

Depois disto ouvi a voz do Senhor, que dizia:
A quem enviarei, e quem há de ir por nós?

Então disse eu: Eis-me aqui, envia-me a mim.

Isaías 6:8

Especialização em Java

Java I

- Prof. José Antonio Gonçalves

Ementa:

- **Orientação a Objetos em Java:** Classes, Objetos, Herança, Polimorfismo, Classes Abstratas, Interface;

Conteúdo já visto

- **Exceções;**
- **Manipulação de Texto e Strings;**
- **Componentes básicos de interface gráfica;**
- **Tratamento de Eventos.**

Nestes Slides:

- **Orientação a Objetos em Java:**

Delegando o tratamento de exceções a outra classe

Delegando o tratamento de exceções

Contextualização (1/ 2)

Existem situações em que não temos como anteceder as possibilidades que podem interferir no funcionamento do nosso programa. São situações que nos obrigam a deixar para o usuário programador, o qual utilizará a nossa classe (ou byteCode), fazer o tratamento das exceções.

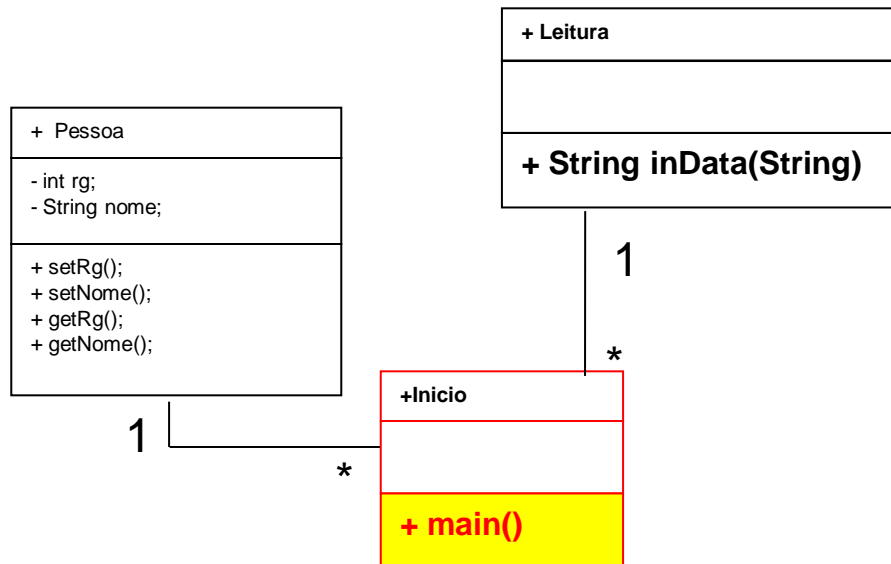
Contextualização (2 / 2)

Neste caso, para delegar o tratamento, devemos utilizar a cláusula **throws** ao construir um método que poderá disparar uma exceção. Para isso colocamos a palavra `throws` após a assinatura do método e, após ela as exceções que este método pode disparar. Este procedimento:

- permite que não tratemos a exceção de imediato;
- obriga que quem utilizar nossa classe faça o tratamento da exceção (ou delegue também)

Exemplificação

Para exemplificar, imagine o exemplo a seguir



Exceções (estudo de caso)

Relembrando a **classe Leitura** (*recebe uma String e retorna uma String*)

```
1. import java.io.BufferedReader;
2. import java.io.IOException;
3. import java.io.InputStreamReader;

4. public class Leitura{
5.     public static String inData(String label){
6.         InputStreamReader c = new InputStreamReader(System.in);
7.         BufferedReader cd = new BufferedReader(c);
8.         System.out.print(label);
9.         String s = "";
10.        try{
11.            s = cd.readLine();
12.        }
13.        catch(IOException e){
14.            System.out.println("Erro de entrada");
15.        }
16.        return s;
17.    }
18. }
```

+ Leitura
+ String inData(String)

Exemplificação

Pode-se perceber um catch (neste caso). É claro que temos pelo menos uma operação (**readLine()**) que poderá apresentar erros e disparar exceções. Para verificar isto basta observar as operações que estão “dentro” do try e como um possível erro é “capturado” pelo seu catch:

Aqui vemos que o método `readLine()` **pode disparar** a exceção **“IOException”**

Exemplificação

Mas e se eu não quisesse tratar esta exceção aqui e sim deixar para outro usuário (da minha classe) tratá-lo?

Neste caso deve utilizar a cláusula throws da seguinte forma:

Exemplificação

Relembrando a **classe Leitura** (*recebe uma String e retorna uma String*)

```
1. import java.io.BufferedReader;
2. import java.io.IOException;
3. import java.io.InputStreamReader;
```

```
4. public class Leitura{
5.     public static String inData(String label) throws IOException {
```

```
6.         InputStreamReader c = new InputStreamReader(System.in);
7.         BufferedReader cd = new BufferedReader(c);
8.         System.out.print(label);
9.         String s = "";
```

```
10.         s = cd.readLine();
```

```
11.         return s;
```

```
12.     }
```

```
13. }
```

+ Leitura
+ String inData(String)

Perceba que, diferente do código anterior, o método **readLine()** não está dentro de um try e também não há **catch**

Exemplificação

De acordo com o código anterior, a possível exceção deverá ser tratada na classe **Inicio**

Exemplificação

```
1. import java.io.IOException;
2.
3. public class Inicio {
4.     public static void main(String args[]){
5.         Leitura l = new Leitura();
6.         Pessoa pes = new Pessoa();
7.         try{
8.
9.             pes.setId(Integer.parseInt(l.inData("\nEntre com o ID <deve ser numero>: ")));
10.            pes.setNome(l.inData("\n Entre com o nome: "));
11.
12.        }
13.        catch(IOException e){
14.            System.out.println("\n Erro de entrada");
15.        }
16.        System.out.println("\n ID....: "+pes.getId());
17.        System.out.println("\n Nome..: "+pes.getNome());
18.    }
19. }
```

Veja que todas operações que utilizam o método `inData` (que pode disparar a exceção `IOException`), pertencente a classe `Leitura`, estão dentro do `try...`

... e a exceção `IOException` que ele pode disparar está sendo tratada na própria classe `Inicio`, e não na classe `Entrada`. Logo percebe-se que a classe `Leitura` delegou o tratamento da exceção `IOException` para quem utilizasse seu método `inData()`.