

UNIP
UNIVERSIDADE PAULISTA

Linguagem de Programação Orientada a Objeto

LPOO 09 – Tratamento de Exceção

www.wg.pro.br

Prof. Msc Wanderley Gonçalves Freitas

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 1 de 30

UNIP
UNIVERSIDADE PAULISTA

Agenda

- bugs: Três tipos de erros de programação
- Três Categoria de Exceções
- Exceção
- Hierarquia de Classes de Exceções
- Terminologia
 - bloco try
 - bloco catch
 - bloco finally
- Tipos de Exceções
 - verificada
 - não verificada
- Exercícios de fixação



Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 2 de 30

UNIP
UNIVERSIDADE PAULISTA

Objetivo

- Entender o funcionamento de exceções
- Aprender como levantar e tratar exceções

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 3 de 30

UNIP
UNIVERSIDADE PAULISTA

Três tipos de erros de programação

- **Erros de compilação (erro tempo compilação)**
 - Erros de compilação, também conhecido como *erros de compilador*, são **erros que impedem seu programa de executar**
 - Exemplo : **Esquecer colocar ponto e virgula no final dos comandos em java**
- **Erros de Tempo de Execução (erro runtime)**
 - Erros em tempo de execução são **erros que ocorrem enquanto o programa é executado**. Elas normalmente ocorrem quando o programa tenta uma operação que é impossível executar.
 - Exemplo : **Tentativa de abertura de um arquivo inexistente**
- **Erros de lógica**
 - **Erros lógicos são erros que impedem seu programa de fazer o que você pretendia fazer**. Seu código pode ser compilado e executado sem erros, mas o resultado de uma operação pode produzir um resultado que você não esperava
 - Exemplo : **operação aritmética é inválida – divisão por zero**

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 4 de 30

UNIP
UNIVERSIDADE PAULISTA

Erros Compilação

```

1
2
3 public class Conta {
4     protected double saldo
5     protected doubl limite;
6     protected taxaJuros;
7
8
9     public abstract void alterarLimite(double pValor);
10
  
```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 5 de 30

UNIP
UNIVERSIDADE PAULISTA

Erros Lógica----adicionar classe - wanderley

```

4     protected double gratificacao;
5
6     @Override
7     public double obterPrimeiraParcela() {
8         return 0;
9     }
10    @Override
11    public double obterSegundaParcela() {
12        return 0;
13    }
14    @Override
  
```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 6 de 30

Erros runtime

```

5 public class Teste {
6     public static void main(String[] args) {
7         Professor funcionario = null;
8         funcionario.setSalario(10000);
9         System.out.println("salario final "
10            + funcionario.calcularSalario());
    }
}

```

```

<terminated> Teste (6) [Java Application] C:\java\jdk1.7.0_65\bin\javaw.exe (13/05/2015 17:14)
Exception in thread "main" java.lang.NullPointerException
    at br.unip.lpoo.parte2.aula.execcao.Teste.main(Teste.java:8)

```

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 7 de 30

Exceções

- As exceções são eventos inesperados que ocorrem durante a execução de um programa. Uma exceção pode ser o resultado de uma condição de erro ou simplesmente uma entrada inesperada
- O objetivo do Tratamento de Exceção é manter o programa funcionando
 - Ao invés de encerrar abruptamente
 - Aumenta a robustez
- As exceções são um *grupo de classes* com a finalidade de tratar erros e condições especiais na execução de um programa

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 8 de 30

Três Categoria de Exceções

As exceções podem ser divididas em três categorias:

- (1) Exceções geradas por erros de programas (lógica):**
 - Em geral são *bugs (ERROS)* de sistema.
 - Exemplo :**
 - acessar um objeto nulo, o que ocasionará uma *NullPointerException*
 - tentar acessar uma posição inválida de um *array*, que devolvem outra exceção famosa: *ArrayIndexOutOfBoundsException*.
 - Em geral, não há nada o que fazer quando esses erros acontecerem, senão, deixar que o erro simplesmente ocorra.

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 9 de 30

Três Categoria de Exceções

(1) Exceções geradas por erros de programas (lógica):

```

4
5     public static void main(String[] args) {
6         int numeros[] = { 40, 50, 60 };
7         System.out.println(numeros[3]);
8     }
9 }

```

```

<terminated> Teste1 [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (17 de mai de 2016 09:53:37)
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
    at br.unip.lpoo.aula.parte2.execcao.Teste1.main(Teste1.java:7)

```

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 10 de 30

Três Categoria de Exceções

- (2) Exceções geradas por violações de contratos entre APIs:** O programa tenta fazer algo não permitido pela API, a qual gera um erro específico para a ocasião:

Exemplo

 - Tentar processar um arquivo XML corrompido. Nestes casos, é possível contornar o problema informando ao usuário sobre o problema.
 - Outro é tentar passar uma URL inválida para o construtor da classe URL. O endereço `http://www.pucpr.br` vai gerar uma *MalformedURLException*. Claro, não existe o protocolo `httpk`.)

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 11 de 30

Três Categoria de Exceções

(2) Exceções geradas por violações de contratos entre APIs

```

try {
    conexao = FactoryConnection.getInstance().getConnectionSimples();
    PreparedStatement instrucaoSQL = conexao.prepareStatement("select * from aluno999");
    ResultSet resultado = instrucaoSQL.executeQuery();
    while (resultado.next()) {
        System.out.printf("matricula = %s nome= %s\n", resultado.getString("matricula")
    )
}

} catch (SQLException e) {
    System.out.printf("Erro comando SQL \n " + e.getMessage());
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

```

```

<terminated> TesteConexao (1) [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (17 de mai de 2016 09:58:14)
Erro comando SQL
Table 'faculdade.aluno999' doesn't existcom.mysql.jdbc.exceptions.jdbc4.MySQLException
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
at java.lang.reflect.Constructor.newInstance(Constructor.java:248)
at com.mysql.jdbc.Util.handleNewInstance(Util.java:409)
at com.mysql.jdbc.Util.getInstance(Util.java:384)
at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:1052)
at com.mysql.jdbc.MySQLIO.checkErrorPacket(MySQLIO.java:432)
at com.mysql.jdbc.MySQLIO.checkErrorPacket(MySQLIO.java:4164)
at com.mysql.jdbc.MySQLIO.sendCommand(MySQLIO.java:2615)
at com.mysql.jdbc.MySQLIO.executeQueryDirect(MySQLIO.java:2776)
at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2838)

```

Prof. Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 11 de 30

Três Categoria de Exceções

- **(3) Exceções geradas por falhas em recursos externos:** São geradas quando algum recurso externo falhar

Exemplo

- Tentar conectar um servidor de FTP e o mesmo estar fora do ar.
- Tentar conectar um servidor de banco de bancos e o mesmo estar fora do ar.

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 13 de 30

Três Categoria de Exceções

(3) Exceções geradas por falhas em recursos externos

```

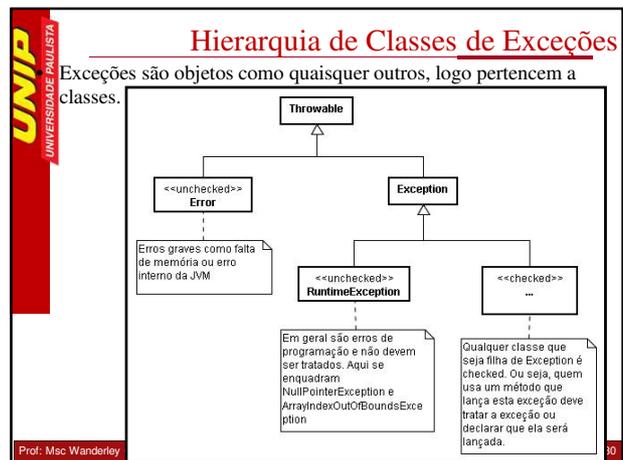
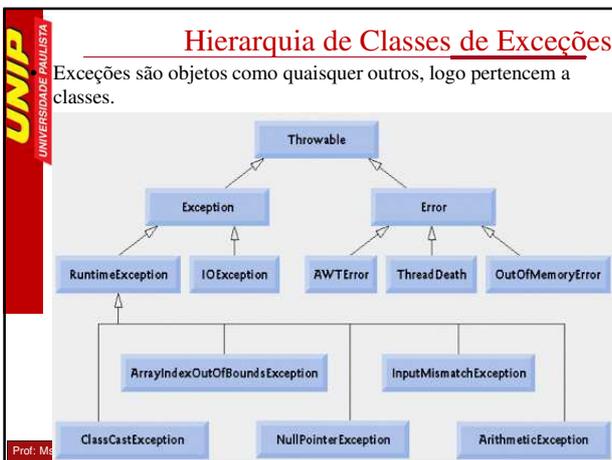
public Connection getConnectionSimples() {
    Connection con = null;
    String driver = "com.mysql.jdbc.Driver";
    String usuario = "root";
    String senha = "wander2017";
    String url = "jdbc:mysql://127.0.0.1/faculdade";
    try {
        Class.forName(driver);
        con = DriverManager.getConnection(url, usuario, senha);
    } catch (SQLException e) {
        System.out.println("URL: " + url + " [USUÁRIO: " + usuario + "].");
        e.printStackTrace();
    }
}
    
```

```

<terminated> TesteConexao (1) [Java Application] C:\Program Files\Java\jre1.8.0_75\bin\javaw.exe [17 de mai de 2018 10:05]
URL: jdbc:mysql://127.0.0.1/faculdade [USUÁRIO: root].
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
at java.lang.reflect.Constructor.newInstance(Unknown Source)
at com.mysql.jdbc.Util.handleNewInstance(Util.java:409)
at com.mysql.jdbc.SQLException.createCommunicationsException(SQLException.java:1127)
at com.mysql.jdbc.MySQLIO.<init>(MySQLIO.java:356)
at com.mysql.jdbc.ConnectionImpl.coreConnect(ConnectionImpl.java:2502)
at com.mysql.jdbc.ConnectionImpl.connectToTryOnly(ConnectionImpl.java:2539)
at com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:2321)
    
```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 14 de 30



Tipos de Exceções - Java

- **Unchecked (não verificada)**- não exigem nenhum tratamento por parte do programador
 - Todas as classes que são filhas de **RuntimeException** não precisam ser tratadas, assim como aquelas que são filhas da classe **Error**.
 - **RuntimeException**, temos as famosas
 - **NullPointerException**, **ClassCastException** e
 - **ArrayIndexOutOfBoundsException**
- **checked (verificada)**, aquelas exceções que devem ser tratadas por parte do programador usando **try..catch**. As filhas de **Exception** exigem tratamento, pois são consideradas **checked**.

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 17 de 30

Exceções Não-verificadas

- Lançadas automaticamente por Java
- Derivadas da classe **RuntimeException** ou **Error**
- Não precisam (mas, podem) ser especificadas usando **throws**

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 18 de 30

Exceções Não Verificadas

```

3 public class Teste2 {
4
5     public static void main(String[] args) {
6         int numeros[] = { 40, 50, 60 };
7
8         try {
9             System.out.println(numeros[3]);
10        } catch (Exception e) {
11
12        }
13        System.out.println("fim do programa");
14    }
15 }

```

“Não suprima ou ignore as exceções”

Quando um método de uma API lança uma exceção, ele está tentando te dizer que você deve fazer algo. Se a exceção não faz o menor sentido para você não hesite em convertê-la em uma *unchecked exception* e lance-a novamente. **Mas nunca, nunca, nunca ignore-a tratando-a com `catch` e continuar como se nada tivesse acontecido.**

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 19 de 30

Exceções Verificadas

- Não são lançadas automaticamente
- Derivadas de classes de exceções que não derivadas de **RuntimeException**
- Devem ser especificadas usando **throws**
- Se um método chama outro método que lança exceções verificadas, estas exceções devem ser especificadas na cláusula **throws** do método que faz a chamada, a não ser que este método capture esta exceção

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 20 de 30

Exceções Verificadas

```

21 Connection con = null;
22 String driver = "com.mysql.jdbc.Driver";
23 String usuario = "root";
24 String senha = "wander2017";
25 String url = "jdbc:mysql://127.0.0.1/faculdade";
26 try {
27
28     Class.forName(driver);
29     con = DriverManager.getConnection(url, usuario, senha);
30 } catch (ClassNotFoundException e) {
31     System.out.println("erro não foi encontrada a classe");
32     e.printStackTrace();
33
34 } finally{
35     System.out.println("sem executa");
36 }

```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 21 de 30

Tipos de Exceções

- Qual a diferença entre exceções verificadas e não-verificadas?

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 22 de 30

Capturando e Tratando Exceções

- **Bloco try** - O código que pode gerar uma exceção deve ser incluído em um **bloco try**
- **Bloco catch** - As exceções devem ser tratadas em **blocos catch**
- **Bloco Finally** – Sempre será executado ocorrendo ou não uma exceção

Qualquer exceção ocorrida em uma linha dentro do try, transferirá o controle para o bloco catch.

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 23 de 30

Bloco try

- Bloco de instruções no qual algumas exceções **podem ser lançadas**
- Usa palavra-reservada **try** seguido de um bloco de instruções entre chaves
- **Sintaxe:**

```

try {
    <Instruções que podem lançar exceções>;
}

```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 24 de 30

Bloco catch

Tratador de exceção: **captura exceções**

O **bloco catch** recebe um parâmetro da classe **Exception**

- Importante** : Se uma exceção é lançada durante a execução de um bloco try este imediatamente termina e o controle passa para o bloco catch apropriado

Sintax:

```
catch(<classe de exceção> <argumento>) {
    <instruções>;
}
```

- Onde: <classe de exceção> é classe de exceção que o tratador pode capturar e <argumento> é um identificador que armazena o valor da exceção lançada

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 25 de 30

Múltiplos catches

- É possível utilizar vários *catches* para um único *try*, como um switch:
- No caso do catch múltiplo, o primeiro bloco que for selecionado é executado, **enquanto que os outros são ignorados**

Exemplo :

```
try {
    // código que poderia gerar exceção
} catch (IOException e) {
    // trata exceções de entrada e saída
} catch (ClassNotFoundException e1) {
    // exceções de classe não encontrada
} catch (InterruptedException e2) {
    // trata de exceções interrompida
}
```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 26 de 30

Bloco Finally

A clausa finally, quando colocada no final dos blocos catch, define um trecho de programa que será sempre executado, quer ocorra exceção ou não:

O bloco finally deve vir após o último bloco catch

Exemplo

```
try {
    readTextFile();
} catch (IOException e) {
    // trata erros de IO
} finally {
    closeTextFile();
}
```

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 27 de 30

Referências Bibliográficas

BRUCE, Eckel. **Thinking in Java**. 4 ed. Massachusetts: Editora Prentice hall: 2006.

FOWLER, Martin. **UML essencial : um breve guia para a linguagem** – padrão de modelagem de objetos. Tradução Vera Pezerico e Christian Thomas Pries. 3. ed. Porto Alegre : Bookman, 2005.

FURLAN, Jose Davi. **Modelagem de Objetos Através da UML** São Paulo: Editora Makron Books, 2000, ISBN 8534609241,

GRADY, Booch; RUMBAUGH, James; JACOBSON, Ivar. **UML, guia do usuário** . tradução de Fábio Freitas da Silva. Rio de Janeiro: Editora Campus. 2000.

SIERRA, Kathy; BATES, Bert. **Certificação Sun para programadores e desenvolvedor Java 2**. Rio de Janeiro: Editora Altas Books. 2003.

SIERRA, Kathy; BATES, Bert. **Head first java 5. 2 ed.** Sebastopol: Editora O' Reilly Media. Books. 2005.

TOM, Pender. **UML, a bíblia**. Rio de Janeiro: Elsevier, 2004.

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 28 de 30

Perguntas



Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 29 de 30

Obrigado

Wanderley

Wanderley.unip@gmail.com

www.wg.pro.br

Prof: Msc Wanderley Gonçalves Freitas LPOO 09 – Tratamento de Exceção 30 de 30