**Tratamento de exceções em Java**

**Otávio A. A. Silva¹**

**¹ Faculdade de Engenharia da Computação e Telecomunicações - Instituto de Tecnologia - Universidade Federal do Pará**

**Cidade Universitária “Prof. José da Silveira Netto” – Campus II (Profissional) Belém/PA, Brasil, 66075-110**

tavioalves@gmail.com

***Abstract.*** *This brief article will describe meta-concepts about exceptions in the Java programming language. Showing and exemplifying through an application benefits from its use.*

***Resumo.*** *Este meta-artigo descreverá breves conceitos a cerca de exceções na linguagem de programação Java. Mostrando e exemplificando através de um aplicativo os benefícios da sua utilização.*

**1. Exceções**

É uma indicação de um problema que ocorre durante a execução do programa. O nome exceção significa que problema não ocorre frequentemente, fugindo a regra. Com o tratamento de exceções um programa pode continuar executando em vez de encerrar. Ou seja, ao lidar com os problemas o seu programa continua com o seu funcionamento e isso ajuda a assegurar a robustez dos seus aplicativos, colaborando para que o programa se torne tolerante a falhas, lidando com os problemas a medida que eles surgem.

Em momentos como um usuário atribui os argumentos de entrada que não são válidos (um denominador igual a zero em uma divisão, por exemplo) ou quando o usuário tenta executar um recurso no qual o programa depende não está disponível (um arquivo não encontrado ou um erro em um disco, por exemplo), uma rotina de exceções seria essencial para que o programa não perdesse o seu fluxo, gerando uma mensagem de erro para o usuário e mostrando o que deve ser feito.

**2. Vantagens e desvantagens do uso de exceções**

As exceções verificadas podem, em tempo de compilação, reduzir consideravelmente (mas não eliminar inteiramente) a ocorrência de casos em que de exceções não tratadas emergem para fora da aplicação; as exceções não verificadas (RuntimeExceptions e Errors) podem permanecer sem tratamento.

Porém, alguns criticam as exceções verificadas alegando que esta sintaxe requer assinaturas de métodos com extensas declarações de cláusula throws, que freqüentemente revelam detalhes da implementação interna e reduzem o [encapsulamento](http://pt.wikipedia.org/wiki/Encapsulamento), ou ainda encorajam o abuso de blocos try/catch mal desenhados, que podem esconder a ocorrência de exceções das rotinas que deveriam tratá-las.[8](http://pt.wikipedia.org/wiki/Tratamento_de_exce%C3%A7%C3%A3o" \l "cite_note-ibm-8) Além disso, o uso destas exceções pode dificultar a manutenção dos programas.

**3. Aplicação de uma rotina de exceção**

O uso de uma rotina de exceções em um código mostrasse indispensável para o tratamento de erros em tempo de execução. Em programas onde são realizadas operações matemáticas, alguns erros operacionais podem ocorrer, como a divisão de um número por zero, gerando então uma inconsistência matemática. Para o computador, a divisão entre 1 e 0(1/0) por exemplo, resultaria em um valor infinito, ou seja, um erro. Com a utilização de exceções podemos mostrar ao usuário o erro que está cometendo e realizar a correção da mesma, se que seja necessário uma interrupção drástica no fluxo do programa.

**4. Código**

A seguir uma aplicação no qual o programa pergunta a usuário, um dividendo e um divisor, para ocorrer á divisão. Caso ocorra algum erro(divisão por zero ou colocação de um caractere), irá ocorrer uma rotina de exceção para que o usuário possa colocar o divisor de maneira correta. Enquanto o usuário persista no erro, mensagens dos seus erros serão gerados e a execução do programa irá ocorrer novamente, ou seja, um loop. No momento que os dados forem colocados de maneira correta, o programa mostra o resultado e então é finalizado.

**boolean** continua = **true**;

*s* = **new** Scanner(System.*in*);

**do**{

**try**{

System.*out*.println("\t\t-- Divisão -- ");

System.*out*.print("Dividendo: ");

**int** a = *s*.nextInt();

System.*out*.print("Divisor: ");

**int** b = *s*.nextInt();

System.*out*.println("Resultado: "+a/b);

continua = **false**;

}

**catch**(ArithmeticException e1){

System.*out*.println("Erro ! Digite um divisor diferente de zero!");

*s*.nextLine();

}

**catch**(InputMismatchException e2){

System.*out*.println("Erro ! Digite um divisor inteiro!");

*s*.nextLine();

}

}**while**(continua);.

**5. Referencias**

Tratamento de exceções -<http://pt.wikipedia.org/wiki/Tratamento_de_exce%C3%A7%C3%A3o>

Divisão por zero **-** <http://pt.wikipedia.org/wiki/Divis%C3%A3o_por_zero>

Universidade XTI – Exceções: <http://www.youtube.com/watch?v=EHnIw3Fn_qk>