inválido: " + texto);
        } else {
                fw = new FileWriter("teste.txt");
                fw.write(texto);
        }
}
```

Definindo suas Próprias Exceções

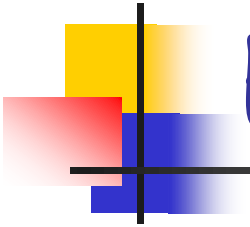
- Exemplo (cont.):

```
public static void main(String[ ] args) {
        String texto = JOptionPane.showInputDialog("Digite um texto:");
        try {
                grava(texto);
        } catch (IOException e) {
                e.printStackTrace();
        } catch (ExcecaoTextoInvalido e) {
                System.out.println(e.getMessage());
        }
}
```



Exercícios

- Exercício 1:
  - Implementar a exceção ExcecaoDivisaoPorZero. Esta exceção será lançada pelo método double calcula(double a, double b) da classe Divisao.
  - Implementar também a aplicação AplicacaoDivisao que recebe dois números informados pelo usuário e faz a divisão do primeiro pelo segundo usando uma chamada ao método calcula da classe Divisao.



Exercícios

- Exercício 2: Implementar o tratamento de exceções para o Cadastro de Contas.
  - Os métodos set das classes básicas de negócio lançarão exceções do tipo ExcecaoDadoInvalido quando o dado passado como parâmetro não for válido.
  - O método inserir da classe CadastroContas deve lançar a exceção ExcecaoRepositorio quando não puder mais inserir contas no *array* e a exceção ExcecaoElementoJaExistente quando uma conta com um mesmo número já estiver cadastrada.
  - O método buscar da classe CadastroContas deve lançar a exceção ExcecaoElementoInexistente quando a conta que se deseja buscar não estiver cadastrada.