TRABALHO DE PESQUISA

TRATAMENTO DE EXCEÇÕES EM JAVA

Aluno: Matheus Henrique

Disciplina: Desenvolvimento de Sistemas

Tema: Tratamento de Exceções em Java

Introdução

O tratamento de exceções em Java é um mecanismo essencial para garantir que programas sejam capazes de lidar com situações inesperadas durante sua execução. Exceções podem ocorrer por diversos motivos, como erros de entrada de dados, falha ao abrir arquivos, problemas de conexão, entre outros. O uso adequado do tratamento de exceções permite que o software continue funcionando de forma segura, evitando interrupções abruptas e fornecendo mensagens úteis para o programador e para o usuário.
Este trabalho apresenta os principais conceitos envolvendo exceções em Java, sua finalidade, tipos, estrutura básica dos comandos utilizados e boas práticas recomendadas.

1. Conceito e Finalidade do Tratamento de Exceções

Uma exceção é um evento que interrompe o fluxo normal de execução de um programa. O tratamento de exceções tem como principal finalidade capturar esses eventos indesejados e tratá-los de maneira adequada, evitando que o programa seja finalizado de forma abrupta.
Com esse mecanismo, o programador consegue prever possíveis falhas e definir ações específicas para lidar com elas, tornando o software mais seguro e confiável.

2. Diferença entre Erros e Exceções

No Java, erros (**Errors**) e exceções (**Exceptions**) não são a mesma coisa:

- **Exceções (Exceptions):** São situações que podem ocorrer durante a execução e que o programador consegue prever e tratar, como divisões por zero, arquivos inexistentes ou entradas inválidas do usuário.
- **Erros (Errors):** São problemas mais graves, geralmente relacionados ao ambiente da JVM, como falta de memória (**OutOfMemoryError**) ou falha no carregamento de uma classe. Esses erros não devem ser tratados, pois indicam falhas no próprio ambiente de execução e não no código do programador.

3. Tipos de Exceções em Java: Checked e Unchecked

As exceções em Java são divididas em duas categorias:

a) Checked Exceptions

São exceções verificadas em tempo de compilação. O compilador exige que o programador trate essas exceções obrigatoriamente, usando try/catch ou lançando com throws.
Exemplos comuns:

- IOException
- SQLException

b) Unchecked Exceptions

Também chamadas de **Runtime Exceptions**, são exceções que ocorrem em tempo de execução e não precisam ser tratadas obrigatoriamente.
Exemplos comuns:

- NullPointerException
- ArithmeticException
- ArrayIndexOutOfBoundsException

4. Estrutura Básica do try, catch, finally e throw

O Java fornece palavras-chave específicas para trabalhar com exceções:

try

É o bloco onde colocamos o código que pode gerar uma exceção.

catch

Captura a exceção gerada no bloco try, permitindo que o programa execute uma ação para tratá-la.

finally

Bloco opcional que sempre é executado, independentemente de ter ocorrido ou não uma exceção.
É muito usado para fechar arquivos, conexões ou liberar recursos.

throw

Usado para lançar uma exceção manualmente dentro do código.

Exemplo simples:

```
try {
    int resultado = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Erro: divisão por zero!");
} finally {
    System.out.println("Finalizando operação...");
}
```

5. Boas Práticas no Tratamento de Exceções

- Tratar apenas exceções que o programa realmente precisa lidar.
- Evitar capturar exceções genéricas como Exception, a menos que seja necessário.
- Escrever mensagens claras ao usuário.
- Registrar erros usando logs.
- Nunca ignorar exceções.
- Não usar exceções como controle de fluxo.
- Sempre usar finally ou try-with-resources ao trabalhar com arquivos e conexões.

6. Exemplos de Aplicações Reais

a) Leitura de Arquivos

Ao tentar abrir um arquivo, caso ele não exista, uma IOException pode ocorrer. O tratamento da exceção evita que o sistema pare de funcionar.

b) Acesso a Banco de Dados

Métodos que fazem consultas e conexões lançam exceções do tipo SQLException, que precisam ser tratadas ou declaradas.

c) Validação de Entrada do Usuário

Entradas inválidas podem gerar exceções como NumberFormatException. Tratar isso garante que o sistema peça ao usuário uma nova entrada, sem travar.

Conclusão

O tratamento de exceções em Java é uma parte fundamental do desenvolvimento de aplicações robustas e seguras. Ele permite que o programa lide com situações inesperadas de forma organizada, evitando que erros interrompam sua execução. Além disso, quando aplicado com boas práticas, contribui para a clareza, segurança e qualidade do código. Compreender como tratar exceções corretamente é essencial para qualquer programador Java, independentemente do nível de experiência.

Referências

DEITEL, Paul; DEITEL, Harvey. *Java: Como Programar*. 10. ed. São Paulo: Pearson, 2016.

ORACLE. *Java Documentation*. Disponível em: https://docs.oracle.com/javase/. Acesso em: 17 nov. 2025.