

Unidade 3

Seção 1

Programação Orientada a Objetos

Webaula 1

Definição e tratamento de exceções.

Neste webaula, trataremos dos recursos que o desenvolvedor dispõe para tratar as exceções que podem ocorrer durante a execução de uma aplicação em Java.



As exceções em Java se referem aos erros que podem ser gerados durante a execução de um programa.

Como o próprio nome sugere, trata-se de algo que interrompe a execução normal do programa. É um problema que não ocorre frequentemente.

O tratamento da exceção serve justamente para que o programa possa continuar sendo executado, ao invés de ser encerrado repentinamente.

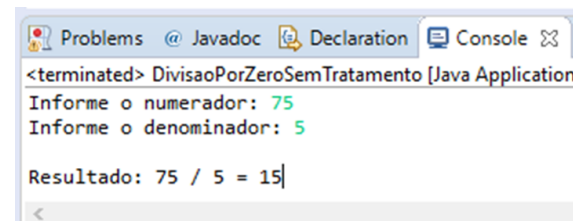
Veja o exemplo de um programa que recebe dois valores inteiros e calcula, por meio do método quociente, o resultado da divisão entre os dois números:

```
import java.util.Scanner;
public class DivisaoPorZeroSemTratamento {
    public static int quociente (int numerador, int
denominador) {
        return numerador / denominador;
    }
    public static void main(String args[]) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Informe o numerador: ");
        int numerador = entrada.nextInt();
        System.out.print("Informe o denominador: ");
        int denominador = entrada.nextInt();
        int resultado = quociente(numerador, denominador);
        System.out.printf("\nResultado: %d / %d = %d",
numerador, denominador, resultado);
    }
}
```

Fonte: adaptado de Deitel e Deitel (2010).

Explore a galeria para entender o tratamento de exceções do programa exemplificado:

De acordo com o código, uma execução possível – e sem entradas com potencial para causar erro – para este programa é:

A screenshot of an IDE's console window. The title bar shows tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output shows the program 'DivisaoPorZeroSemTratamento [Java Application]' has terminated. It displays two prompts: 'Informe o numerador: 75' and 'Informe o denominador: 5'. The final output is 'Resultado: 75 / 5 = 15'.

```
<terminated> DivisaoPorZeroSemTratamento [Java Application]  
Informe o numerador: 75  
Informe o denominador: 5  
Resultado: 75 / 5 = 15
```

Fonte: elaborado pelo autor.

Em Java, as exceções são divididas em duas categorias (FURGERI, 2013). Clique nas abas para ver o conteúdo:

Unchecked Exception

Significa “exceção não verificada”. Neste tipo de exceção, o Java não verifica o código-fonte para determinar se a exceção está sendo capturada. Por isso, o tratamento aqui é opcional. Fazem parte dessas exceções de tratamento opcional, por exemplo, a verificação de acesso a um índice inexistente num vetor, a tentativa de se usar um método de um objeto ainda não instanciado e a conversão de um `String` em inteiro.

Checked Exception

No Java, a estrutura que trata as exceções é formada pelos comandos `try-catch-finally`:

```
try {  
    comandos  
} catch (exceção_tipo1 identificador1) {  
    comandos  
} catch (exceção_tipo2 identificador2) {  
    comandos  
    ...  
} finally {  
    comandos  
}
```

Fonte: elaborado pelo autor.

Essa estrutura pode ser usada, tanto com *Unchecked Exceptions*, como com *Checked Exceptions* e tem como função desviar a execução de um programa, caso ocorram certos tipos de erro predefinidos durante o processamento das linhas.

✓ **try {..}**: neste bloco, são escritas todas as linhas de código que podem vir a lançar uma exceção;

✓ **catch** (tipo_excecao e) { ... }: neste bloco é descrita a ação que ocorrerá quando a exceção for capturada;

✓ **finally** é opcional e fornece um conjunto de códigos que é sempre executado, independentemente da ocorrência da exceção. O uso do finally pode ser exemplificado por meio de operações de banco de dados.



Criação de tipos de exceções

A linguagem Java oferece controle para geração e tratamento de muitas exceções, em que o desenvolvedor Java pode criar suas próprias exceções e dispará-las quando necessitar.

A instrução `throw` (que não deve ser confundida com a cláusula `throws`) serve para forçar a ocorrência de uma determinada exceção.

Exceções são objetos e todos os seus tipos devem estender a classe `Throwable` ou uma de suas subclasses. A classe `Throwable` possui uma cadeia de caracteres que pode servir como descritor da exceção e recuperada com o método `getMessage`.



Ligando e desligando asserções

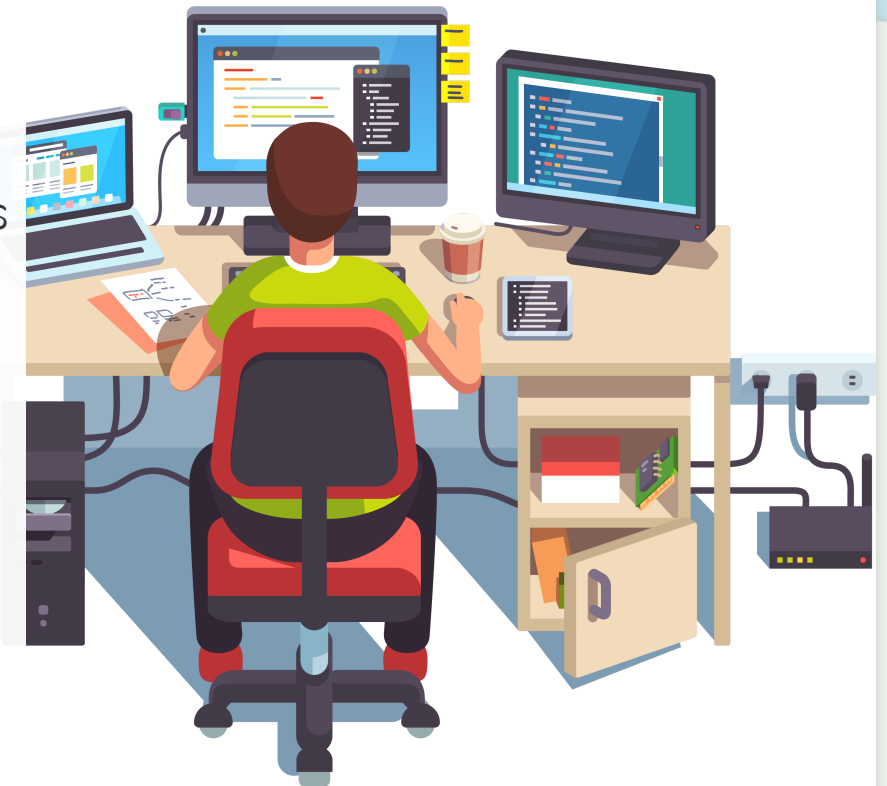
Por padrão, a avaliação das asserções permanece desligada. O desenvolvedor, no entanto, pode ligar e desligar toda a avaliação de asserções para classes e pacotes específicos.

O controle de liga/desliga das asserções deve ser feito, preferencialmente, na linha de comando de execução da aplicação, embora possa também ser feito no código.

Pela linha de comando a forma geral dos comandos é:

✓ `enableassertions /-ea [descriptor]`: habilita a avaliação das asserções conforme definido em descriptor. Caso descriptor não seja definido, as asserções são carregadas para todas as classes;

✓ `disableassertions /-da [descriptor]`: desabilita a avaliação das asserções para todos os membros definidos pelo descriptor. Caso o descriptor não exista, as asserções serão desabilitadas para todas as classes.



Busque mais informação e resolva exercícios para entender mais sobre o tratamento de exceções.



Você já conhece o Saber?

Aqui você tem na palma da sua mão a **biblioteca digital** para sua **formação profissional**.

Estude no celular, tablet ou PC em qualquer hora e lugar sem pagar mais nada por isso.

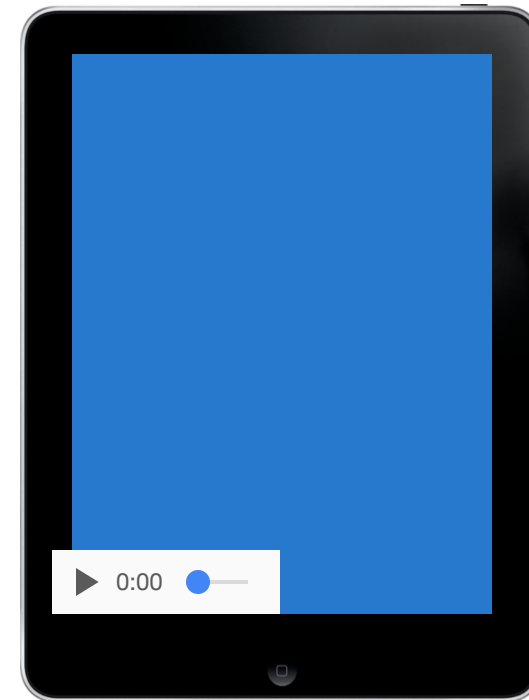
Mais de 450 livros com interatividade, vídeos, animações e jogos para você.



Android:
<https://goo.gl/yAL2Mv>



iPhone e iPad - IOS:
<https://goo.gl/OFWqcq>





Bons estudos!

