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

Arnald: Aula 17 - 06/10
Guilherme C. Ramalho: Cópia de POCO4A - Aula 17 - 06/10/2022 - Turma A - Exercícios…
Leonardo G. Fagote: Cópia de POCO4A - Aula 17 - 06/10/2022 - Turma A - Exercícios pr…
Matheus H. A. Pereira: Cópia de POCO4A - Aula 17 - 06/10/2022 - Turma A - Exercícios …
MARIA EDUARDA PEDROSO :
Mariana Naves: Aula 17 - POO

Exemplo de Tratamento de Exceções:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package principal;

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

/**
 *
 * @author a2150336
 */
public class Principal {

public void iniciar(){

        Scanner entrada = new Scanner(System.in);
        int numerador=0;
        try {
           System.out.println("\nNumerador: ");
           numerador = entrada.nextInt();
        } catch(InputMismatchException   e){
                System.out.println("1: Excecao na entrada de dados.");
                System.out.println(e.getMessage());
                //e.printStackTrace();//trava seu programa
        }
        System.out.println("Programa continua");
```

```
    }
   public static void main( String [ ] args ){

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

}
```

```
/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();
  }
}
```

//Exemplo de tratamento de excecoes com metodos

```
//Palavra-reservada throws


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();
  }
}
```

//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();
  }
}
```

//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 {
              b = vetor[20];
              leitura( );
       } catch ( Exception e ){
             System.out.println("4: Excecao generica");
       } catch (InputMismatchException e ){
             System.out.println("1: Excecao na entrada de dados");
       }

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

```
   }

   public static void main( String [ ] args ){

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

```
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[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();
   }
}
```

//Exemplo de tratamento de exceção personalizado:

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

  public class MinhaExcecao extends InputMismatchException {

  }

  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 (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();
    }
}
```

//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();
   }
}
```

//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;
        int denominador=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 ){
                  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();
   }
}
```

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

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

  public class ExcecaoVerificada extends Exception {


  }


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

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


    public static void main( String [ ] args ){

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

```
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 (MinhaExcecao e ){
               System.out.println( e );
               b = e.corrigir( b );
               try {
                  leitura( b );   //Programa em um estado valido.
               } catch (MinhaExcecao e2 ){
                     System.out.println( e2 );
               }
      }
      catch ( ExcecaoVerificada e ){
            System.out.println("4: Excecao generica");
      }

      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.

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.

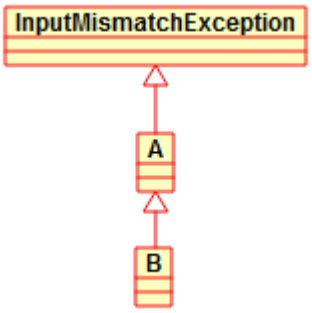
c) Crie um método de leitura de dados do usuário que possa disparar uma exceção não-verificada personalizada do tipo ArithmeticException quando o quando o CPF tiver mais do que 11 caracteres. Corrija a entrada inválida.

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.