














-Compartilhar o seu link com: luciorocha @ professores.utfpr.edu.br

Filipe Augusto Parreira Almeida <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Plinio Koyama e Raphael Uematsu <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
João Pedro Cavani Meireles<  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Matheus Hirata & Thiago Cristovão <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Isabella Melo Almeida<  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Angélica <  Cópia Aula 17 - Exercícios propostos >
Daniel Martins de Carvalho <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Séfora<https://docs.google.com/document/d/1IOo_Ryy3dCbydMDDI3jTo3xDsi_A8puYgpZjqOOAM50/edit>
Guilherme Ramalho <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Deivid da Silva Galvão <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
João Vitor N. Yoshida <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Thiago Tieghi e Pedro Reis: <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Rafael Kendy Naramoto Lopes <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
mabyllly <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Gabriel Takeshi <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Henrique Cois <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Roberto Furlani Neto <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Alexandre Aparecido <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Lucas Viana e Victor Ramos<  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Fernando Rafael  Aula 17 - Exercícios propostos >
João Pedro de Paula//Gabriel Reis:<  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Vitor Hugo Leite A. de Oliveira <  Aula 17 - Exercícios propostos >
Julio Farias <  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Rodrigo Leandro Benedito:  Cópia de POCO4A - Aula 17 - Exercícios propostos >
Felipe Antonio Magro:  Cópia de POCO4A - Aula 17 - Exercícios propostos >

//Exemplo de Sala

```
package execucao;  
  
import java.util.InputMismatchException;  
import java.util.Scanner;  
public class Execucao {  
  
    int numerador=0;  
    int denominador=0;  
    float resultado=0;
```

```
//MinhaExcecao eh uma Excecao nao-verificada
public class MinhaExcecao extends ArithmeticException {

    public MinhaExcecao(){
        super();
    }

    public int corrigir(){

        return 1;

    }

}

//Excecao nao-verificada: RuntimeException e subclasses
//throws: 'pode' disparar uma excecao
//Excecao verificada: Excecao e subclasses
public float leitura()
    throws MinhaExcecao {

    Scanner entrada = new Scanner(System.in);
    System.out.print("Numerador: ");
    numerador = entrada.nextInt();

    System.out.print("Denominador: ");
    denominador = entrada.nextInt();

    if( denominador == 0 )
        throw new MinhaExcecao(); //vai acontecer

    float resultado = numerador / denominador;

    entrada.close();

    return resultado;

}

public Excecao(){
```

```
try {

    resultado = leitura();
} catch (MinhaExcecao e) {
    System.out.println("EXCECAO PERSONALIZADA");

    denominador = e.corrigir();
    resultado = numerador / denominador;

} finally {

    System.out.println("Resultado: " + resultado);

}

/*catch( ArithmeticException e ){
    e.printStackTrace();
    System.out.println("EXCECAO 1: " + e.getMessage());
} catch( InputMismatchException e ){
    //e.printStackTrace();
    System.out.println("EXCECAO 2: " + e.getMessage());
} catch( Exception e ){
    //e.printStackTrace();
    System.out.println("EXCECAO 3: " + e.getMessage());
} finally {
    System.out.println("Continua...");
}*/

}

public static void main(String[] args) {
    new Excecao();
}

}
```

1) Exemplo de Tratamento de Exceções:

```
//Leitura de dados
//Descoberta do tipo da Exception

import java.util.Scanner;
public class Principal {

    public void iniciar(){

        Scanner entrada = new Scanner(System.in);
        int numerador=0;
        int denominador=0;
        float resultado=0;
        try {
            System.out.println("\nNumerador: ");
            numerador = entrada.nextInt();
            System.out.println("\nDenominador: ");
            denominador = entrada.nextInt();
            resultado = numerador / denominador;
        } catch( ArithmeticException e ) {
            System.out.println("2: Divisao por zero");
        } catch(InputMismatchException e){
            System.out.println("1: Excecao na entrada de dados.");
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
        System.out.println("Programa continua");
    }

    public static void main( String [ ] args ){

        Principal principal = new Principal();
        principal.iniciar();
    }
}
```

//2) Exemplo de tratamento de excecoes com metodos

```
//Palavra-reservada throw
```

```
import java.util.Scanner;
public class Principal {

    public void leitura() {
        Scanner entrada = new Scanner(System.in);

        int numerador=0;
        int denominador=0;
        float resultado=0;

        System.out.println("\nNumerador: ");
        numerador = entrada.nextInt();
        System.out.println("\nDenominador: ");
        denominador = entrada.nextInt();
        if ( denominador == 0 )
            throw new ArithmeticException();
        else
            resultado = numerador / denominador;
    }

    public void iniciar(){

        try {
            leitura( );
        } catch( ArithmeticException e ) {
            System.out.println("2: Divisao por zero");
        } catch(InputMismatchException e){
            System.out.println("1: Excecao na entrada de dados.");
            System.out.println( e.getMessage() );

            e.printStackTrace();
        }
        System.out.println("Programa continua");
    }

    public static void main( String [ ] args ){

        Principal principal = new Principal();
        principal.iniciar();
    }
}
```

```
}
```

//3) Exemplo com múltiplas capturas (try-catch) de excecao:

//Acesso a posição não definida no Vetor estático

```
import java.util.Scanner;
public class Principal {

    public void leitura() {
        Scanner entrada = new Scanner(System.in);

        int numerador=0;
        int denominador=0;
        float resultado=0;

        System.out.println("\nNumerador: ");
        numerador = entrada.nextInt();
        System.out.println("\nDenominador: ");
        denominador = entrada.nextInt();
        if ( denominador == 0 )
            throw new ArithmeticException();
        else
            resultado = numerador / denominador;
    }

    public void iniciar(){

        Integer [ ] vetor = new Integer[2];

        int b = 0;

        try {
            b = vetor[20];
            leitura( );
        }
        catch ( ArrayIndexOutOfBoundsException e ){
            System.out.println("3: Erro de indice");
            b = 0;
        }
    }
}
```

```

        catch( ArithmeticException e ) {
            System.out.println("2: Divisao por zero");
        } catch( InputMismatchException e){
            System.out.println("1: Excecao na entrada de dados.");
            System.out.println( e.getMessage() );

            e.printStackTrace();
        }
        System.out.println("Programa continua");
    }

    public static void main( String [ ] args ){

        Principal principal = new Principal();
        principal.iniciar();
    }
}

```

//4) Excecao com ordem de captura (catch): Mais específico (subclasses) primeiro → Mais genéricos (superclasses)

```

import java.util.Scanner;
public class Principal {

    public void leitura() {
        Scanner entrada = new Scanner(System.in);

        int numerador=0;
        int denominador=0;
        float resultado=0;

        System.out.println("\nNumerador: ");
        numerador = entrada.nextInt();
        System.out.println("\nDenominador: ");
        denominador = entrada.nextInt();
        if ( denominador == 0 )
            throw new ArithmeticException();
        else
            resultado = numerador / denominador;
    }
}

```

```

}

public void iniciar(){

    Integer [ ] vetor = new Integer[2];

    int b = 0;

    try { vetor[0]=1;
        b = vetor[0];
        leitura( );
    } catch (InputMismatchException e ){
        System.out.println("1: Excecao na entrada de dados");
    }
    catch ( Exception e ){
        System.out.println("4: Excecao generica");
    }

    System.out.println("Programa continua");
}

public static void main( String [ ] args ){

    Principal principal = new Principal();
    principal.iniciar();
}
}

```

//5) Exemplo de tratamento de exceção personalizado:

```

import java.util.Scanner;
public class Principal {

    public class MinhaExcecao extends ArithmeticException {

    }

    public void leitura() {
        Scanner entrada = new Scanner(System.in);

```



```

int numerador=0;
int denominador=0;
float resultado=0;

    System.out.println("\nNumerador: ");
    numerador = entrada.nextInt();
    System.out.println("\nDenominador: ");
    denominador = entrada.nextInt();
    if ( denominador == 0 )
        throw new MinhaExcecao();
    else
        resultado = numerador / denominador;
}

public void iniciar(){

    Integer [ ] vetor = new Integer[2];

    int b = 0;

    try { vetor[0]=1;
        b = vetor[0];
        leitura( );
    } catch (MinhaExcecao e ){
        System.out.println("1: Excecao na entrada de dados");
    }
    catch ( Exception e ){
        System.out.println("4: Excecao generica");
    }

    System.out.println("Programa continua");
}

public static void main( String [ ] args ){

    Principal principal = new Principal();
    principal.iniciar();
}
}

```

//6) Exemplo de tratamento de exceção personalizada com mensagem personalizada

```
import java.util.Scanner;
```

```
public class Principal {

    public class MinhaExcecao extends ArithmeticException {

        public MinhaExcecao( String mensagem ){
            super(mensagem);
        }

    }

    public void leitura() {
        Scanner entrada = new Scanner(System.in);

        int numerador=0;
        int denominador=0;
        float resultado=0;

        System.out.println("\nNumerador: ");
        numerador = entrada.nextInt();
        System.out.println("\nDenominador: ");
        denominador = entrada.nextInt();
        if ( denominador == 0 )
            throw new MinhaExcecao("1: Excecao - divisao por zero");
        else
            resultado = numerador / denominador;
    }

    public void iniciar(){

        Integer [ ] vetor = new Integer[2];

        int b = 0;

        try { vetor[0]=1;
            b = vetor[0];
            leitura( );
        } catch (MinhaExcecao e ){
            System.out.println( e );
        }
        catch ( Exception e ){
            System.out.println("4: Excecao generica");
        }

        System.out.println("Programa continua");
    }
}
```

```

public static void main( String [ ] args ){

    Principal principal = new Principal();
    principal.iniciar();
}
}

```

//7) Como corrigir entradas incorretas?

```

import java.util.Scanner;
public class Principal {

    public class MinhaExcecao extends ArithmeticException {

        public MinhaExcecao( String mensagem ){
            super(mensagem);
        }
        public int corrigir(int denominador){
            int result=0;
            if( denominador == 0 )
                result = -1;
            return result;
        }
    }

}

public void leitura(int denominador) {
    Scanner entrada = new Scanner(System.in);

    int numerador=0;

    float resultado=0;

    System.out.println("\nNumerador: ");
    numerador = entrada.nextInt();

    if ( denominador == 0 )
        throw new MinhaExcecao("1: Excecao - divisao por zero");
}

```

```

        else
            resultado = numerador / denominador;
    }

    public void iniciar(){

        Integer [ ] vetor = new Integer[2];

        int b = 0;

        try {

            leitura( b );
        } catch (MinhaExcecao e ){
            System.out.println( e );
            b = e.corrigir( b );
            leitura( b ); //Programa em um estado valido.
        }
        catch ( Exception e ){
            System.out.println("4: Excecao generica");
        }

        System.out.println("Programa continua");
    }

    public static void main( String [ ] args ){

        Principal principal = new Principal();
        principal.iniciar();
    }
}

```

//8) Classes de Exceções: Exemplo de exceções verificadas e não-verificadas:

```
import java.util.Scanner;
```

```

public class Principal {

    public class ExcecaoVerificada extends Exception {

        public ExcecaoVerificada(String mensagem){
            super(mensagem);
        }
    }

    public class MinhaExcecao extends ArithmeticException {

        public MinhaExcecao( String mensagem ){
            super(mensagem);
        }
        public int corrigir(int denominador){
            int result=0;
            if( denominador == 0 )
                result = -1;
            return result;
        }
    }

}

//throws: metodos → "pode" disparar uma Exception
//throw: ação → "vai" disparar a Exception

public void leitura(int denominador) throws ExcecaoVerificada {
    Scanner entrada = new Scanner(System.in);

    int numerador=0;

    float resultado=0;

    System.out.println("\nNumerador: ");
    numerador = entrada.nextInt();

    if ( denominador == 0 )
        throw new ExcecaoVerificada("1: Excecao - divisao por zero");
    else
        resultado = numerador / denominador;
}

public void iniciar(){

    Integer [ ] vetor = new Integer[2];

```

```

int b = 0;

try { //Obrigatorio usar try-catch

    leitura( b );
} catch (ExcecaoVerificada e ){
    System.out.println( e );
    //b = e.corrigir( b );
    try {
        leitura( b ); //Programa em um estado valido.
    } catch (ExcecaoVerificada e2 ){
        System.out.println( e2 );
    }
}

System.out.println("Programa continua");
}

public static void main( String [ ] args ){

    Principal principal = new Principal();
    principal.iniciar();
}
}

```

//Exemplo de captura de exceção com método de instância

//Exemplo de captura de múltiplas exceções

//Exemplo de exceção personalizada

//Exemplo de encadeamento de exceções:

//Exemplo de várias capturas de exceções com try-catch

Exercícios propostos:

- 1) Implemente o tratamento de exceções no trecho do cálculo no código a seguir:

```
import java.util.Scanner;

public class TratamentoExcecao1 {

    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.print("\nDigite o numerador: ");
        int numerador = obj.nextInt();
        System.out.print("\nDigite o denominador: ");
        int denominador = obj.nextInt();
        //Aritmetica de inteiros: nao eh permitida divisao por zero
        int resultado = numerador / denominador;
        //double resultado = (double) numerador / denominador;
        System.out.println("\nResultado: " + resultado);
    }
}
```

- a) Crie um método de leitura de dados do usuário que capture a exceção não-verificada `InputMismatchException`. Corrija a entrada inválida.

```
//Filipe Augusto Parreira Almeida
import java.util.Scanner;

public class TratamentoExcecao1 {

    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.print("\nDigite o numerador: ");
        int numerador;
        try {
            numerador = obj.nextInt();
        } catch( InputMismatchException e ){
            e.printStackTrace();
            numerador = 1;
        }

        System.out.print("\nDigite o denominador: ");
        int denominador = obj.nextInt();
        //Aritmetica de inteiros: nao eh permitida divisao por zero
        int resultado;
        try{
            resultado = numerador / denominador;

        } catch (ArithmeticException e){
            System.out.println(e.getMessage());
        }

        //double resultado = (double) numerador / denominador;
        System.out.println("\nResultado: " + resultado);
    }
}
```

- b) Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada `ArithmeticException` quando o denominador=0. Corrija a entrada inválida.


```
//Isabella Melo Almeida

import java.util.Scanner;

public class TratamentoExcecao1 {
    private int numerador;
    private int denominador;
    private float resultado;

    public void leitura() throws ArithmeticException {

        Scanner obj = new Scanner(System.in);
        System.out.print("\nDigite o numerador: ");

        try {
            numerador = obj.nextInt();
        } catch ( InputMismatchException e ){
            e.printStackTrace();
            numerador = 1;
        }

        System.out.print("\nDigite o denominador: ");
        denominador = obj.nextInt();

        if (denominador == 0)
            throw new ArithmeticException();

        //Aritmetica de inteiros: nao eh permitida divisao por zero

        try{
            resultado = numerador / denominador;

        } catch (ArithmeticException e){
            System.out.println(e.getMessage());
        }

        //double resultado = (double) numerador / denominador;
        System.out.println("\nResultado: " + resultado);
    }

    public TratamentoExcecao1(){
```

```

        try {
            leitura();
        } catch( ArithmeticException e ){
            System.out.println("Passei por aqui");
            denominador = 1;
        }
    }

    public static void main(String[] args) {
        new TratamentoExcecao1();
    }
}

```

- c) Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada personalizada do tipo `ArithmeticException` quando o `denominador=0`. Corrija a entrada inválida.

```
//Joao Vitor
```

- d) Crie um método de leitura de dados do usuário que possa disparar uma exceção verificada `Exception` quando o `denominador=0`. Corrija a entrada inválida.

```
//Lucas dos Reis
```

- e) Crie um método de leitura de dados do usuário que possa disparar uma exceção verificada personalizada do tipo `Exception` quando o `denominador=0`. Corrija a entrada inválida.

```
//Matheus Hirata
import java.util.Scanner;
```

```
public class Excecao {

    int numerador = 0;
    int denominador = 0;
    float resultado = 0;

    public class MinhaExcecao extends Exception{

        public MinhaExcecao(){
            super();
        }

        public int corrigir(){
            return 1;
        }
    }

    public float leitura() throws MinhaExcecao{

        Scanner entrada = new Scanner(System.in);

        System.out.print("Numerador: ");
        numerador = entrada.nextInt();

        System.out.print("Denominador: ");
        denominador = entrada.nextInt();

        if( denominador == 0 )
            throw new MinhaExcecao();

        float resultado = numerador / denominador;

        entrada.close();

        return resultado;
    }

    public Excecao(){

        try{

            resultado = leitura();

        }catch(MinhaExcecao e){

            System.out.println("essessao");
```

```

        denominador = e.corrigir();
        resultado = numerador / denominador;
    }
}

public static void main(String[] args) {
    new Excecao();
}
}

```

- f) Ilustre um exemplo de captura seletiva das exceções: `Exception`, `ArrayListOutOfBoundsException` e `FileNotFoundException`.

```
//Rafael Kendy
```

2) Implemente o tratamento de exceções no trecho do cálculo no código a seguir:

```

public class Principal {

    public void iniciar(){

        //Leitura de nome do usuario
        //Leitura de CPF do usuario

    }

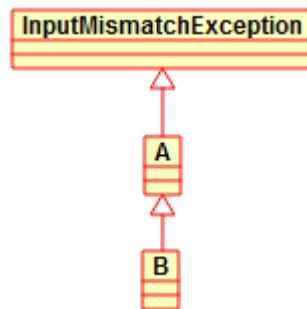
    public static void main(String [ ] args){
        Principal principal = new Principal();
        principal.iniciar();
    }
}

```

- Crie um método de leitura de dados do usuário que capture a exceção não-verificada `InputMismatchException`. Corrija a entrada inválida.
- Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada `ArithmeticException` quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.
- Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada personalizada do tipo `ArithmeticException` quando o quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.

- d) Crie um método de leitura de dados do usuário que possa disparar uma exceção verificada Exception quando o quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.
- e) Crie um método de leitura de dados do usuário que possa disparar uma exceção verificada personalizada do tipo Exception quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.
- f) Ilustre um exemplo de captura seletiva das exceções: Exception, InputMismatchException e NullPointerException.

3) Observe o diagrama UML a seguir:



- a) Crie um método de leitura de dados do usuário que capture a exceção do tipo A.
- b) Crie um método de leitura de dados do usuário que capture a exceção do tipo B.
- c) Crie um método de leitura de dados do usuário que capture a exceção do tipo A. A seguir, dispare a exceção do tipo B e faça a captura.

4) Modifique o programa anterior para que a superclasse seja uma classe de Exceção verificada.