# Lista de Exercícios 9 – Tratamento de Exceções

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

Herculano De Biasi
herculano.debiasi@unoesc.edu.br

# Lista de Exercícios 9

1. Digite e execute o programa abaixo

modulo2 - Problemas1e2Tratados.java

```java
public class Ex1a {

    public static void main(String[] args) {
        try {
            int[] numeros = {10, 0};

            System.out.println(numeros[0] / numeros[2]);
        } catch (Exception e) {
            System.out.println("Erro capturado!!!");
        }
    }

}
```

Utilize o método `getMessage()` dentro do catch para imprimir a mensagem gerada pelo Java para este erro

# Lista de Exercícios 9

2. O erro gerado é uma subclasse de `RuntimeException` – intercepte essa classe de exceções usando o bloco `catch` e imprima a mensagem 'Erro de *runtime* identificado' e também a mensagem do Java com o método `getMessage()`

# Lista de Exercícios 9

3. Conserte o erro de acesso ao *array* da linha 7, o código agora deverá gerar outra exceção – trate esta nova situação acrescentando mais um bloco `catch` para tratar a exceção `ArithmeticException`

# Lista de Exercícios 9

4. Modifique o programa ao lado de forma que ele aceite agora entrada de dados pelo teclado do usuário

   Se o usuário entrar com letras ou números em formato ponto-flutuante o código irá gerar novas exceções

   Trate essas exceções com a `InputMismatchException`

modulo2 - Ex1bSolucao.java

```java
import java.util.Scanner;

public class Ex1d {

    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);

        try {
            int[] numeros = new int[2];

            System.out.print("Digite o 1o. número inteiro: ");
            numeros[0] = ler.nextInt();

            System.out.print("Digite o 2o. número inteiro: ");
            numeros[1] = ler.nextInt();

            System.out.println(numeros[0] / numeros[1]);
        } catch (ArithmeticException e) {
            System.out.println("Erro de divisão por zero");
            System.out.println(e.getMessage());
        } catch (RuntimeException e) {
            System.out.println("Erro de runtime identificado");
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("Erro capturado!!!");
            System.out.println(e.getMessage());
        }
    }

}
```

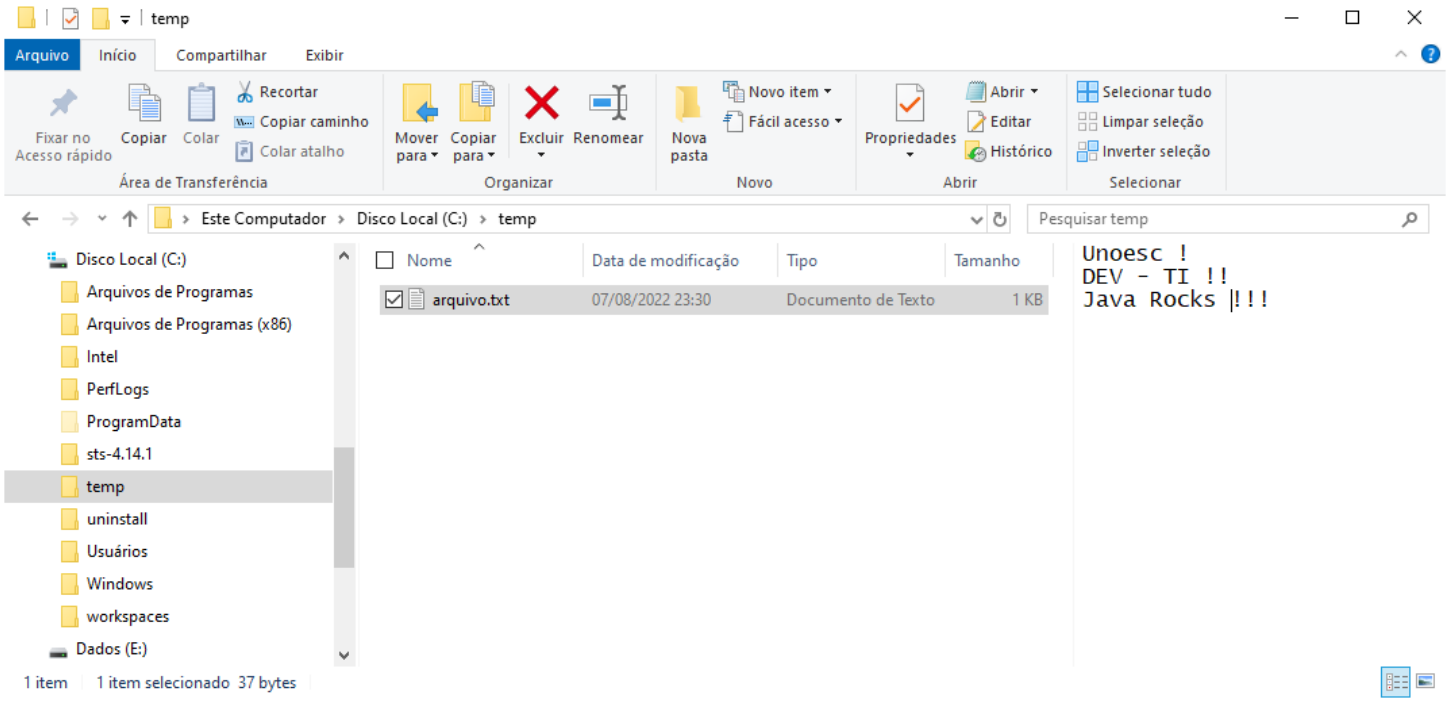
UNOESC

# Lista de Exercícios 9

5. A IDE do STS irá apontar uma advertência na linha da declaração do *scanner* – isto está sendo causado porque o recurso (objeto `ler`) não está sendo fechado e liberado da memória – resolva este problema usando o bloco `finally`

# LISTA DE EXERCÍCIOS 9

6. Para realizar o próximo exercício crie uma pasta chamada *temp* dentro do *C:* e dentro dela crie um arquivo chamado *arquivo.txt*, com o seguinte conteúdo



# Lista de Exercícios 9

6. Crie agora o programa abaixo em Java, que tem por objetivo ler o conteúdo do arquivo criado no *slide* anterior e mostrá-lo na tela

modulo2 - Ex1eSolucao.java

```java
import java.io.File;
import java.util.Scanner;

public class Ex2 {

    public static void main(String[] args) {
        // Cria um objeto representando o arquivo em disco
        File arquivo = new File("c:\\temp\\arquivo.txt");

        // Abre o arquivo usando o objeto arquivo
        Scanner sc = new Scanner(arquivo);

        // Enquanto houver linhas a serem lidas
        while (sc.hasNextLine()) {
            // Lê a próxima linha e a mostra na tela
            System.out.println(sc.nextLine());
        }
    }

}
```

# Lista de Exercícios 9

6. Este programa não funciona pois o Java obriga a realizar o tratamento de exceções – modifique o programa de forma a tratar a exceção de 'arquivo não existe' (`FileNotFoundException`), além disso crie um bloco `finally` para liberar o recurso do *scanner* caso ele não seja nulo