Hortolândia, 20 de novembro de 2013.

Aulas 65 e 66 – Tratamento de Exceções em Java

1. Introdução

Na presente aula será abordado o assunto de **tratamento de exceções** em Java.

Uma exceção representa uma situação que normalmente não ocorre e representa algo de estranho ou inesperado no sistema.

A linguagem de programação, dando suporte ao tratamento de exceções, permite que a aplicação continue sendo executada (permaneça "em pé"), mesmo na ocorrência de uma situação de exceção.

A Figura 1 ilustra a **hierarquia de exceções** no Java.

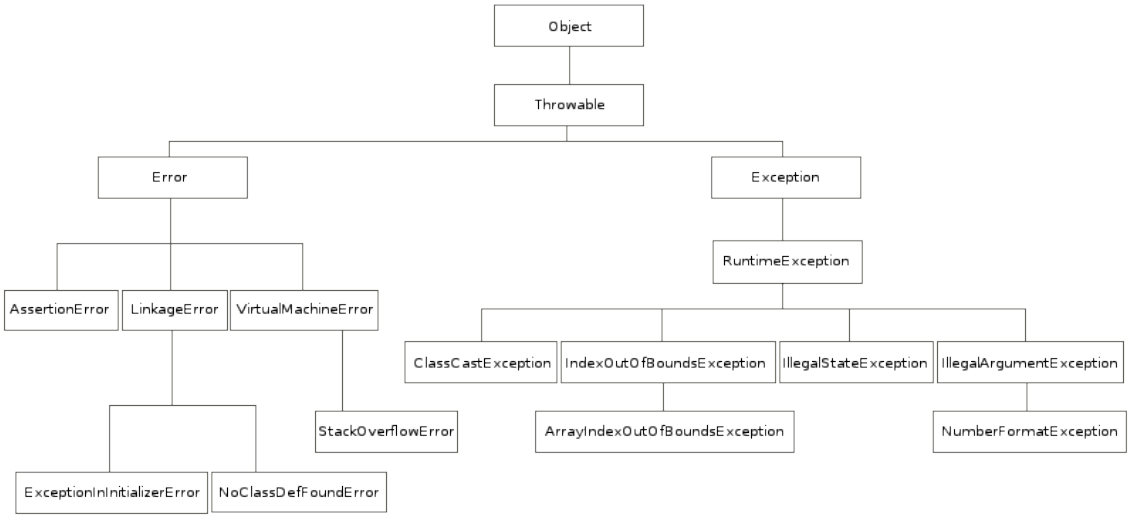

Figura 1 – Hierarquia de Exceções do Java.

2. Blocos try, catch e finally

Sempre que um método de alguma classe é passível de causar algum erro, então, podemos usar o método de tentativa "**try**".

Tudo que estiver dentro do bloco **try** será executado até que alguma exceção seja lançada, ou seja, até que algo dê errado.

Quando uma exceção é lançada, ela sempre deve ser capturada. O trabalho de captura da exceção é executado pelo bloco **catch**.

Um bloco **try** pode possuir vários blocos de **catch**, dependendo do número de exceções que podem ser lançadas por uma classe ou método.

O bloco **catch** obtém o erro criando uma instância da exceção. Portanto, a sintaxe do bloco try catch é:

```
try {
  // código a ser executado
} catch (ClasseDeExceção instânciaDaExceção) {
```

```
      // tratamento da exceção

    }
```

3. Classes de Exceções

As exceções são, na verdade, instâncias de classes. E como qualquer classe ou objeto, podemos facilmente manipular.

Existem **métodos comuns** entre todas as **classes de Exceções**, dentre os quais podem-se citar:

- **toString()**: Converte os dados da exceção para String para visualização.
- **printStackTrace()**: Imprime na saída de erro padrão (geralmente console) todos os frames de onde foram detectados erros. Útil para depuração no desenvolvimento, pois mostra todo o histórico do erro, além das linhas onde foram ocasionados.
- **getCause()**: Retorna a causa da Exceção, ou null se a causa for desconhecida ou não existir.
- **getMessage()**: Retorna uma string com o erro. É uma forma simples e elegante de mostrar a exceção causada, e é geralmente utilizada como forma de apresentação ao usuário.

4. Bloco finally

Finally é o trecho de código final. A função básica de **finally** é sempre executar seu bloco de dados mesmo que uma exceção seja lançada.

É muito útil para liberar recursos do sistema quando utilizamos, por exemplo, conexões de banco de dados e abertura de *buffer* para leitura ou escrita de arquivos.

Finally virá após os blocos de **catch**.

```
    try {
      // código a ser executado
    } catch (ClasseDeExceção instânciaDaExceção) {
      // tratamento da exceção
    } finally {
       // código a ser executado mesmo que uma exceção seja
lançada
    }
```

5. Exemplo de utilização de Exceção

A seguir apresenta-se um exemplo de código em Java com tratamento de erro ou exceção.

Será tentado atribuir uma string de letras a um objeto inteiro.

Como não é possível atribuir uma string de letras a um objeto inteiro, uma exceção de formato de número será lançada.

```
public class ExemploDeExcecao {

 public static void main(String[] args) {

 String var = "ABC";

 try {

       Integer i = new Integer(var);

       System.out.println("A variável i vale " + i);

 } catch (NumberFormatException nfe) {

 System.out.println("Não é possível atribuir a string " +
var + " a um Objeto Inteiro.\n" + "A seguinte mensagem foi
retornada:\n\n" + nfe.getMessage());

 }

 }

}
```

O código acima apresentará algo como:

Não é possível atribuir a string ABC a um Objeto Inteiro.

A seguinte mensagem foi retornada:

For input string: "ABC"

Perceba que a linha **System.out.println("A variável i vale " + i)** não foi executada, pois houve um erro na linha anterior.

Portanto, apenas a mensagem de tratamento do erro **NumberFormatException** será impressa na tela, com a execução desse trecho de código.

6. Referências Bibliográficas

BORATTI, Isaias Camilo. **Programação Orientada a Objetos usando Delphi.** Quarta Edição. Editora Visual Books. Florianópolis, 2007.

DEITEL, H.M., DEITEL, P.J. **Java – Como programar**. Terceira edição. Porto Alegre: Bookman Editora, 2001.

HORSTMAN, C. ***Conceitos de Computação com o Essencial de Java***. 3ª Edição. Porto Alegre: Bookman, 2003.

SANTOS, Rafael. **Introdução à Programação orientada a objetos usando Java**. 8ª Reimpressão. Rio de Janeiro: Campus - Elsevier, 2003.

TI EXPERT.NET. ***Try, Catch e Finally - Tratamento de Exceções e Erro***. URL: http://www.tiexpert.net/programacao/java/try-catch-finally.php. Última consulta: 20/11/2013.