

Programação Orientada a Objetos - POOS3

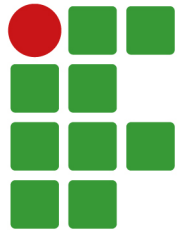
1

Tecnologia em Análise e Desenvolvimento de Sistemas

Aula 10

Arquivos e Exceções

2º semestre de 2018

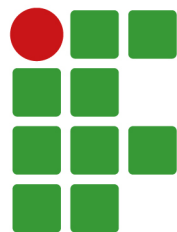


INSTITUTO FEDERAL

São Paulo

Câmpus Araraquara

Arquivos



INSTITUTO FEDERAL

São Paulo

Câmpus Araraquara

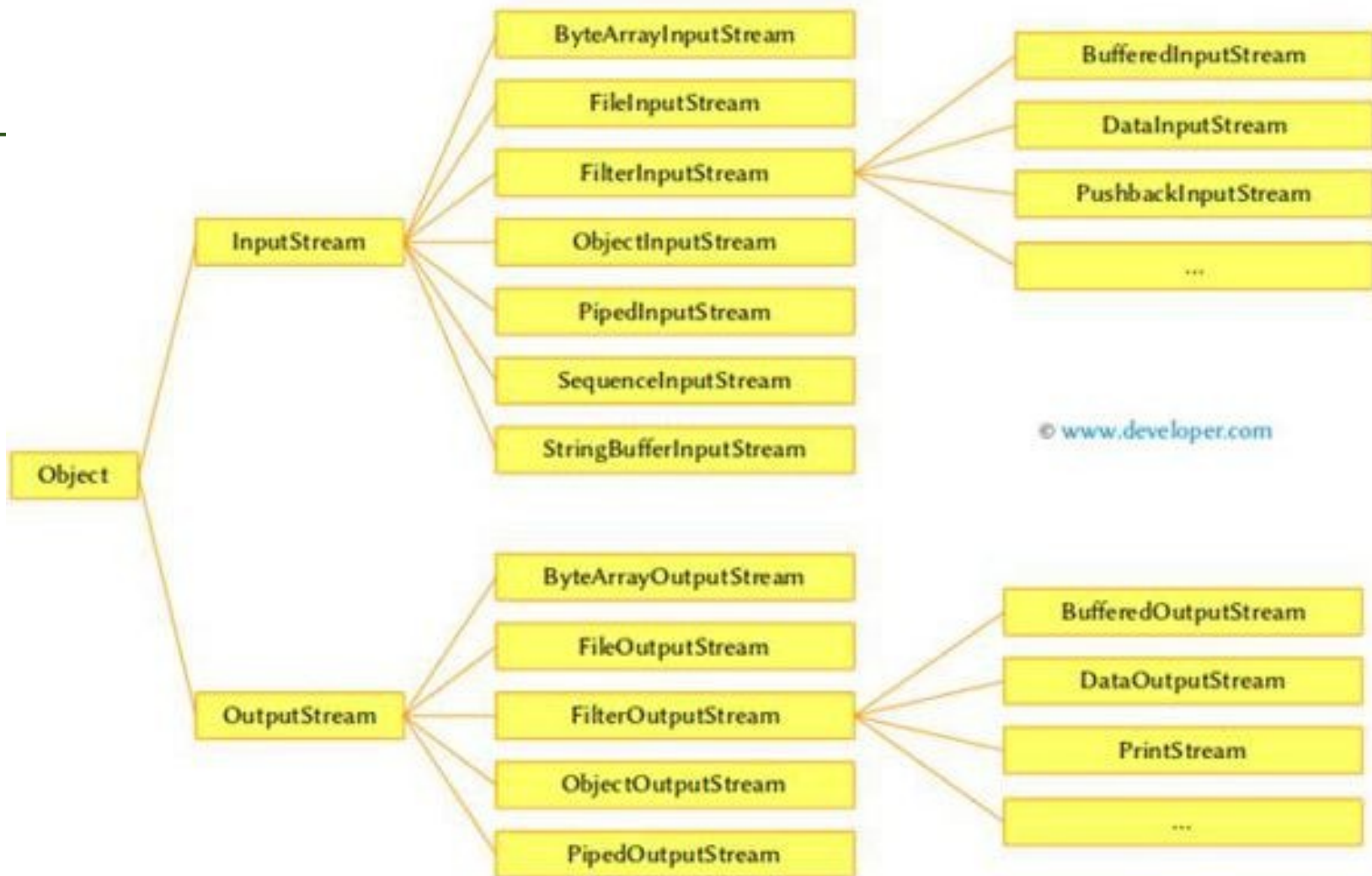
java.io

Object

- File
- InputStream
 - FileInputStream
 - FilterInputStream
 - DataInputStream
- OutputStream
 - FileOutputStream
 - FilterOutputStream
 - DataOutputStream
- RandomAccessFile
- Reader
 - BufferedReader
 - InputStreamReader
 - FileReader
- Writer
 - OutputStreamWriter
 - FileWriter
 - PrintWriter

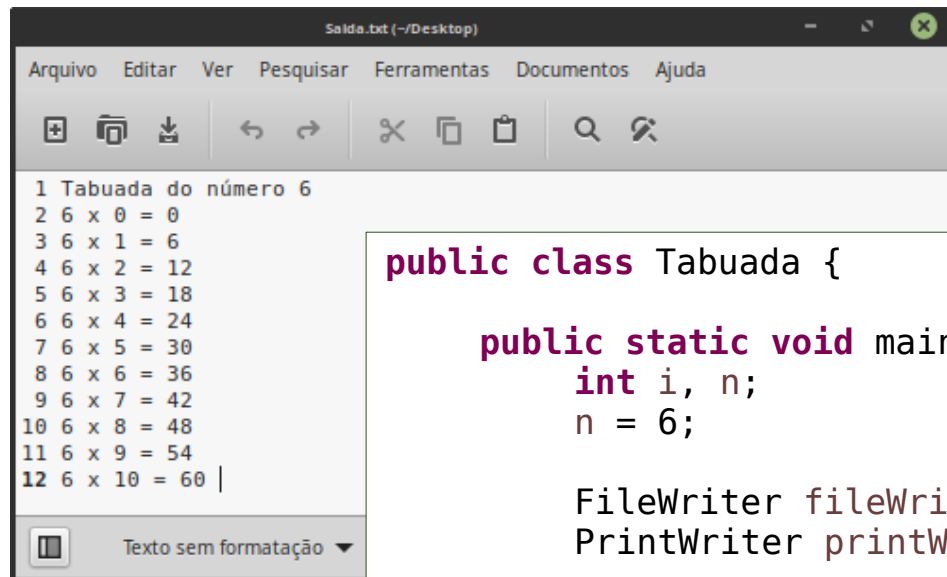
Classes para entrada ou saída baseada em bytes

Classes para entrada ou saída baseada em caracteres



Problema

- Implementar um sistema que apresenta a tabuada de um número inteiro.
 - Saída deve ser um arquivo com a tabuada.



```
1 Tabuada do número 6
2 6 x 0 = 0
3 6 x 1 = 6
4 6 x 2 = 12
5 6 x 3 = 18
6 6 x 4 = 24
7 6 x 5 = 30
8 6 x 6 = 36
9 6 x 7 = 42
10 6 x 8 = 48
11 6 x 9 = 54
12 6 x 10 = 60 |
```

```
public class Tabuada {

    public static void main(String[] args) throws IOException {
        int i, n;
        n = 6;

        FileWriter fileWriter = null;
        PrintWriter printWriter = null;

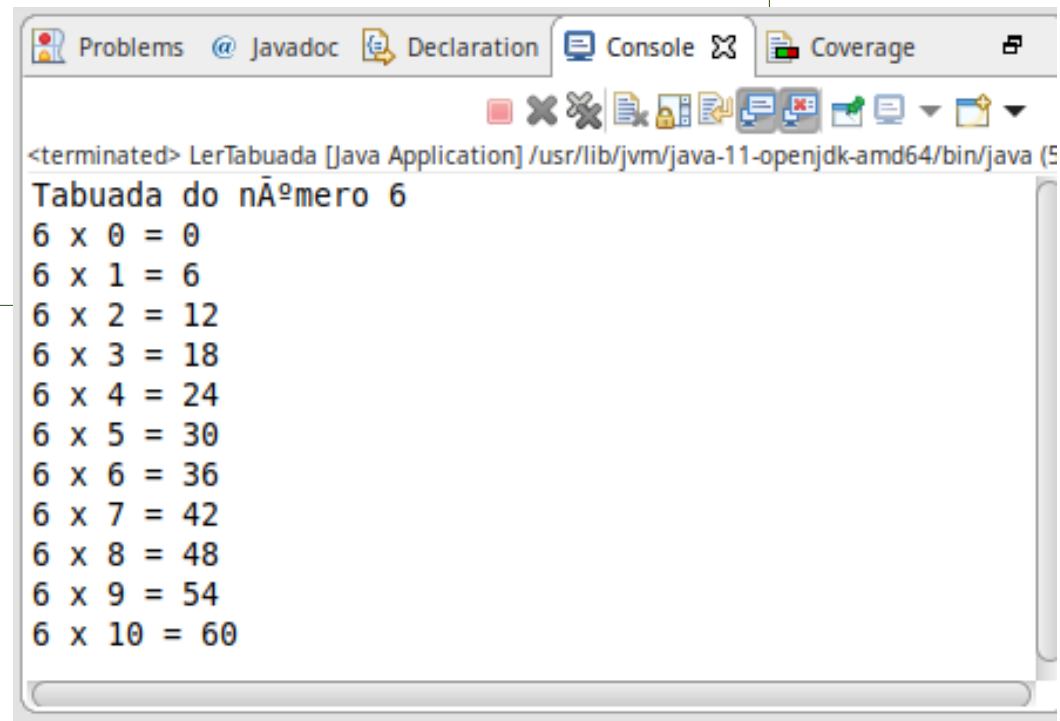
        fileWriter = new FileWriter("/home/ednilsonrossi/Desktop/Saida.txt");
        printWriter = new PrintWriter(fileWriter);

        printWriter.printf("Tabuada do número %d\n", n);
        for(i = 0; i <= 10; i++) {
            printWriter.printf("%d x %d = %d \n", n, i, n*i);
        }
        printWriter.close();
        fileWriter.close();
    }
}
```

Problema

- Implemente um sistema que apresente os dados de um arquivo (Saida.txt)

```
public class LerTabuada {  
    public static void main(String[] args) throws IOException, FileNotFoundException{  
        InputStream inputStream = null;  
        int lido;  
  
        inputStream = new FileInputStream("/home/ednilsonrossi/Desktop/Saida.txt");  
        do{  
            lido = inputStream.read();  
            if(lido != -1){  
                System.out.print((char) lido);  
            }  
        }while (lido != -1);  
        inputStream.close();  
    }  
}
```



The screenshot shows an IDE window with the 'Console' tab selected. The output text is as follows:

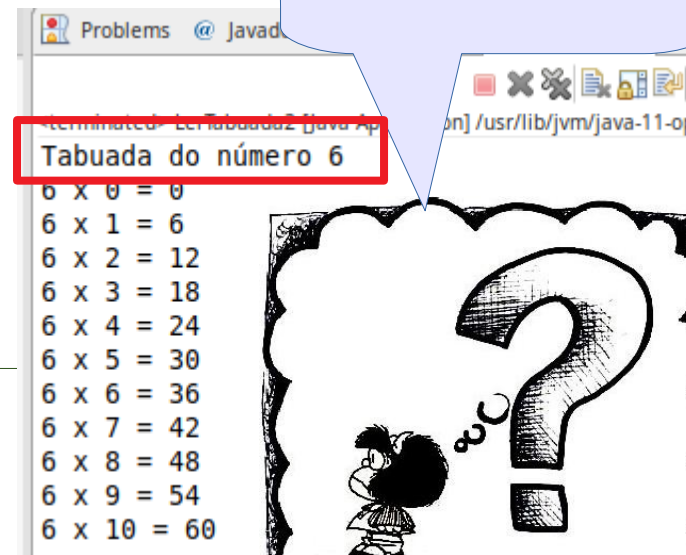
```
<terminated> LerTabuada [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (5  
Tabuada do número 6  
6 x 0 = 0  
6 x 1 = 6  
6 x 2 = 12  
6 x 3 = 18  
6 x 4 = 24  
6 x 5 = 30  
6 x 6 = 36  
6 x 7 = 42  
6 x 8 = 48  
6 x 9 = 54  
6 x 10 = 60
```


Outra solução

```
public class LerTabuada2 {  
    public static void main(String[] args) throws IOException{  
        InputStream inputStream = null;  
        InputStreamReader reader = null;  
        int lido;  
  
        inputStream = new FileInputStream("/home/ednilsonrossi/Desktop/Saida.txt");  
        reader = new InputStreamReader(inputStream);  
  
        while ((lido = reader.read()) != -1){  
            System.out.print((char) lido);  
        }  
  
        reader.close();  
        inputStream.close();  
    }  
}
```

InputStreamReader

Realiza a ponte de comunicação de fluxos de bytes para fluxos de caracteres: ele lê bytes e os decodifica em caracteres.



Agora lendo a linha toda

```
public class LerTabuada3 {  
    public static void main(String[] args) throws IOException{  
        InputStream inputStream = null;  
        InputStreamReader inputStreamReader = null;  
        BufferedReader bufferedReader = null;  
        String linha;  
  
        inputStream = new FileInputStream("/home/ednilsonrossi/Desktop/Saida.txt");  
        inputStreamReader = new InputStreamReader(inputStream);  
        bufferedReader = new BufferedReader(inputStreamReader);  
  
        while((linha = bufferedReader.readLine()) != null){  
            System.out.println(linha);  
        }  
  
        bufferedReader.close();  
    }  
}
```

BufferedReader

Lê o texto de um fluxo de entrada de caracteres, armazenando-os em um buffer de caracteres para fornecer a leitura eficiente de caracteres, matrizes e linhas.

Fazendo Eco

```
public class Eco {  
    public static void main(String[] args) throws IOException{  
        InputStream inputStream;  
        InputStreamReader inputStreamReader;  
        BufferedReader bufferedReader;  
        String linha;  
  
        inputStream = System.in;  
        inputStreamReader = new InputStreamReader(inputStream);  
        bufferedReader = new BufferedReader(inputStreamReader);  
  
        do{  
            linha = bufferedReader.readLine();  
            if(linha != null) {  
                linha = "Disse: " + linha;  
                System.out.println(linha);  
            }  
        }while (linha != null);  
    }  
}
```



Escrevendo no arquivo

```
public class Escreve {  
    public static void main(String[] args) throws IOException{  
        OutputStream outputStream;  
        OutputStreamWriter outputStreamWriter;  
        String frase;  
  
        frase = "Prova de POOS3 em 30/11/2018";  
        outputStream = new FileOutputStream("/home/ednilsonrossi/Desktop/mensagem.txt");  
        outputStreamWriter = new OutputStreamWriter(outputStream);  
        for(char c : frase.toCharArray()){  
            outputStream.write(c);  
        }  
        outputStreamWriter.close();  
        outputStream.close();  
    }  
}
```

O que acontece com o arquivo se executar o programa novamente?

OutputStreamWriter
Realiza a ponte de comunicação de fluxos de caracteres para fluxos de bytes: ele lê caracteres e os decodifica para bytes.



Melhorando

```
public class Escreve2 {  
    public static void main(String[] args) throws IOException{  
        OutputStream outputStream;  
        OutputStreamWriter outputStreamWriter;  
        BufferedWriter bufferedWriter;  
        String frase;  
  
        frase = "Prova de POOS3 em 30/11/2018";  
        outputStream = new FileOutputStream("/home/ednilsonrossi/Desktop/mensagem.txt");  
        outputStreamWriter = new OutputStreamWriter(outputStream);  
        bufferedWriter = new BufferedWriter(outputStreamWriter);  
  
        bufferedWriter.write(frase);  
        bufferedWriter.newLine();  
        bufferedWriter.write(frase);  
        bufferedWriter.newLine();  
        bufferedWriter.write(frase);  
        bufferedWriter.newLine();  
        bufferedWriter.write("Vou estudar muito até lá!!!");  
  
        bufferedWriter.close();  
    }  
}
```

BufferedWriter

Escreve o texto para um fluxo de saída, armazenando-os em um buffer de caracteres para fornecer a escrita eficiente de caracteres, matrizes e linhas.

Problema aplicado



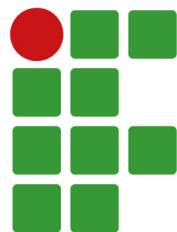
- Implementar um sistema de cadastro.
- Sistema deve permitir salvar pessoas em um arquivo de dados e recuperar as pessoas cadastradas que estão no arquivo.
- O arquivo deve ser incrementado, ou seja, os dados cadastrados anteriormente não podem ser perdidos em uma nova execução do programa.

Muito fácil mestre,
manda outro que
esse não dá para o
começo!



Mostra o Exemplo19
para ver que está
fácil!

Exceções



INSTITUTO FEDERAL

São Paulo

Câmpus Araraquara

Teste de software

- Teste de software é a processo de verificar se o produto esta de acordo com as especificações determinadas e que funciona corretamente no qual foi projetado.
- O objetivo dos testes é de achar **erros** ou **falhas** para que a equipe de desenvolvimento possa corrigi-lo o mais rápido possível, pois quanto mais rápido descobrir um erro, mais barato ele fica.
- Por essas características a disciplina de teste é considerável "**destrutiva**" e não "**construtiva**".

Defeito x Erro x Falha

- Defeito
 - É qualquer imperfeição ou inconsistência no produto do software ou em seu processo, um defeito é também uma não conformidade. O Defeito faz parte do produto, é algo que está implementada no código de maneira errada.
- Erro
 - O Erro pode ser um resultado de um defeito ou uma falha, como um retorno esperado, que por causa de uma falha teve um valor diferente do que esperado.
- Falha
 - Esta está ligada ao hardware, como uma rede inacessível, queda de energia. Uma falha pode ocorrer por causa de um erro, por exemplo, houve um retorno de um valor não esperado, como null, isso é um erro, e por causa desse null ocasionou uma falha no sistema.

Causa de defeitos

- Usuários especificam os requisitos errados;
- Analistas interpretam erradamente os requisitos
- Especificações Funcionais e Técnicas elaboradas erroneamente;
- Codificação errada;
- Dados errados;
- Correções erradas de defeitos;
- Inconsistência nos dados.

Exceções: Conceito

- Uma exceção é um evento que ocorre durante a execução de uma aplicação e que interrompe o fluxo normal das instruções.
- Muitos tipos de erros podem causar exceções como erros de hardware ou software.

```
public class LeNumero {

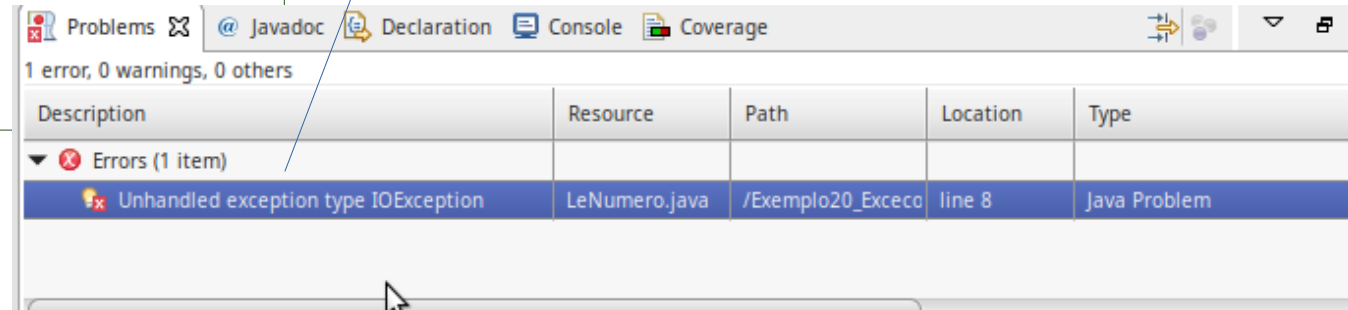
    public static void main(String[] args) {
        byte[] dados = new byte[10];
        System.out.println("Digite um número: ");
        System.in.read(dados);
    }

}
```

O erro de compilação apresentado demonstra que a exceção de entrada e saída (IOException) deve ser tratada obrigatoriamente.

Muito fácil resolver!!!

Basta clicar em alguma solução que o Eclipse oferece que está tudo certo!



```
public static void main(String[] args) {
    byte[] dados = new byte[10];
    System.out.println("Digite um número: ");
    System.in.read(dados);
}
```

- 🔧 Add throws declaration
- 🔧 Surround with try/catch
- 🕒 Assign statement to new local variable (Ctrl+2 L)
- 📦 Assign statement to new field (Ctrl+2 F)

```
...
package exemplo1;

import java.io.IOException;

public class LeNumero {
    ...
    System.out.println("Digite um número: ");
    try {
        System.in.read(dados);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Press 'Tab' from proposal table or click for focus

```
public class LeNumero {  
  
    public static void main(String[] args) {  
        byte[] dados = new byte[10];  
        System.out.println("Digite um número: ");  
        try {  
            System.in.read(dados);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Seus pedidos foram
atendidos glorioso
aluno!



try-catch-finally

- Para tratar exceções, a linguagem Java disponibiliza a estrutura de controle try-catch-finally, que é utilizada para o tratamento de exceções e fluxo de execução da aplicação

```
try{  
    Bloco de comandos  
}catch (Exception ex){  
    Bloco de comandos  
}finally{  
    Bloco de comandos  
}
```

Bloco de comandos que desejamos que o programa execute, mas que pode acarretar em um erro.

Bloco de comandos que é executado caso ocorra algum erro no bloco acima.

O bloco finally é executado sempre, ou seja, o finally é executado com o fluxo normal ou se ocorrer algum erro.

```

public class LeNumero2 {
    public static void main(String[] args) {
        byte[] dados = new byte[10];
        int nro=0;
        System.out.println("Digite um número: ");
        try {
            System.in.read(dados);
            nro = Integer.parseInt(new String(dados).trim());
        } catch (IOException e) {
            System.out.println("Ocorreu um erro e o número não pode ser lido.");
            System.out.println("Mensagem: " + e.getMessage());
            nro = -1;
        } finally{
            System.out.println("Nro lido: " + nro);
        }
    }
}

```

Agora sim, tudo resolvido!!!



Problems @ Javadoc Declaration Console

<terminated> LeNumero2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (7 de nov de 2018 19:23:50)

Digite um número:
12345
Nro lido: 12345

Problems @ Javadoc Declaration Console Coverage

<terminated> LeNumero2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (7 de nov de 2018 19:23:50)

Digite um número:
dez
Nro lido: 0Exception in thread "main"
java.lang.NumberFormatException: For input string: "dez"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
at java.base/java.lang.Integer.parseInt(Integer.java:652)
at java.base/java.lang.Integer.parseInt(Integer.java:770)
at exemplo1.LeNumero2.main(LeNumero2.java:12)

O que é isso !?



Observe que o finally foi executado de qualquer forma.



```
<terminated> LeNumero2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (7 de nov de 2018 19:23:50)
Digite um número:
dez
Nro lido: 0Exception in thread "main"
java.lang.NumberFormatException: For input string: "dez"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
    at java.base/java.lang.Integer.parseInt(Integer.java:770)
    at exemplo1.LeNumero2.main(LeNumero2.java:12)
```

Isso porque tratamos uma exceção e o sistema encontrou outra exceção. Tratamos IOException mas obtivemos NumberFormatException.


```

public class LeNumero3 {
    public static void main(String[] args) {
        byte[] dados = new byte[10];
        int nro=0;
        System.out.println("Digite um número: ");
        try {
            System.in.read(dados);
            nro = Integer.parseInt(new String(dados).trim());
        } catch (IOException e) {
            System.out.println("Ocorreu um erro e o número não pode ser lido.");
            System.out.println("Mensagem: " + e.getMessage());
            nro = -1;
        } catch (NumberFormatException nfe){
            System.out.println("Erro ao converter o número");
            nro = -1;
        }
        finally{
            System.out.println("Nro lido: " + nro);
        }
    }
}

```

Agora ficou certo?



Se o erro ocorrido for outro o sistema continuará apresentando defeito!!!

Vamos resolver com o erro genérico!



Problems @ Javadoc Declaration Console

<terminated> LeNumero3 [Java Application] /usr/lib/jvm/java-11-

Digite um número:
dez
Erro ao converter o número
Nro lido: -1

<terminated> LeNumero3 [Java Application] /usr/lib/jvm/java-11-

Digite um número:
10
Nro lido: 10

```
public class LeNumero4 {  
    public static void main(String[] args) {  
        byte[] dados = new byte[10];  
        int nro=0;  
        System.out.println("Digite um número: ");  
        try {  
            System.in.read(dados);  
            nro = Integer.parseInt(new String(dados).trim());  
        } catch (IOException e) {  
            System.out.println("Ocorreu um erro e o número não pode ser lido.");  
            System.out.println("Mensagem: " + e.getMessage());  
            nro = -1;  
        } catch (NumberFormatException nfe){  
            System.out.println("Erro ao converter o número");  
            nro = -1;  
        } catch (Exception e) {  
            System.out.println("Não sei o que deu errado. Veja abaixo o erro:\n");  
            e.printStackTrace();  
            nro = -1;  
        }  
        finally{  
            System.out.println("Nro lido: " + nro);  
        }  
    }  
}
```

Outro exemplo

```
public class LeArray {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        int i;  
        int[] nros = new int[10];  
  
        i=0;  
        do{  
            nros[i] = input.nextInt();  
            i++;  
        }while(true);  
    }  
}
```

O que mais pode dar de errado?

O que vai dar de errado aqui?

Onde buscar informação?



```
public class LeArray2 {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        int i;  
        int[] nros = new int[10];  
  
        try {  
            i = 0;  
            do {  
                nros[i] = input.nextInt();  
                i++;  
            } while (true);  
        } catch (ArrayIndexOutOfBoundsException aiobEx){  
            System.out.println("Vetor está cheio, impossível inserir mais dados");  
        } catch (InputMismatchException imEx){  
            System.out.println("Erro na entrada de dados");  
        } catch (Exception ex){  
            System.out.println("Outro erro desconhecido!");  
        }  
  
        for(i=0; i<nros.length; i++){  
            System.out.println(nros[i]);  
        }  
    }  
}
```

Scanner (Java Platform SE 8) - Mozilla Firefox

Scanner (Java Platform SE 8) x +

https://docs.oracle.com/javase/8/docs/api/ Pesquisar

Java™ Platform Standard Ed. 8

All Classes All Profiles

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- java.awt.dnd

SafeVarargs

SampleModel

Sasl

SaslClient

SaslClientFactory

SaslException

SaslServer

SaslServerFactory

Savepoint

SAXException

SAXNotRecognizedException

SAXNotSupportedException

SAXParseException

SAXParser

SAXParserFactory

SAXResult

SAXSource

SAXTransformerFactory

Scanner

ScatteringByteChannel

ScheduledExecutorService

ScheduledFuture

ScheduledThreadPoolExecutor

Schema

SchemaFactory

SchemaFactoryConfigurationError

SchemaFactoryLoader

SchemaOutputResolver

SchemaViolationException

Throws:

IllegalStateException - if this scanner is closed

nextInt

public int nextInt()

Scans the next token of the input as an int.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the int scanned from the input

Throws:

- InputMismatchException - if the next token does not match the `Integer` regular expression, or is out of range
- NoSuchElementException - if input is exhausted
- IllegalStateException - if this scanner is closed

nextInt

public int nextInt(int radix)

Scans the next token of the input as an int. This method will throw `InputMismatchException` if the next token cannot be translated into a valid int value as described below. If the translation is successful, the scanner advances past the input that matched.

If the next token matches the `Integer` regular expression defined above then the token is converted into an int value as if by removing all locale specific prefixes, group separators, and locale specific suffixes, then mapping non-ASCII digits into ASCII digits via `Character.digit`, prepending a negative sign (-) if the locale specific negative prefixes and suffixes were present, and passing the resulting string to `Integer.parseInt` with the specified radix.

Parameters:

Tratamento de exceções é apenas colocar um código dentro de um try-catch?



dreamstime.com

```

public class Calculo {

    public int divide(int x, int y) throws Exception{
        if(x < y){
            throw new Exception("X é menor que Y");
        }
        return x / y;
    }

    public static void main(String[] args) {
        Calculo c = new Calculo();
        int r;
        try {
            r = c.divide(1, 2);
            System.out.println("Divisão: " + r);
        } catch (Exception ex) {
            System.out.println("Exceção gerada: " + ex);
        }
    }
}

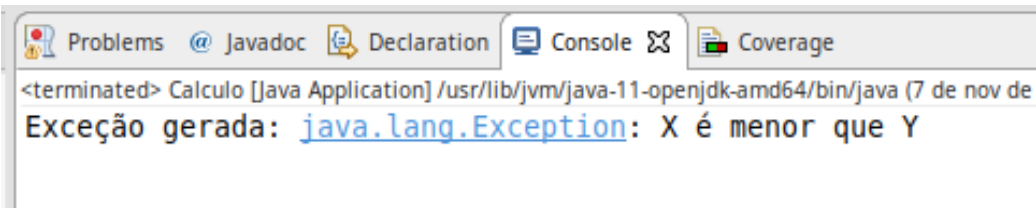
```

Quando programa-se um método é possível forçar com que o usuário “trate” (use try-catch) a exceção gerada.

Isso é feito declarando a clausula throws para o método e instanciando (Sim uma Exception é um objeto) uma Exception.

Como foi previsto pelo método a possibilidade de erro, o programador deverá tratar a exceção.

Qual a razão de fazer com que o usuário trate a exceção se meu método pode ter um if que não deixa o erro ocorrer?



```

public class Calculo {

    public int divide(int x, int y) throws Exception{
        if(x < y){
            throw new Exception("X é menor que Y");
        }
        return x / y;
    }

    public static void main(String[] args) {
        Calculo c = new Calculo();
        int r;
        try {
            r = c.divide(1, 2);
            System.out.println("Divisão: " + r);
        } catch (Exception ex) {
            System.out.println("Exceção gerada: " + ex);
        }
    }
}

```

O método divide() tem o objetivo de dividir dois números inteiros.

Se o argumento X for menor que o argumento Y, o resultado será algo entre 0 e 1, ou seja, um número de ponto flutuante.

Tudo bem, se $x < y$ basta retornar zero.

```

int r = 0;
if(x >= y){
    r = x/y;
}
return r;

```

Mas esse é um resultado correto?

A exceção permite a diminuição de POG em nosso sistema.

Se o método divide() retornar zero para casos onde $x < y$ é uma POG.


```
public class Calculo2 {  
    public int divide(int x, int y) throws ExceptionDivisaoImpossivel{  
        if(x < y){  
            throw new ExceptionDivisaoImpossivel("X é menor que Y");  
        }  
        return x / y;  
    }  
  
    public static void main(String[] args) {  
        Calculo2 c = new Calculo2();  
        int r;  
        try {  
            r = c.divide(1, 2);  
            System.out.println("Divisão: " + r);  
        } catch (ExceptionDivisaoImpossivel ex) {  
            System.out.println("Exceção gerada: " + ex);  
        }  
    }  
}
```

```
class ExceptionDivisaoImpossivel extends Exception{  
    public ExceptionDivisaoImpossivel(String msg){  
        super(msg);  
    }  
}
```

Para que isso
é útil?



É possível a criação de exceções
personalizadas.

Material adicional

- **Leitura Obrigatória**

- Winder, R.; Roberts, G. Desenvolvendo Software em Java. 3 ed. Rio de Janeiro: LTC, 2009.
 - Capítulo 8

- **Na internet**

- <https://www.devmedia.com.br/testes-de-software-entendendo-defeitos-erros-e-falhas/22280>
- <https://pt.stackoverflow.com/questions/71670/como-criar-uma-exception-exce%C3%A7%C3%A3o-customizada-em-java#71671>
- <https://www.caelum.com.br/apostila-java-orientacao-objetos/excecoes-e-control-e-de-erros/#motivao>



Trabalhando

- **Exercícios avaliativos**
 - 13 e 14
- **APS**

