Compartilhar o seu link com: luciorocha @ professores.utfpr.edu.br
Andrei Fernandes Zani: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Lucas Ribeiro Penna Maroja: POCO4A - Aula 18 - 18/10/2022
Gustavo Naoki Jodai Kurozawa: Aula 18 - 18/10/2022 - Exercícios propostos
Michael Pariz Pereira: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Guilherme Ramalho: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Gabriel Candelária: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Camila Costa: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Matheus H. A. Pereira: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
ArithmeticException
Erik Noda: Aula18
Raphael Uematsu: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Felipe Antonio Magro: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Rodrigo Leandro Benedito:
Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
William Eizo Hatakeyama:
Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Plinio Koiama: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Alexandre Calisto: Aula 18
Lucas Prado: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Gabriel Takeshi: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Leonardo G. Fagote Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Iago Macarini: Aula 18 - POO
Arnald: Aula 18 - 18/10
Maria Pedroso: POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Gustavo Nunes: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Ruan Perondi Urbanjos: Copy of POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
João Pedro de Paula : Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
Mariana Naves : Aula 18 - POO
Lucas Santana - Aula 18 - 18/10/2022 - Exercícios propostos
Vinicius Letroche: Cópia de POCO4A - Aula 18 - 18/10/2022 - Exercícios propostos
MARIA EDUARDA PEDROSO
Exercícios propostos:

Parte 1

1) (Online) Descoberta da Exceção: Acesse o link da atividade e faça a implementação do código-fonte conforme solicitado: https://codeboard.io/projects/351116

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
        super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

       int numerador=0;
```

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

   public void iniciar(){

         int numerador=0;
         int denominador=0;
         float resultado=0;
          try {
            resultado = numerador / denominador;
          } catch ( ArithmeticException objeto){
             System.out.println(
                "1) Excecao: divisao por zero." + objeto.getMessage());
                objeto.printStackTrace();
          }
    }


    public static void main( String [ ] args ){

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

2) (Online) Tratamento de exceções com métodos: Acesse o link da atividade e faça a implementação do código-fonte conforme solicitado: https://codeboard.io/projects/351126

```
package javaapplication1;


/**
 * TODO 1: Crie uma classe interna 'MinhaExcecao' que seja subclasse da
 *         classe ArithmeticException.
 * TODO 2: No metodo leitura, dispare a excecao personalizada 'MinhaExcecao'
quando
 *         o denominador for igual a zero.
 * TODO 3: Implemente no bloco catch do metodo 'iniciar' a
 *         captura da excecao personalizada 'MinhaExcecao'
 */

import java.util.Scanner;
public class Principal {

  //TODO 1
   public class MinhaExcecao extends ArithmeticException{
      public MinhaExcecao(String message){
        super(message);
      }
   }

  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 )
              //TODO 2:
              throw new MinhaExcecao("Divisao por zero");
          else
              resultado = numerador / denominador;
  }

   public void iniciar(){

       try {
         leitura( );
       } catch (MinhaExcecao ex){
            System.out.println(
               "1: Excecao personalizada: Erro na entrada de dados"
            );
```

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
         super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

       int numerador=0;
                    System.out.println(ex.getMessage());
                }
                System.out.println("Programa continua");
            }


            public static void main( String [ ] args ){

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

```
import java.util.Scanner;
public class Principal {
  //TODO1
  public void leitura(int numerador, int denominador){

     float resultado;

     if(denominador == 0)
       throw new ArithmeticException(); //ordem
     else
       resultado = numerador/denominador;



  }
```

```
  //TODO 2

  public void iniciar(){
        int numerador=0;
        int denominador=0;
        float resultado=0;

        try {
          //TODO3
          leitura( numerador, denominador);
        } catch( ArithmeticException ex  ){
        //TODO 4
          System.out.println("Excecao: " + ex.getMessage());
        }
  }
  public static void main(String [] args){
     Principal principal = new Principal();
     principal.iniciar();
  }

}
```

3) (Online) Tratamento de exceções com múltiplas capturas: Acesse o link da atividade e faça a implementação do código-fonte conforme solicitado: https://codeboard.io/projects/351134

```
public class Principal {

   public void leitura(int numerador, int denominador) {

         float resultado=0;

            if ( denominador == 0 )
                throw new ArithmeticException();
            else
                resultado = numerador / denominador;
   }

   public void iniciar(){

        Integer [ ] vetor = new Integer[2];

        //TODO 1
        int a=0, b=0;
```

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
         super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

       int numerador=0;
          try {
             //b = vetor[20];
             leitura( a, b );
          } catch ( ArrayIndexOutOfBoundsException ex){
              //TODO 2
             System.out.println("\n1) Excecao: indice invalido");
          } catch ( ArithmeticException ex ){
             System.out.println("\n2) Excecao: divisao por zero");
          }



          //TODO 3

          //TODO 4

       }


       public static void main( String [ ] args ){

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

4) (Online) Tratamento de exceções com múltiplas capturas: Acesse o link da atividade e faça a implementação do código-fonte conforme solicitado: https://codeboard.io/projects/351157

```
o denominador: ");
            denominador = obj.nextInt();
            if(denominador == 0){
               throw new ArithmeticExcepo denominador: ");
            denominador = obj.nextInt();
            if(denominador == 0){
               throw new ArithmeticExcepimport java.util.Scanner;
import java.util.InputMismatchException;
public class Principal {

  public void leitura() {
       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();
          if ( denominador == 0 )
              throw new ArithmeticException(); //ordem
          else
              resultado = numerador / denominador;

      } catch (InputMismatchException ex){
         System.out.println("1) Excecao: entrada invalida");
      } catch( ArithmeticException ex){
         System.out.println("2) Excecao: divisao por zero");
      } catch( Exception ex ){
         System.out.println("3) Excecao: generica");
      }
 }

  public void iniciar(){

      //TODO1

      //TODO2

      //TODO3
```

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
         super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

       int numerador=0;
```

```
       //TODO4

       //TODO5
          leitura();


    }

    public static void main( String [ ] args ){

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

5) (Online) Captura de exceção personalizada: Acesse o link da atividade e faça a implementação do código-fonte conforme solicitado: https://codeboard.io/projects/351158

//Exercicio 6: REVISÃO DE CONTEÚDOS

```
package javaapplication1;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Principal {
   private static int contador=0;

   public class Bolsista {
      public Bolsista(){
         contador++;
      }
   }

   public void iniciar(){
      Bolsista joao = new Bolsista();
      Bolsista maria = new Bolsista();
      System.out.println(
            "# Bolsistas: " + contador);

      MembroUniversitario jose;
      jose = new Tecnico();
      jose.pagamento();

      jose = new Professor();
      jose.pagamento();

   ArrayList<MembroUniversitario> lista;
   lista = new ArrayList<>();

   lista.add ( jose );

   jose = new Tecnico();
   lista.add( jose );

   jose = new Professor();
   lista.add( jose );

   for( MembroUniversitario i : lista )
      System.out.println( i );

   try {
      Scanner entrada =
```

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
         super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

        int numerador=0;
              new Scanner(System.in);
        int RA = entrada.nextInt();

     } catch ( MinhaExcecao ex ){
        System.out.println(ex.getMessage());
     }

     }//fim iniciar

     public class MinhaExcecao
           extends InputMismatchException{

     }

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

}
```

Parte 2

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.

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.

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.

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.

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.

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

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

```
public class Principal {
```

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

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {

     public MinhaExcecao(){
     }
     public MinhaExcecao(String mensagem){
        super(mensagem);
     }
     public int corrigir(int entrada){
        return -1;
     }
  }

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

       int numerador=0;
```

```
    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();
    }
}
```

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

b) 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.

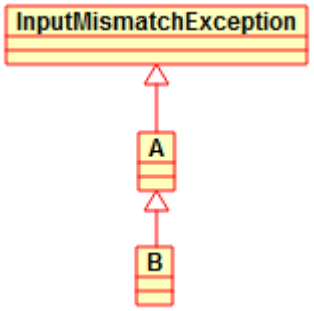
c) Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada personalizada do tipo ArithmeticException quando o quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.

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 Exceção verificada.