fusberti@ic.unicamp.br

Tratamento de Exceções

prof. Fábio Luiz Usberti

MC322 - Programação Orientada a Objetos

Instituto de Computação - UNICAMP - 2014

Sumário

Introdução

Exceções

- Uma exceção é uma indicação que um evento anormal ocorreu durante a execução de um aplicativo.
- A linguagem Java oferece meios para realizar tratamentos de exceções.
- O tratamento de exceções permite que programas sejam mais tolerantes à falha.
- Alguns problemas de execução podem ser tratados a ponto de manter a execução do programa ou terminar de forma não abrupta a execução.

Exceções

Método de divisão inteira

- Será demonstrando um exemplo de um aplicativo que realiza a divisão de dois números inteiros, sem nenhum tratamento de exceções.
- Esse aplicativo recebe do terminal dois números inteiros e retorna o quociente inteiro resultante.
- Duas exceções do aplicativo que não são tratadas são:
  - Divisão por zero: uma exceção do tipo `ArithmeticException` é lançada pelo operador de divisão (`/`) quando o denominador é zero.
  - Entrada inválida de dados: uma exceção do tipo `InputMismatchException` é lançada pelo método `nextInt` (classe `Scanner`) quando o valor lido não é um número inteiro.

Exceções

```
// IntegerDivision.java
// A classe IntegerDivision testa a divisão inteira de dois números
import java.util.Scanner;

public class IntegerDivision {
    // método que retorna o quociente da divisão
    public static int quotient(int numerator, int denominator) {
        return numerator / denominator; // não trata divisão por zero
    } // fim método quotient

    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Entre com o numerador: ");
        int numerator = scanner.nextInt();
        System.out.print("Entre com o denominador: ");
        int denominator = scanner.nextInt();

        int result = quotient(numerator, denominator);
        System.out.printf("\nResultado: %d / %d = %d\n", numerator, denominator, result);
    } // fim main
} // fim classe IntegerDivision
```

Exceções

```
Entre com o numerador: 5
Entre com o denominador: 2

Resultado: 5 / 2 = 2
```

```
Entre com o numerador: 5
Entre com o denominador: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
at IntegerDivision.quotient(IntegerDivision.java:8)
at IntegerDivision.main(IntegerDivision.java:19)
```

```
Entre com o numerador: 5
Entre com o denominador: hello
Exception in thread "main" java.util.InputMismatchException
at java.util.Scanner.throwFor(Scanner.java:857)
at java.util.Scanner.next(Scanner.java:1478)
at java.util.Scanner.nextInt(Scanner.java:2108)
at java.util.Scanner.nextInt(Scanner.java:2067)
at IntegerDivision.main(IntegerDivision.java:17)
```

Tratamento de exceções

Exceções `ArithmeticExceptions` e `InputMismatchExceptions`

- O aplicativo de divisão inteira de números foi reescrito para tratar as exceções causadas por divisão por zero e entrada inválida de dados.
- Nesta nova versão, se o usuário entrar com um dado inválido, será dada a oportunidade dele entrar novamente com um dado.
- Os tipos de exceções `ArithmeticExceptions` e `InputMismatchExceptions` são classes que estendem da classe `Exception`.
- Para o tratamento dessas exceções, elas precisam estar visíveis para a classe que está sendo definida.
- A exceção `InputMismatchExceptions` deve ser importada do pacote `java.util`.
- A exceção `ArithmeticExceptions` não precisa ser importada, pois pertence ao pacote `java.lang`.

Tratamento de exceções

Bloco `try`

- Em Java, um bloco `try` consiste na palavra-chave `try` seguido de um conjunto de instruções delimitadas por chaves `{ }`.
- A função do `try` é de indicar ao compilador que as instruções dentro do bloco eventualmente podem lançar uma exceção.
- Na ocorrência de uma exceção dentro de um bloco `try`, o código remanescente deixa de ser executado.

Tratamento de exceções

Bloco `catch`

- Um bloco `try` pode ser seguido de mais de um bloco `catch`, que também é denominado de tratador de exceção.
- No exemplo, dois blocos `catch` tratam as exceções `ArithmeticExceptions` e `InputMismatchExceptions`.
- Cada tratador de exceção deve definir um parâmetro (em parênteses) cujo tipo (classe) corresponde à exceção a ser tratada.
- O nome do parâmetro, se possível, deve estar associado a seu tipo, facilitando a legibilidade do código.

Tratamento de exceções

Bloco `catch`

- Na ocorrência de uma exceção, o bloco `catch` executado é o primeiro cujo tipo definido corresponde à exceção lançada.
- Nesse caso, o parâmetro recebe a referência do objeto do tipo da exceção.
- O bloco `catch` poderá interagir com o objeto da exceção permitindo, por exemplo, a impressão das informações pertinentes do erro.
- Por convenção, a impressão de informações de exceções ocorre na saída de erro padrão `System.err`.
- Depois de executar um bloco `catch`, as variáveis locais (incluindo a referência ao objeto da exceção), definidas dentro do bloco, saem do escopo e são liberadas da memória. Além disso, o fluxo de execução é levado para a primeira instrução que sucede o último bloco `catch`.

Exemplo de tratamento de exceções

```
// IntegerDivision2.java
// A classe IntegerDivision2 testa a divisão inteira de dois números, tratando possíveis exceções
import java.util.InputMismatchException;
import java.util.Scanner;

public class IntegerDivision2 {

    // método que retorna o quociente da divisão
    public static int quotient(int numerator, int denominator) throws ArithmeticException {
        return numerator / denominator;
    } // fim método quotient

    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        boolean continueLoop = true; // repete enquando as entradas forem inválidas

    /* continua na próxima página */
```

Exemplo de tratamento de exceções

```
    /* continua da página anterior */

        do {
            try {
                System.out.print("Entre com o numerador: ");
                int numerator = scanner.nextInt();
                System.out.print("Entre com o denominador: ");
                int denominator = scanner.nextInt();

                int result = quotient(numerator, denominator);
                System.out.printf("\nResultado: %d / %d = %d\n", numerator, denominator, result);

                continueLoop = false; // entrada correta; laço pode encerrar

            } catch (InputMismatchException inputMismatchException) {
                System.err.println("\nExceção: " + inputMismatchException);
                scanner.nextLine(); // descarte a entrada para que o usuário tente novamente
                System.out.println("É necessário entrar dois números inteiros. Por favor, tente novamente.\n");

            } catch (ArithmeticException arithmeticException) {
                System.err.println("\nExceção: " + arithmeticException);
                System.out.println("Zero não é um denominador válido. Por favor, tente novamente.\n");
            } // fim catch

        } while (continueLoop); // fim do ... while

    } // fim main
} // fim classe IntegerDivision2
```

Exemplo de tratamento de exceções

```
Entre com o numerador: 5
Entre com o denominador: 0

Exceção: java.lang.ArithmeticException: / by zero
Zero não é um denominador válido. Por favor, tente novamente.

Entre com o numerador: 5
Entre com o denominador: hello

Exceção: java.util.InputMismatchException
É necessário entrar dois números inteiros. Por favor, tente novamente.

Entre com o numerador: 5
Entre com o denominador: 2

Resultado: 5 / 2 = 2
```

Tratamento de exceções

Exceções não tratadas

- Uma exceção não tratada é aquela que não foi capturada por nenhum bloco `catch` definido pelo programador.
- Em geral, exceções não tratadas provocam a interrupção abruta do aplicativo.
- Se um aplicativo possui múltiplos threads, uma exceção não tratada provocará somente a interrupção do seu thread correspondente.
- Os vários threads de um aplicativo normalmente dependem um do outro para a consolidação de alguma informação. Desse modo, exceções não tratadas podem provocar comportamentos adversos em um aplicativo com essas características.

Tratamento de exceções

Palavra-chave `throws`

- A palavra-chave `throws` pode ser utilizada em um método para indicar os tipos de exceções lançadas pelo método.
- As exceções podem ser lançadas por instruções do próprio método ou por outros métodos que são chamados internamente.
- No exemplo, o método `quotient` foi declarado com a palavra-chave `throws` para que os clientes da classe saibam que esse método eventualmente pode lançar uma exceção `ArithmeticException`.
- Antes de usar algum método de API, leia a documentação verificando que tipos de exceções são lançadas e quais são as fontes causadoras dessas exceções. Isso possibilitará tratar adequadamente as exceções na classe sendo implementada, aumentando sua tolerância a falhas.

Referências

1. Java: Como Programar, Paul Deitel & Heivey Deitel; Pearson; 7a. Ed. (no. chamada IMECC – 05.133 D368j)
2. Data Structures and Algorithms with Object Oriented Design Patterns in Java, Bruno Preiss; (http://www.brpreiss.com/books/opus6/)
3. The Java Tutorials (Oracle) (http://docs.oracle.com/javase/tutorial/)
4. Guia do Usuário UML, Grady Booch et. al.; Campus(1999)
5. Java Pocket Guide - Robert Liguori & Patricia Liguori; O'Reilley, 2008.