Exceções

Prof. Ítalo Assis

Ajude a melhorar este material =]

Encontrou um erro? Tem uma sugestão?

Envie e-mail para italo.assis@ufersa.edu.br

Agenda

- Exceções
- A instrução *try...catch...finally*
- Hierarquia de exceção Java
- Rastreamento de pilha
- Exceções verificadas e não verificadas
- Cláusula *throws*

Prática

- Escreva um programa que utiliza um *array* para representar as respostas de um aluno a um questionário com 10 perguntas
- O questionário possui as alternativas 0 a 5 para cada questão
- Seu programa deve contar o total de ocorrências de cada uma das respostas
- O que acontece se usuário digitar a alternativa 14 para uma das questões?

Exceções

- Uma exceção indica um problema que ocorre quando um programa é executado
- Alguns problemas podem evitar que um programa continue executando
  - Exemplos: índice de *array* inválido, tipo de dado de entrada incorreto, divisão por zero...
- Quando a JVM ou um método detecta um problema, ele lança uma exceção
  - Os métodos nas suas classes também podem lançar exceções, como veremos em breve
- O tratamento de exceção ajuda a criar programas tolerantes a falhas
  - Em muitos casos, isso permite que um programa continue a executar como se nenhum problema tivesse ocorrido

A instrução *try*... *catch*

- Para lidar com uma exceção, coloque qualquer código que possa lançar uma exceção em uma instrução *try*
- O bloco *try* contém o código que pode lançar uma exceção
- O bloco *catch* contém o código que trata a exceção se uma ocorrer
    - Podem haver muitos blocos *catch* para tratar com diferentes tipos de exceções que podem ser lançadas no bloco *try* correspondente

```
try {
  // código que pode lançar exceções
} catch (ExcecaoA a) {
  // código que trata a exceção ExcecaoA
} /* ... */ {
  // ...
} catch (ExcecaoN n) {
  // código que trata a exceção ExcecaoA
}
```

Multi-catch

- É comum que um bloco *try* seja seguido por vários blocos *catch* para tratar vários tipos de exceção
- Se os corpos dos vários blocos *catch* forem idênticos, use o recurso *multi-catch*
    - *catch (Tipo1 | Tipo2 | Tipo3 e)*

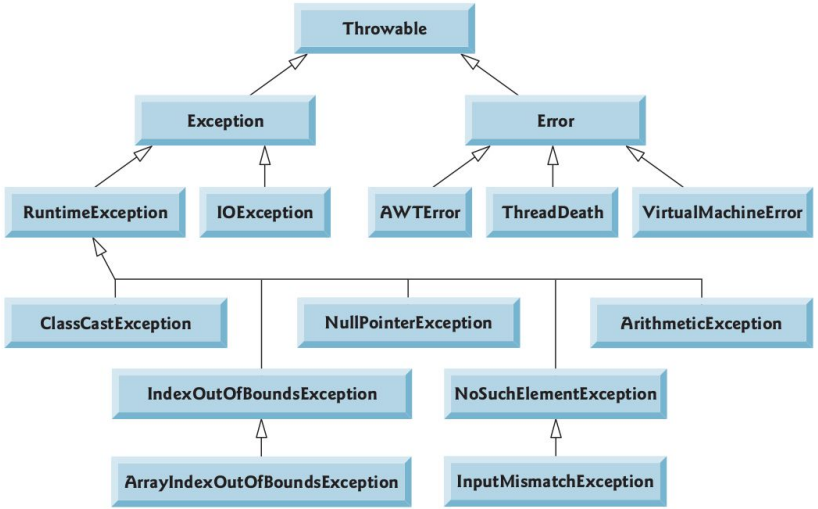
Bloco *finally*

- É executado independente de uma exceção ser lançada ou não
- Se estiver presente, ele é colocado depois do último bloco *catch*

```
try {
  // código que pode lançar exceções
} catch (ExcecaoA a) {
  // código que trata a exceção ExcecaoA
} /* ... */ {
  // ...
} finally {
  // código que será executado
  //independente de haver exceção
}
```

Hierarquia de exceção Java

- Todas as classes de exceção do Java herdam direta ou indiretamente da classe *Exception*
- Somente objetos *Throwable* podem ser utilizados com o mecanismo de tratamento de exceção
- *Exception*s representam situações excepcionais que podem ser capturadas pelo aplicativo
- A classe *Error* e suas subclasses representam situações anormais
    - geralmente os aplicativos não podem se recuperar de *Error*s



Prática

- Escreva um programa que utiliza um *array* para representar as respostas de um aluno a um questionário com 10 perguntas
- O questionário possui as alternativas 0 a 5 para cada questão
- Seu programa deve contar o total de ocorrências de cada uma das respostas
- O que acontece se usuário digitar a alternativa 14 para uma das questões?
- **Vamos adicionar o tratamento de exceção?**

Prática

- Escreva um método que receba um numerador e um denominador inteiros e retorno o resultado da divisão inteira
- No método *main*, leia do usuário dois inteiros e utilize o método recém criado para calcular a divisão
- Não vamos fazer tratamento de exceções ainda
- O que acontece quando:
  - o denominador inserido é um inteiro diferente de zero?
  - o denominador inserido é um inteiro igual a zero?
  - o denominador inserido é uma palavra?

Rastreamento de pilha

- Informações exibidas quando uma exceção é lançada
- Inclui:
  - o nome da exceção (*java.lang.ArithmeticException*) em uma mensagem descritiva que indica o problema que ocorreu
  - a pilha de chamadas de método no momento em que a exceção ocorreu
- Ajuda a depurar o programa

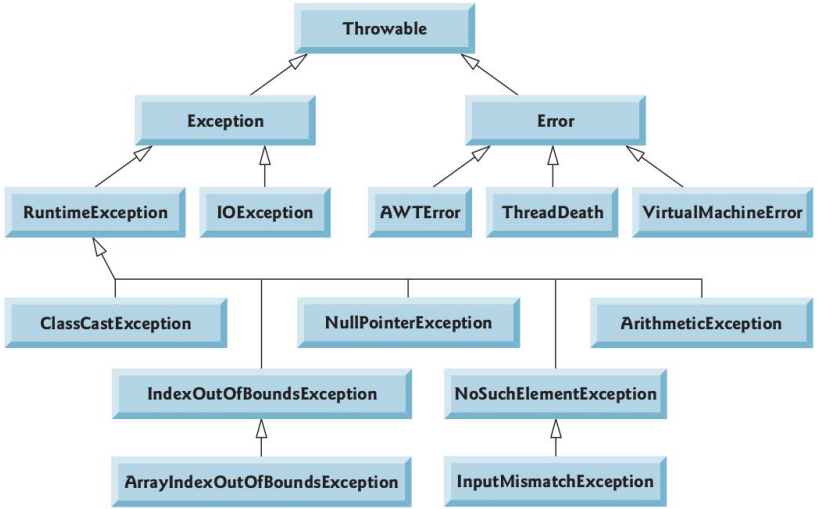
```
Exception in thread "main"
java.lang.ArithmeticException: / by zero

at DivideByZeroNoExceptionHandling.quotient(
DivideByZeroNoExceptionHandling.java:10)

at DivideByZeroNoExceptionHandling.main(
DivideByZeroNoExceptionHandling.java:22)
```

Prática

- Agora sim! Vamos implementar o tratamento de exceções no exemplo anterior?

Exceções verificadas e não verificadas

- Subclasses da classe *RuntimeException* são exceções não verificadas
- Classes herdeiras de *Exception*, mas não de *RuntimeException*, são consideradas exceções verificadas
- O compilador verifica cada chamada de método e declaração de método para determinar se ele lança uma exceção verificada
    - Se sim, o compilador checa se a exceção verificada é capturada ou é declarada em uma cláusula *throws*



Cláusula *throws*

- Especifica as exceções que o método pode lançar se ocorrerem problemas
- Deve aparecer após a lista de parâmetros e antes do corpo do método
  - *public static int quotient(int numerator, int denominator) throws ArithmeticException*
- A exceção pode ser lançada em um método:
  - pela chamada de um método que lance uma exceção;
  - pela realização de uma operação que gere uma exceção;
  - explicitamente:
    - *throw new Exception();*

```
public class UsoFinallyThrow {
 public static void main(String[] args) {
  try {
   lancaExcecao();
  } catch (Exception excecao) {
   System.err.println("Excecao tratada na main");
  }
  naoLancaExcecao();
 }

 public static void lancaExcecao() throws Exception {
  try {
   System.out.println("Metodo lancaExcecao");
   throw new Exception();
  } catch (Exception excecao) {
   System.err.println("Excecao tratada no metodo lancaExcecao");
   throw excecao;
  } finally {
   System.err.println("Finally executado em lancaExcecao");
  }
 }

 public static void naoLancaExcecao() {
  try {
   System.out.println("Metodo naoLancaExcecao");
  } catch (Exception exception) {
   System.err.println(exception);
  } finally {
   System.err.println("Finally executado em naoLancaExcecao");
  }
  System.out.println("Fim do metodo naoLancaExcecao");
 }
}
```

Os códigos relacionados a esta aula estão disponíveis em

https://github.com/italoaug/Programacao-Orientada-a-Objetos/blob/main/codigos/excecoes

Referências

DEITEL, Paul J; DEITEL, Harvey M. **Java: como programar**. 8.ed. São Paulo: Pearson Education do Brasil, 2010. xxix, 1144 p. ISBN: 9788576055631.