PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
TÉCNICAS DE PROGRAMAÇÃO 1 – CMP 1046
PROF. MSC. ANIBAL SANTOS JUKEMURA

[Tratamento de Exceções]

Agenda:

- Tratamento de Exceção
- Blocos try/cacth
- Multi-catch
- Exemplos
- Bloco finally

PUC GOIÁS

Tratamento de Exceção

- Um bloco **try** inclui o código que pode lançar (**throw**) uma exceção e o código que não deve ser executado se ocorrer uma exceção (isto é, se ocorrer uma exceção, o código restante no bloco try será pulado).

- Um bloco try consiste na palavra-chave **try** seguida por um bloco de código entre chaves ({ }).

- Observação: o termo “bloco try” às vezes só se refere ao bloco de código que segue a palavra-chave try (não incluindo a própria palavra-chave try). Para simplificar, utilizamos o termo “bloco try” para nos referirmos ao bloco de código que se segue à palavra-chave try, bem como a palavra-chave try.

PUC GOIÁS

Tratamento de Exceção

- Um bloco **catch** (também chamado de cláusula catch ou rotina de tratamento de exceção) captura (isto é, recebe) e **trata uma exceção**.

- Um bloco **catch** inicia com a palavra-chave **catch seguido por um parâmetro entre parênteses** (chamado **parâmetro de exceção**) e um bloco de código entre chaves.

- Observação: o termo “cláusula catch” é às vezes utilizado para referir-se à palavra-chave catch seguida por um bloco de código, em que o termo “bloco catch” refere-se apenas ao bloco de código que se segue à palavra-chave catch, mas que não a inclui. Para simplificar, utilizamos o termo “bloco catch” para nos referirmos ao bloco de código que se segue à palavra-chave catch, bem como à própria palavra-chave.

PUC GOIÁS

Tratamento de Exceção

- **Pelo menos um bloco catch** ou um **bloco finally** deve se seguir imediatamente ao bloco try.

- Cada bloco catch especifica entre parênteses um parâmetro de exceção que identifica o tipo de exceção que a rotina de tratamento pode processar.

- Quando ocorrer uma exceção em um bloco try, o bloco catch que é executado é o primeiro cujo tipo corresponde ao tipo da exceção que ocorreu (isto é, o tipo no bloco catch corresponde exatamente ao tipo de exceção lançado ou é uma superclasse direta ou indireta dele).

PUC GOIÁS

Tratamento de Exceção

- O nome do parâmetro de exceção permite ao bloco catch interagir com um objeto de exceção. Por padrão, os métodos de impressão de **System.err**.

- Se **ocorrer uma exceção** em um bloco try, **o bloco try termina imediatamente** e o controle do programa **é transferido para o primeiro dos blocos catch** seguintes em que o tipo do parâmetro de exceção corresponde ao tipo da exceção lançada.

- Depois que a exceção é tratada, o controle do programa não retorna ao ponto de lançamento, **porque o bloco try expirou** (e suas variáveis locais foram perdidas). Em vez disso, o controle retoma depois do último bloco catch.

- Isso é conhecido como o **modelo de terminação do tratamento de exceção**.

PUC GOIÁS

Tratamento de Exceção – Exemplo 1

```
boolean continueLoop = true; // determina se mais entradas são necessárias
do
{
   try // lê dois números e calcula o quociente
   {
      System.out.print("Digite um nomerador inteiro: ");
      int numerador = scanner.nextInt();
      System.out.print("Digite um denominador inteiro: ");
      int denominador = scanner.nextInt();
      int resultado = quociente(numerador, denominador);
      System.out.printf("%nResultado: %d / %d = %d%n", numerador, denominador, resultado);
      continueLoop = false; // entrada bem-sucedida; fim do loop
   }
   catch (InputMismatchException inputMismatchException)
   {
      //System.err.printf("%Exceção: %s%n", inputMismatchException);
      System.err.printf("Excecao: %s\n", inputMismatchException);
      scanner.next(); // descarta entrada para o usuário tentar de novo
      System.out.printf( "Digite somente inteiros. Tente novamente.\n\n");
   }

} while (continueLoop);
```

PUC GOIÁS

Tratamento de Exceção – multi catch

- É relativamente 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 (introduzido no Java SE 7) para capturar esses tipos de exceção em uma única rotina de tratamento catch e realizar a mesma tarefa.

- A sintaxe para um multi-catch é:

```
catch (Tipo1 | Tipo2 | Tipo3 e)
```

PUC GOIÁS

Tratamento de Exceção – multi catch – Exemplo 2

```
boolean continueLoop = true; // determina se mais entradas são necessárias
do
{
   try // lê dois números e calcula o quociente
   {
      System.out.print("Digite um numerador inteiro: ");
      int numerador = scanner.nextInt();
      System.out.print("Digite um denominador inteiro: ");
      int denominador = scanner.nextInt();
      int resultado = quociente(numerador, denominador);
      System.out.printf("%nResultado: %d / %d = %d%n", numerador, denominador, resultado);
      continueLoop = false; // entrada bem-sucedida; fim do loop
   }
   catch (InputMismatchException inputMismatchException)
   {
      System.err.printf("Excecao: %s\n", inputMismatchException);
      scanner.next(); // descarta entrada para o usuário tentar de novo
      System.out.printf("Digite somente inteiros. Tente novamente.\n\n");
   }
   catch (ArithmeticException arithmeticException)
   {
      System.err.printf("Exceção: %s\n", arithmeticException);
      System.out.printf("Divisão por ZERO. Tente novamente.\n\n");
   }
} while (continueLoop);
```

PUC GOIÁS

Tratamento de Exceção

- Hierarquia de Exceção JAVA

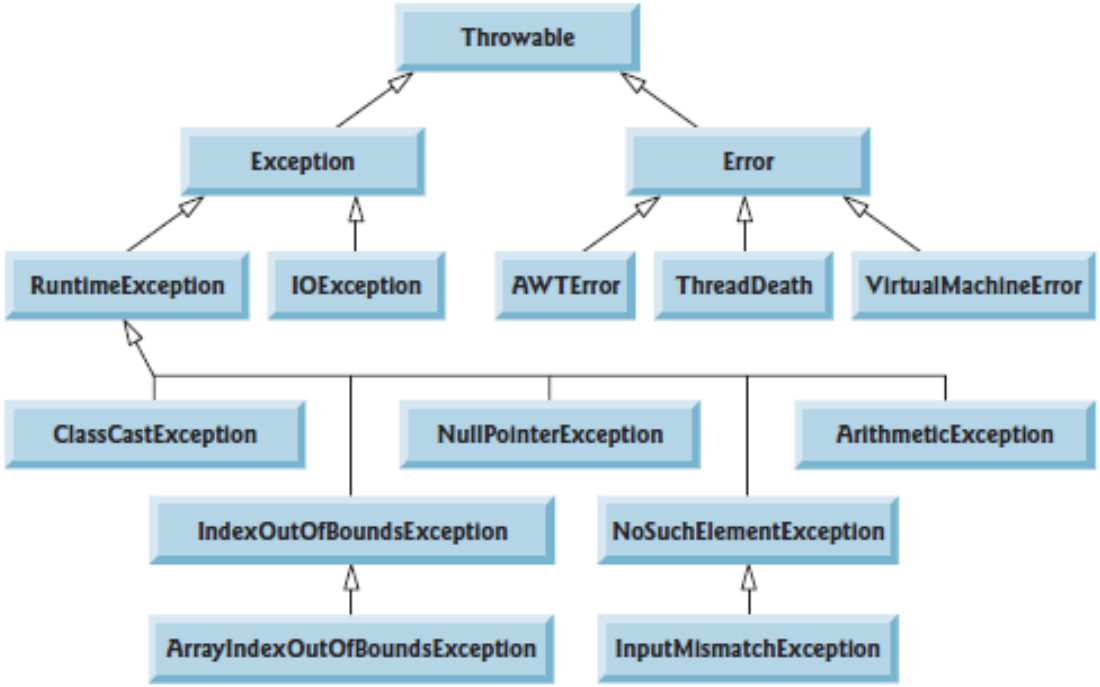

Figura 11.4 | Parte da hierarquia de herança da classe `Throwable`.

PUC GOIÁS

Tratamento de Exceção – Bloco finally

- Programas que obtêm certos recursos devem retorná-los ao sistema para evitar os assim chamados vazamentos de recurso.

- Em linguagens de programação como C e C++, o tipo mais comum de vazamento de recurso é um vazamento de memória. O Java realiza coleta automática de lixo de memória não mais utilizada por programas, evitando assim a maioria dos vazamentos de memória.

- Entretanto, outros tipos de vazamentos de recurso podem ocorrer. Por exemplo, arquivos, conexões de banco de dados e conexões de rede que não são fechadas adequadamente depois que não são mais necessárias talvez não estejam disponíveis para uso em outros programas.

PUC GOIÁS

Tratamento de Exceção – Bloco finally

- Se uma exceção que ocorre em um bloco try não puder ser capturada por rotinas de tratamento catch desse bloco try, o programa pula o restante do bloco try e o controle prossegue para o bloco finally. Então, o programa passa a exceção para o próximo bloco try externo — normalmente no método chamador — onde um bloco catch associado pode capturá-lo. Esse processo pode ocorrer pelos muitos níveis de blocos try. Além disso, a exceção talvez não seja capturada .

- **Se um bloco catch lançar uma exceção, o bloco finally ainda executará**. Então, a exceção é passada para o próximo bloco try externo — novamente, em geral no método chamador.

- **Como um bloco finally sempre é executado**, ele normalmente contém código de liberação do recurso.

PUC GOIÁS

Tratamento de Exceção – Bloco finally (com captura de exceção)

```
public class ExemploFinally1 {

public static void throwException() throws Exception
{
    try // lança uma exceção e imediatamente a captura
    {
        System.out.println("Metodo throwException");
        throw new Exception(); // gera a exceçãoÿ
    }
    catch (Exception exception) // captura exceção lançada em try
    {
        System.err.println( "Exceção.");
    }

}
    public static void main(String[] args) {
        try
        {
          throwException();
        }
        catch (Exception exception) // exceção lançada por throwException
        {
            System.err.println("Exception handled in main");
        }
        finally // executa independentemente do que ocorre em try...catchÿ
        {
            System.err.println("Finally ainda executado por causa da exceção");
        }

    }
}
```

PUC GOIÁS

Tratamento de Exceção – Bloco finally (sem captura de exceção)

```
public class ExemploFinally1 {

public static void throwException() throws Exception
{
    try // lança uma exceção e imediatamente a captura
    {
       System.out.println("Metodo throwException");
       //throw new Exception(); // gera a exceçãoÿ
    }
    catch (Exception exception) // captura exceção lançada em try
    {
       System.err.println( "Exceção.");
    }

}
    public static void main(String[] args) {
       try
       {
         throwException();
       }
       catch (Exception exception) // exceção lançada por throwException
       {
           System.err.println("Exception handled in main");
       }
       finally // executa independentemente do que ocorre em try...catchÿ
       {
          //System.err.println("Finally ainda executado após exceção");
          System.err.println("Finally ainda executado mesmo sem exceção");
       }

    }
}
```

PUC GOIÁS

Tratamento de Exceção – Exemplo de throw

```
import java.lang.IllegalArgumentException;

public class Exemplo3 {

     public static void saque(double valor) {
        if(valor > 400) {
            System.out.println("Esse programa gerou um erro:");
            IllegalArgumentException erro = new IllegalArgumentException();
            throw erro;
        }else {
            System.out.println("Valor retirado da conta: R$"+valor);
        }
    }


    public static void main(String[] args) {
       // saque(200);
        saque(1500);
    }

}
```

PUC GOIÁS

Tratamento de Exceção – Exemplo de getMessage()

```
public class Exemplo4 {



    public static void main(String[] args) {
        try {
            int[] numero = new int[5];

            for(int i = 0; i <= 10; i++){
                numero[i] = i;
                System.out.println(i);
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array fora do índice: "+e.getMessage());
        }
    }
}
```

PUC GOIÁS

BOAS PRÁTICAS DE PROGRAMAÇÃO

Observação de engenharia de software 11.1

As exceções emergem pelo código explicitamente mencionado em um bloco `try`*, por chamadas de método profundamente aninhadas iniciado pelo código em um bloco* `try` *ou a partir da Java Virtual Machine à medida que ela executa os bytecodes do Java.*

Erro comum de programação 11.1

É um erro de sintaxe colocar código entre um bloco `try` *e seus blocos* `catch` *correspondentes.*

Dica de prevenção de erro 11.1

Leia a documentação on-line da API para obter informações sobre um método antes de utilizá-lo em um programa. A documentação especifica as exceções lançadas pelo método (se houver alguma) e indica as razões pelas quais tais exceções podem ocorrer. Em seguida, leia na documentação da API on-line as classes de exceção especificadas. A documentação para uma classe de exceção normalmente contém razões potenciais por que essas exceções ocorrem. Por fim, forneça o tratamento para essas exceções em seu programa.

Observação de engenharia de software 11.2

Incorpore seu tratamento de exceção e a estratégia de recuperação de erro a seu sistema desde o início do processo de projeto — incluí-las depois que um sistema foi implementado pode ser difícil.

PUC GOIÁS

BOAS PRÁTICAS DE PROGRAMAÇÃO

Observação de engenharia de software 11.3

O tratamento de exceção fornece uma técnica única e uniforme para documentar, detectar e recuperar-se de erros. Isso ajuda os programadores que trabalham em grandes projetos a entender o código de processamento de erro uns dos outros.

Dica de prevenção de erro 11.5

O bloco `finally` *é um lugar ideal para liberar os recursos adquiridos em um bloco* `try` *(como arquivos abertos), o que ajuda a eliminar vazamentos de recurso.*

Dica de desempenho 11.1

Sempre libere um recurso explicitamente e logo que ele não for mais necessário. Isso faz com que os recursos disponíveis possam ser reutilizados o mais rápido possível, melhorando assim a utilização dos recursos e o desempenho do programa.

PUC GOIÁS

BOAS PRÁTICAS DE PROGRAMAÇÃO

Erro comum de programação 11.4

Se uma exceção não tiver sido capturada quando o controle entrar em um bloco `finally` e esse bloco lançar uma exceção que não será capturada por ele, a primeira exceção será perdida e a exceção do bloco será retornada ao método chamador.

Dica de prevenção de erro 11.6

Evite inserir em um bloco `finally` código que pode usar `throw` para lançar uma exceção. Se esse código for necessário, inclua o código em um `try...catch` dentro do bloco `finally`.

Erro comum de programação 11.5

Supor que uma exceção lançada de um bloco `catch` será processada por esse bloco `catch` ou qualquer outro bloco `catch` associado com a mesma instrução `try` pode resultar em erros de lógica.

Boa prática de programação 11.1

O tratamento de exceção remove o código de processamento de erro da linha principal do código de um programa para melhorar a clareza do programa. Não coloque `try...catch...finally` em torno de toda instrução que possa lançar uma exceção. Isso diminui a legibilidade. Em vez disso, coloque um bloco `try` em torno de uma parte significativa do código. Esse bloco `try` deve ser seguido por blocos `catch` que tratam cada possível exceção e os blocos `catch` devem ser seguidos por um único bloco `finally` (se algum for necessário).

Referência Bibliográfica Principal

• DEVMEDIA. Disponível em https://www.devmedia.com.br. Acessado em Julho de 2019.

• DEITEL, Harvey M. Java: Como Programar – 10 ed. Cap 11. 2015.