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

Nome: Link:

Filipe Augusto Parreira Almeida < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Matheus Mazali Maeda < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Angélica < Cópia de Aula 18 - Exercícios propostos >
Vitor Luiz de Castro Viana < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Daniel Martins de Carvalho < Cópia de POCO4A - Aula 18 - Exercícios propostos >
João Pedro Cavani Meireles< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Wesley Zimmer< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Deivid da Silva Galvao< Cópia de POCO4A - Aula 18 - Exercícios propostos >
João Vitor N. Yoshida< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Thales Kohki Hasegawa < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Raphael Uematsu Cópia de POCO4A - Aula 18 - Exercícios propostos
Rafael Kendy Naramoto Lopes < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Isabella Melo Almeida < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Alexandre Aparecido< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Marcos Tadao Shoji< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Matheus HIrata < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Thiago Cristovão de Souza < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Séfora < POO - Aula 18 >
João Pedro de Paula:< Cópia de POCO4A - Aula 18 - Exercícios propostos >
mabylly < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Plinio < Cópia de POCO4A - Aula 18 - Exercícios propostos >
Felipe Antonio Magro: Cópia de POCO4A - Aula 18 - Exercícios propostos
Gabriel Takeshi:< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Gustavo Nunes: Cópia de POCO4A - Aula 18 - Exercícios propostos
Fernando Rafael: POCO4A - Aula 18 - Exercícios propostos
Rodrigo Leandro Benedito: Cópia de POCO4A - Aula 18 - Exercícios propostos
Julio Farias< Cópia de POCO4A - Aula 18 - Exercícios propostos >
Victor Ramos Cópia de POCO4A - Aula 18 - Exercícios propostos
Lucas Viana Cópia de POCO4A - Aula 18 - Exercícios propostos
TODO: Feedback dos comentários dos alunos.

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

```
//Gustavo
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 e){
           System.out.println("Divisao por zero ");
           e.printStackTrace(); //Exibe a pilha de rastreamento da excecao
        }

   }

   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

```
//João Vitor
```

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




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

        try {
          leitura(numerador, denominador);

        } catch(ArithmeticException e ){
        //TODO 3
          System.out.print(“Excecao”);
        //TODO 4

        }
  }

 //TODO 1
          public void leitura(int numerador, int denominador)
                              throws ArithmeticException{  //pode acontecer
          //TODO 2
             if (denominador == 0) throw new ArithmeticException();   //vai acontecer

            float resultado = numerador / denominador;
            System.out.print(“Resultado: ”+resultado);
          }


  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

```
//Rafael Kendy

package exc3;

public class Principal {
   public void leitura(int numerador, int denominador) {
      float resultado=0;
         if ( denominador == 0 )
            throw new ArithmeticException();
         else
            resultado = numerador / denominador;
   }//leitura

   public void iniciar(){
      Integer [ ] vetor = new Integer[2];

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

      try{
         b = vetor[20];
         leitura( a, b );

      }catch(ArrayIndexOutOfBoundsException e){
         //TODO 2
         System.out.println("Exceção: " +e.getClass());
      } catch(ArithmeticException e){
         //TODO 4
         System.out.println("Exceção aritmética.");
      } catch(Exception e){
         System.out.println(e.getClass());
         e.printStackTrace();
      }


      //TODO 5
      try{
         b = vetor[20];
         leitura( a, b );
      }catch(Exception e){
         System.out.println(e.getClass());
         e.printStackTrace();
```

```
    }//try
  }//iniciar

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

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

```
//Thiago Cristovão de Souza
import 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;

        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 (InputMismatchException e) {
      System.out.println("Excecao 1: " + e.getClass());
    } catch (ArithmeticException e) {
      System.out.println("Excecao 2: " + e.getClass());
    } catch (Exception e) {
      System.out.println("Excecao generica");
```

```
      }


     /* try {
         leitura();
      } catch (ArithmeticException e) {
         System.out.println("Excecao 2: " + e.getClass());
      }
      try {
         leitura();
      } catch (Exception e) {
         System.out.println("Excecao generica");
      }
      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

```
//nome: Gabriel Reis Macedo

import java.util.Scanner;
public class Principal {

  //TODO 1
  public class MinhaExcecao extends ArithmeticException {
        public MinhaExcecao(){
        }
        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 )
               //ArithmeticException a;
               throw new MinhaExcecao(“Mensagem personalizada”);
           else
               resultado = numerador / denominador;
  }

   public void iniciar(){

        try {
          leitura( );
        } catch ( MinhaExcecao   a ){
             System.out.println( a.getMessage() );
        }
        System.out.println("Programa continua");
    }


    public static void main( String [ ] args ){

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

//Exercicio 6: REVISÃO DE CONTEÚDOS

```
//Excecao verificada: É aquela onde a classe é derivada da classe Exception.
```

```
//Excecao não-verificada: É aquela onde a classe é derivada da classe RuntimeException (ex.: ArithmeticException, Input…, Array… )
```

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 {

	int numerador;
	int denominador;

	public class MinhaExcecao extends ArithmeticException {

		public MinhaExcecao(String mensagem){ //Construtor sobrecarregado
			super( mensagem );
		}
		public int corrigir(){
		     return 1;
		}
	}

	//letra e)
	public class MinhaExcecaoVerificada  extends Exception {

		 public MinhaExcecaoVerificada(String mensagem){
			super( mensagem );
		 }
		 public int corrigir(){
			  return 1;
		 }
	}


	public void leitura(){

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

```
                try {
                      numerador = obj.nextInt();
                } catch( InputMismatchException  obj ){
                      numerador = 0;
                }


                System.out.print("\nDigite o denominador: ");
                try {
                denominador = obj.nextInt();
                } catch( InputMismatchException  obj ){
                      denominador = 1;
                }

                //Aritmetica de inteiros: nao eh permitida divisao por zero
                int resultado = numerador / denominador;
                //double resultado = (double) numerador / denominador;
                System.out.println("\nResultado: " + resultado);
        }




public void leitura2   throws  ArithmeticException {

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

                      numerador = obj.nextInt();

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

                denominador = obj.nextInt();
                if( denominador == 0 )
                   throw new ArithmeticException();

                //Aritmetica de inteiros: nao eh permitida divisao por zero
                int resultado = numerador / denominador;
                //double resultado = (double) numerador / denominador;
                System.out.println("\nResultado: " + resultado);
        }

public void leitura3   throws  MinhaExcecao {

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

```
                numerador = obj.nextInt();

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

            denominador = obj.nextInt();
            if( denominador == 0 )
               throw new MinhaExcecao(“Excecao personalizada”);

            //Aritmetica de inteiros: nao eh permitida divisao por zero
            int resultado = numerador / denominador;
            //double resultado = (double) numerador / denominador;
            System.out.println("\nResultado: " + resultado);
    }


public void leitura4   throws  Exception {

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

                numerador = obj.nextInt();

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

            denominador = obj.nextInt();
            if( denominador == 0 )
               throw new Exception(“Excecao personalizada”);

            //Aritmetica de inteiros: nao eh permitida divisao por zero
            int resultado = numerador / denominador;
            //double resultado = (double) numerador / denominador;
            System.out.println("\nResultado: " + resultado);
    }

public void leitura5   throws  MinhaExcecaoVerificada {

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

                numerador = obj.nextInt();

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

            denominador = obj.nextInt();
            if( denominador == 0 )
               throw new MinhaExcecaoVerificada(“Excecao personalizada”);
```

```
            //Aritmetica de inteiros: nao eh permitida divisao por zero
            int resultado = numerador / denominador;
            //double resultado = (double) numerador / denominador;
            System.out.println("\nResultado: " + resultado);
    }


     public TratamentoExcecao1(){
            leitura();

            //letra b)
            try {
                leitura2();
            } catch( ArithmeticException e ){
                  denominador = 1;
            }

            //letra c)
            try {
               leitura3();
            } catch( MinhaExcecao  e ){
                     System.out.println( e.getMessage() );
                     denominador = e.corrigir();
            }

            //letra d)
            try {
                 leitura4();

            } catch(Exception  e ){
                   System.out.println( e.getMessage() );
                   denominador = 1;
           }


            //letra e)
            try {
                  leitura5();

            } catch(MinhaExcecaoVerificada  e ){

                 System.out.println( e.getMessage() );
                 denominador = e.corrigir();
            }


     }
```

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

a) Crie um método de leitura dos 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 {

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.

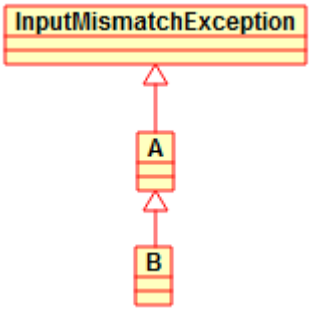
c) Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada personalizada do tipo ArithmeticException quando o quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.

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.