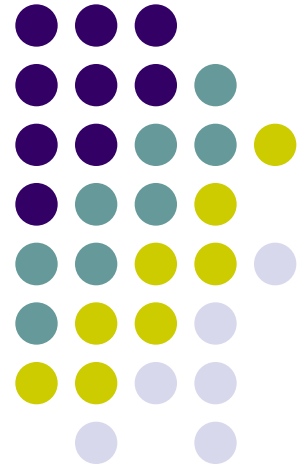


Tratamento de Exceções

Alberto Costa Neto
DComp - UFS



Erros e Exceções

[Conteúdo]



- Introdução
- Hierarquia de Erros e Exceções
- Erros e Exceções predefinidos
- Bloco try
- Bloco finally
- Cláusula throws
- Lançando exceções
- Capturando e relançando uma exceção
- Criando uma classe de exceção
- Exercícios
- Bibliografia

Erros e Exceções

[Introdução]



- Uma exceção é um desvio do fluxo normal de execução de um programa provocado por um evento
- Um evento por ser, por exemplo:
 - Arquivo não encontrado
 - Falta de memória
 - Acesso a um elemento inexistente de um array
 - Divisão por zero
 - Conexão com SGBD fechada inesperadamente
 - Tentativa de conversão de uma String (contendo um número inválido) para um número

Erros e Exceções

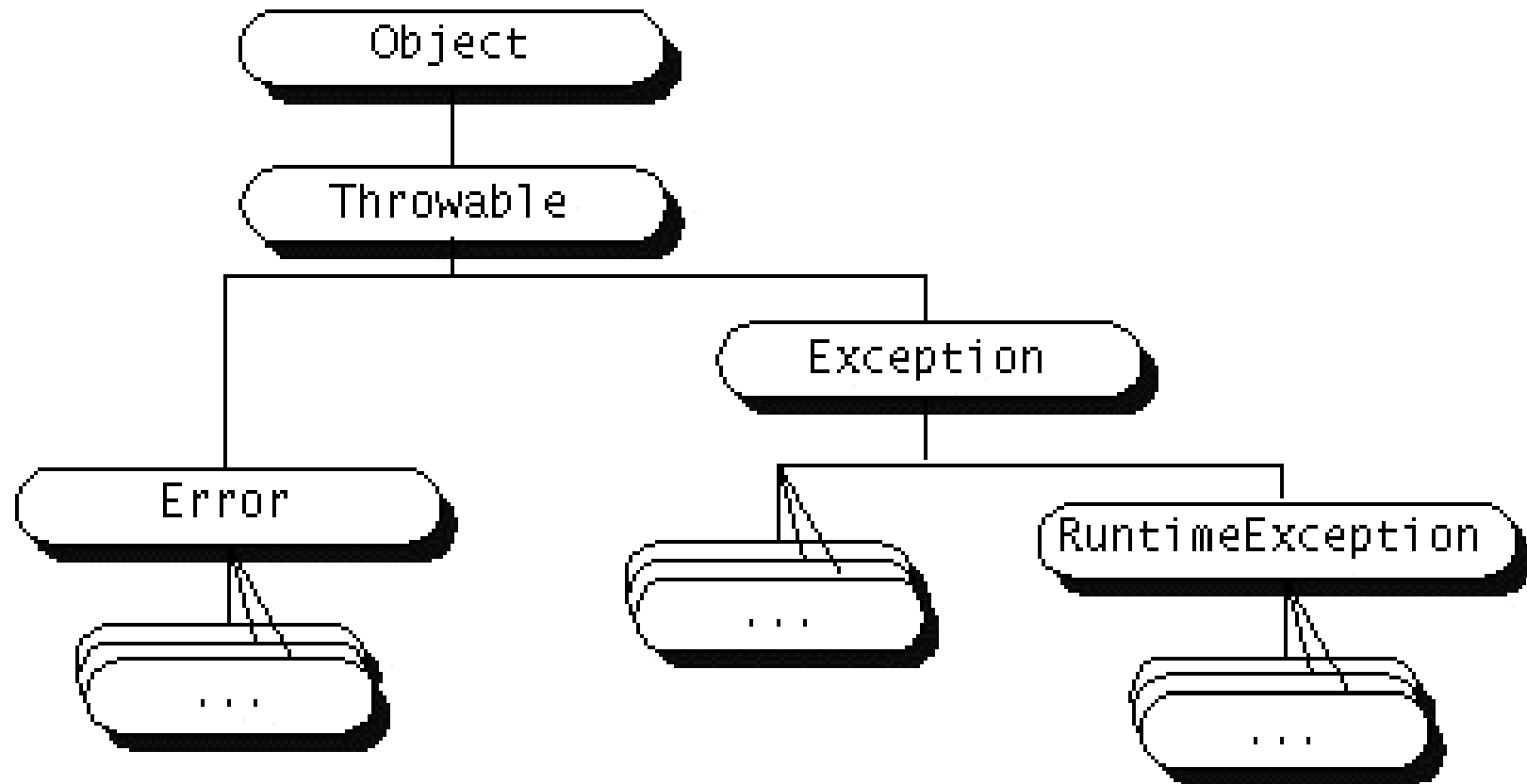
[Introdução]



- Incluir na aplicação código para identificar esses tipos de erro dificulta a programação
 - Os comandos que executam a lógica real ficam perdidos no código que verifica as situações de erro
 - Dificulta a propagação do erro entre as camadas da aplicação

Erros e Exceções

[Hierarquia de Erros e Exceções]



Erros e Exceções

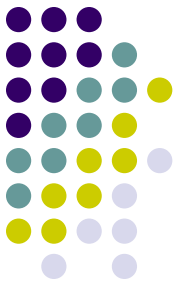
[Hierarquia de Erros e Exceções]



- **Throwable**
 - É a classe pai de todas as classes de erros e exceções
 - Métodos
 - **printStackTrace:** Imprime uma descrição sobre a exceção e a pilha de execução (seqüência de chamadas de métodos) até o ponto onde foi lançada
 - **getMessage:** Retorna a mensagem de erro passada no construtor
 - **toString:** Retorna uma descrição curta
 - **fillStackTrace:** Preenche a pilha de execução (útil ao relançar uma exceção)

Erros e Exceções

[Hierarquia de Erros e Exceções]



- Error
 - Quando uma falha de ligação dinâmica ou alguma outra falha na JVM, ela lança um Error
 - Os programas Java normais não devem capturá-los.
 - Porém, é provável que programas Java típicos nunca lancem um Error.

Erros e Exceções

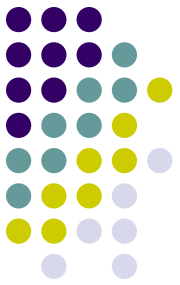
[Hierarquia de Erros e Exceções]



- Exception:
 - Exceptions indicam que um problema ocorreu mas que não é um problema sério.
 - A maioria dos programas devem lançar ou capturar Exceptions
 - A classe Exception tem muitos descendentes predefinidos
 - Esses descendentes indicam os vários tipos de exceções que podem ocorrer
 - Exemplo:
 - `NegativeArraySizeException` indica que um programa tentou criar um array com um tamanho negativo

Erros e Exceções

[Hierarquia de Erros e Exceções]



- **RuntimeException:**
 - A classe RuntimeException representa exceções que ocorrem na JVM (durante a execução)
 - Um exemplo é `java.lang.NullPointerException`, que ocorre quando um método tenta acessar um membro de um objeto através de um referência contendo o valor null
 - O custo de checar por esta exceção a todo momento seria muito grande pois a todo momento são acessados membros de objetos

Erros e Exceções

[Erros e Exceções predefinidos]



- Descendentes de Exception
 - **java.lang.ClassNotFoundException:** Lançada quando uma aplicação tenta carregar uma classe através de seu nome (Class.forName), mas a definição da classe não foi encontrada
 - **java.io.IOException:** Indica que uma exceção de Entrada ou Saída de algum tipo ocorreu. É a classe pai de várias exceções de E/S.
 - **java.sql.SQLException:** Exceção que indica algum erro ocorrido ao acessar um SGBD

Erros e Exceções

[Erros e Exceções predefinidos]



- Descendentes de Error
 - **java.lang.NoClassDefFoundError:** A máquina virtual não consegue encontrar a definição de uma classe (arquivo .class)
 - **java.lang.OutOfMemoryError:** Ocorre ao tentar alocar memória para um objeto e não há mais memória disponível, mesmo após a coleta de lixo
 - **java.lang.StackOverflowError:** Quando uma aplicação provoca um estouro de pilha

Erros e Exceções

[Erros e Exceções predefinidos]



- Descendentes de RuntimeException
 - **java.lang.ArithmeticException:** Lançada quando ocorre uma condição aritmética excepcional, como uma divisão por zero
 - **java.lang.ArrayIndexOutOfBoundsException:** Ocorre quando tenta-se acessar um índice do array que não existe
 - **java.lang.StringIndexOutOfBoundsException:** Lançada para indicar uma tentativa de acessar um caractere em um índice inexistente

Erros e Exceções

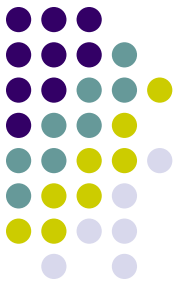
[Erros e Exceções predefinidos]



- **java.lang.ClassCastException:** Lançada quando tenta-se fazer um type cast incorreto
- **java.lang.IllegalArgumentException:** Indica que o valor de algum argumento é inválido
- **java.lang.NullPointerException:** Ocorre quando tenta-se chamar acessar um membro de um objeto e a referência contém null

Erros e Exceções

[Bloco try]



- Vários tipos de exceções podem e devem ser tratados.
- Para isso, existe o bloco try (tentar), que consiste em um bloco de comandos no qual as exceções lançadas, poderão ser tratadas em um do(s) bloco(s) catch (capturar).

Erros e Exceções

[Bloco try]



- Sintaxe:

```
try {  
    // Comandos que podem lançar exceções  
} catch (ClasseExcecaoA ea) {  
    // Comandos que tratam a exceção ClasseExcecaoA  
} catch (ClasseExcecaoB eb) {  
    // Comandos que tratam a exceção ClasseExcecaoB  
} catch (ClasseExcecaoN en) {  
    // Comandos que tratam a exceção ClasseExcecaoN  
} finally {  
    // Comandos sempre são executados  
}
```

Erros e Exceções

[Bloco try]



- Funcionamento do try
 - Os comandos do try são executados seqüencialmente
 - Caso não ocorram exceções, nenhum dos comandos contidos dos catches será executado
 - Caso ocorra uma exceção, desvia-se o fluxo de controle do try e busca-se por um catch cuja classe de exceção seja igual ou ancestral da exceção lançada
 - Se for encontrado, os comandos nele contidos são executados
 - Se não for, o método irá repassar a exceção para quem executou o método que contém o try

Erros e Exceções

[Bloco try]



- Exemplo
 - InstanciarClasse

Erros e Exceções

[Bloco finally]



- Os comandos contidos no finally (caso exista) são sempre executados, seja logo após o último comando do try ou após o último comando de um dos catches
- Serve para garantir a liberação de recursos alocados no try (arquivos abertos, conexões com SGBD's abertas, etc.)
- Exemplo
 - ExemploFinally

Erros e Exceções

[Cláusula throws]



- Em algumas situações não é possível tratar uma exceção onde ela ocorre
- A cláusula throws permite que um método deixe de tratar um exceção ocorrida durante a sua execução e repasse essa responsabilidade para quem o chama
- Exemplo
 - ExemploThrows

Erros e Exceções

[Lançando exceções]



- Além das exceções que acontecem devido a certos eventos, é permitido lançar uma exceção a qualquer momento
- Para fazê-lo, utiliza-se o comando **throw** seguido de uma instância da exceção
- A exceção `IllegalArgumentException` poderia ser lançada quando algum dos parâmetros tem um valor inválido (A detecção do valor inválido depende da semântica da classe)
- Exemplo
 - `ExemploThrow`

Erros e Exceções

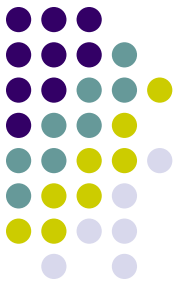
[Capturando e relançando uma exceção]



- Há situações em que deseja-se mudar o tipo da exceção ou de acordo com algum critério relançá-la ou não
- Como exemplo, seria possível a partir de uma `SQLException` detectar que o servidor não está ativo e tentar a conexão com outro. Caso se tratasse de outro tipo de erro (Comando SQL inválido), não haveria solução senão relançar a exceção
- Exemplo
 - CapturarRelancar

Erros e Exceções

[Criando uma classe de exceção]



- Para criar uma exceção é necessário criar uma classe que seja descendente de Throwable
- Exemplo
 - DateException

Erros e Exceções

[Exercícios]



- 1) Crie um programa que provoque o erro **OutOfMemory**. Para isso, crie uma instância de **LinkedList** e execute um laço infinito no qual, a cada iteração, é inserido um novo objeto.
- 2) Implementar uma classe de exceção chamada **ParametroInvalido** que armazene o nome do parâmetro e o valor considerado inválido

Erros e Exceções

[Bibliografia]



- Deitel (capítulo 14)
- Core Java 2 – Fundamentals (capítulo 11)