Tratamento de Exceções

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

Herculano De Biasi
herculano.debiasi@unoesc.edu.br

Tópicos

- Motivação
- Erros
- Erros de compilação
- Exceções
- `Errors` e `Exceptions`
- Estrutura `try-catch-finally`
- Pilha de chamada de métodos (*stack trace*)

Motivação

- Código que gera várias exceções

modulo2 - Problemas.java

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;

public class Problemas {
    public static void main(String[] args) {
        int[] arranjo = {1, 2, 3};
        System.out.println(arranjo[2]);
        System.out.println(arranjo[3]);     // Gera a exceção 'ArrayIndexOutOfBounds'

        List<Integer> numeros = new ArrayList<>();
        numeros.add(10);
        numeros.add(20);
        numeros.add(30);
        System.out.println(numeros.get(2));
        System.out.println(numeros.get(3));  // Lança a exceção 'IndexOutOfBounds'

        Locale.setDefault(Locale.US);
        Scanner ler = new Scanner(System.in);
        System.out.print("Digite um número com parte decimal: ");
        double numero = ler.nextDouble();    // Dependendo do Locale dará problema com , ou .
                                             // gerando a exceção InputMismatchException
        System.out.println(numero);

        numero = Double.parseDouble("10,5"); // Levanta a exceção NumberFormatException

        System.out.println(10/0);            // Gera a exceção ArithmeticException

        String frase = null;
        System.out.println(frase.toUpperCase());  // Lança a exceção NullPointerException
    }
}
```

UNOESC

Erros

- Tipos de erros
  - Erros em tempo de compilação
  - Erros em tempo de execução (exceções de *runtime*)

Erros de Compilação

- Erros em tempo de compilação
  - Normalmente detectados pelo compilador/IDE
  - Programa não irá executar a partir da linha onde o erro foi detectado

modulo2 - ProgramaComErro.java

```
1 }public class ProgramaComErro {
2
3     public static void main(String[] args) {
4         System.out.println("Deu boa!");
5         System.out.println("Até aqui tudo bem...");
6         System.out.printlm();
7     }
8
9 }
```

EXCEÇÕES

- Exceção: Qualquer condição de erro ou comportamento inesperado encontrado por um programa em execução
- Outra definição de exceção é qualquer situação que modifique ou interrompa o fluxo normal de execução de um programa
- Uma exceção pode estar relacionada com alguma condição externa que impede o programa de funcionar corretamente, como por exemplo falta de conexão de rede
- Quando uma exceção acontece, é dito que ela é lançada, levantada ou gerada

EXCEÇÕES

- Causas internas comuns de exceção
  - Tentar manipular um objeto que está com o valor nulo
  - Dividir um número inteiro por zero
  - Tentar manipular um tipo de dado como se fosse outro
  - Tentar utilizar um método ou classe não que não existe
  - Argumentos inválidos na chamada de um método
  - Índice fora de faixa em uma lista ou *array*
- Causas externas comuns de exceção
  - Tentar escrever algo em um arquivo sobre o qual não se tem permissão de escrita
  - Tentar fazer consulta a um banco de dados que não está disponível
  - Tentar conectar em servidor inexistente
  - Tentar abrir um arquivo que não existe
  - Problemas de acesso à rede

UNOESC

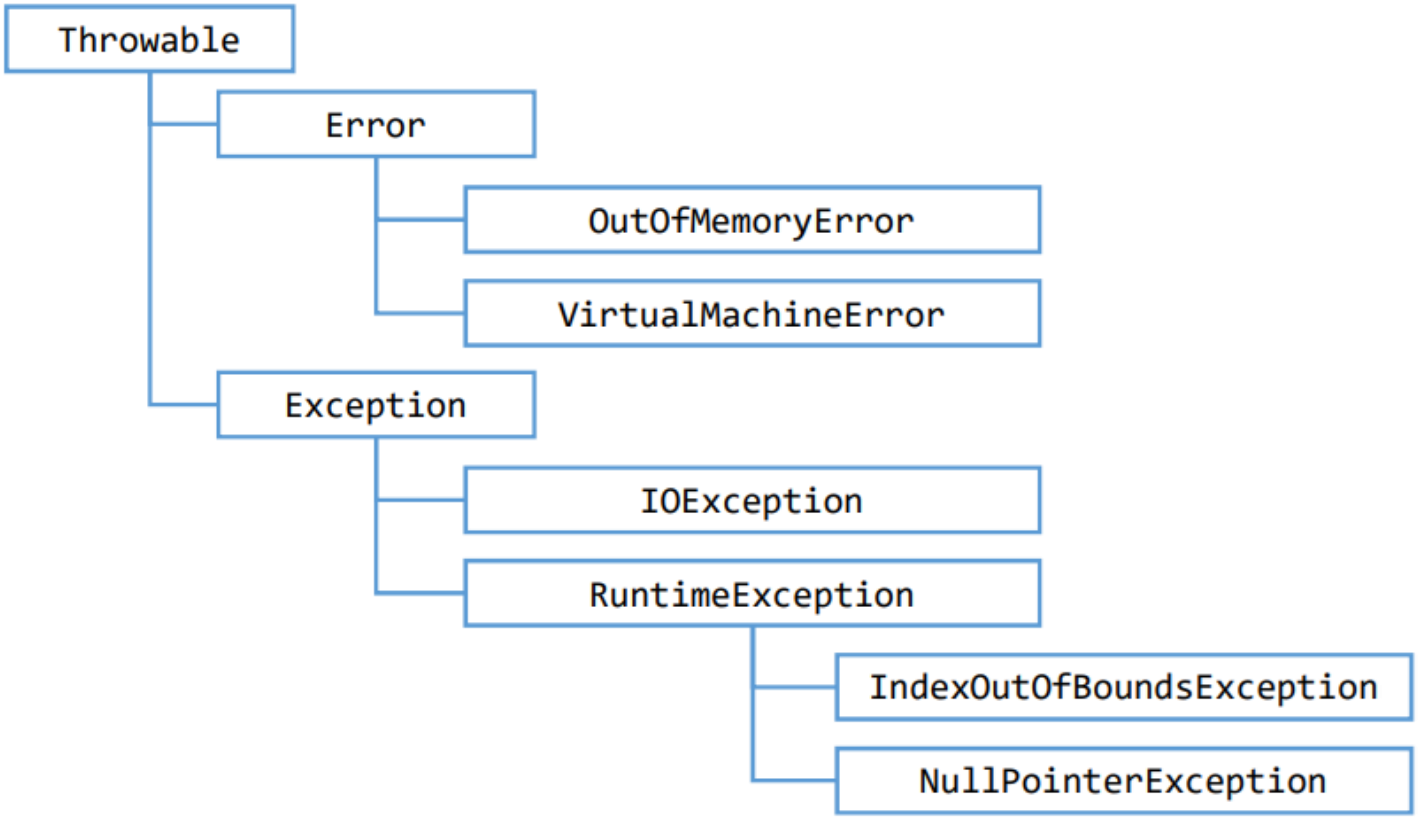
EXCEÇÕES

- A forma como o Java lida com exceções permite um tratamento mais estruturado (de forma hierárquica), organizado, consistente e ao mesmo flexível, melhorando a legibilidade e manutenção dos códigos

- Vantagens
  - Organização: Separa o código de tratamento dos erros do restante das linhas do programa, ou seja, concentra esse tratamento em um lugar específico em vez de espalhá-lo por todo o código
  - O tratamento do erro é delegado para a classe que conhece as regras de negócio
  - Consegue reportar um erro ocorrido entre os métodos chamados anteriormente até o momento do surgimento do erro
  - Agrupa os erros em categorias
  - Permite criar tratamentos personalizados

Errors e Exceptions

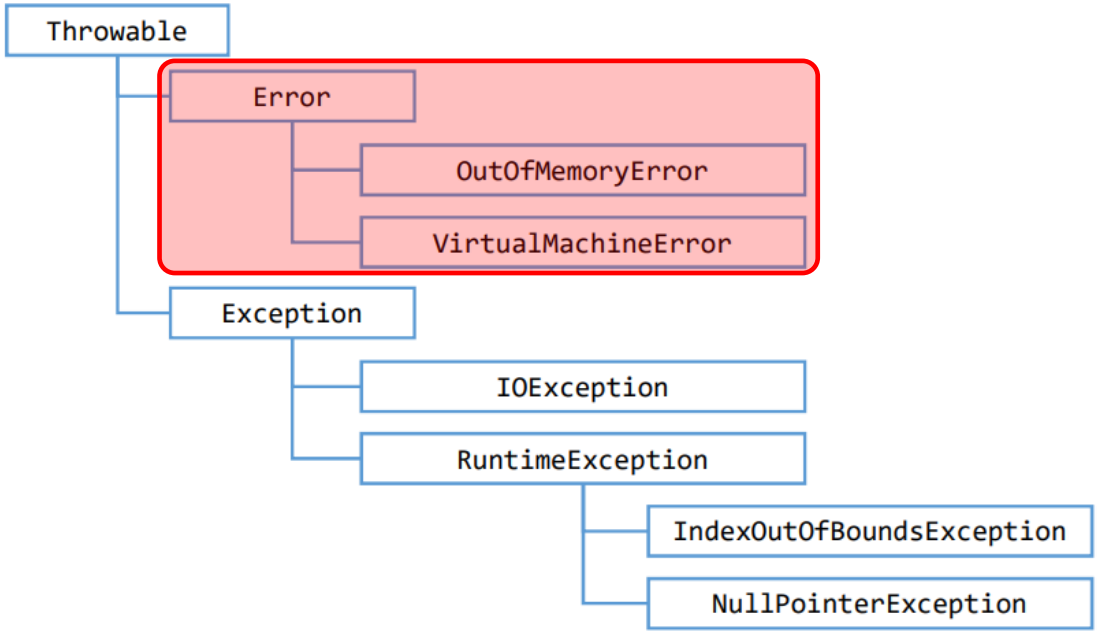
- Hierarquia de exceções: `Throwable` é a superclasse dos erros e exceções do Java



Errors e Exceptions

- Classe `Error`: Situação mais séria e crítica originada por situações das quais o sistema não pode se recuperar, ou seja, são erros fatais, incontornáveis
  - `OutOfMemoryError`: Falta de memória na aplicação
  - `VirtualMachineError`: Erro interno na máquina virtual

- No STS, pressione a combinação $Ctrl + Shift + T$ e pesquise pelas classes abaixo
  - `Error`
  - `OutOfMemoryError`

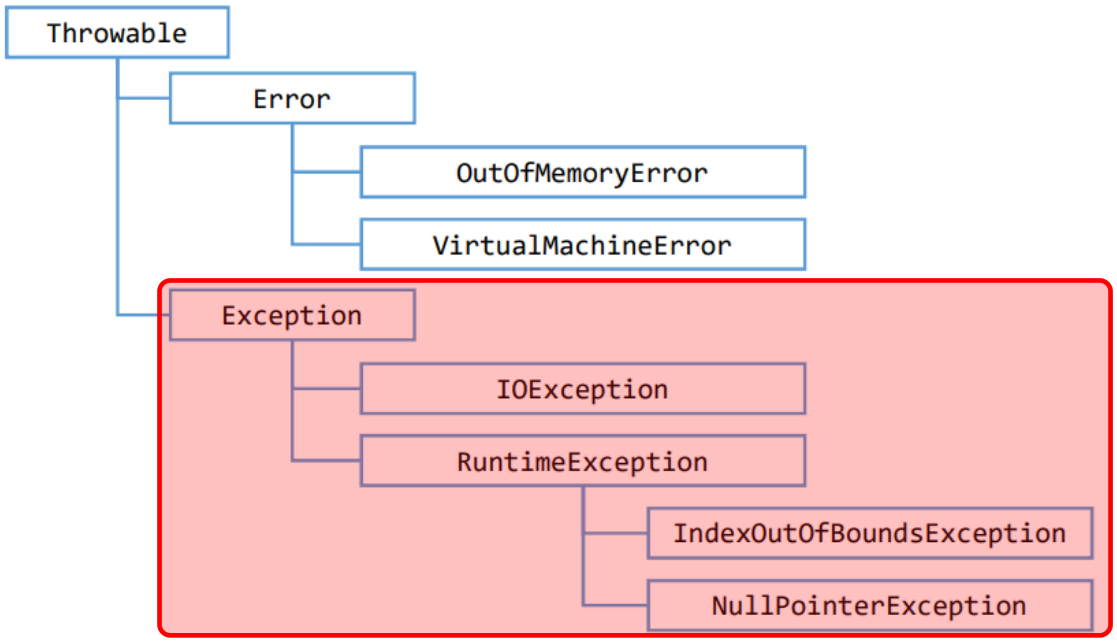


UNOESC

Errors e Exceptions

- Classe `Exception`: Superclasse das exceções, o Java entende que é possível tratá-las e o programa se recuperar da situação anormal
  - Subclasses
    - `IOException`: Erros de entrada e saída, como problema de leitura de dados em disco ou rede; o compilador obriga o programador a tratá-las ou propagá-las
    - `RunTimeException`: O compilador Java NÃO obriga o programador a tratá-las ou propagá-las

- Quando uma exceção é lançada, ela é propagada na pilha de chamadas de métodos em execução, até que seja capturada (tratada) ou o programa seja encerrado



UNOESC

ESTRUTURA TRY–CATCH–FINALLY

- Forma geral

```
try {
   // bloco_de_instrucoes
} catch (subclasse_1 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_2 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_n de Exception e) {
   // bloco_de_instrucoes
} finally {
   // bloco_de_instrucoes
}
```

ESTRUTURA TRY–CATCH–FINALLY

- Bloco `try`
  - A tradução de *try* é tentar
  - Contém o código que representa a execução normal de um trecho de programa que pode causar uma exceção
  - Esse bloco representa um código de risco, por isso é chamado de bloco ou região protegida
  - Caso ocorra algum problema nas instruções dentro do bloco `try`, a execução do programa será desviada para o bloco `catch` apropriado

```
try {
   // bloco_de_instrucoes
} catch (subclasse_1 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_2 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_n de Exception e) {
   // bloco_de_instrucoes
} finally {
   // bloco_de_instrucoes
}
```

ESTRUTURA TRY–CATCH–FINALLY

- Blocos `catch`
  - A tradução de *catch* é capturar, agarrar
  - Contêm o código que permite tratar a exceção caso alguma ocorra
  - Em cada bloco é especificado o tipo da exceção a ser capturada e tratada, assim como um apelido para exceção (no caso o objeto `e`)
  - É possível usar tantos blocos `catch` quanto se julgar necessário para tratar do problema

```
try {
    // bloco_de_instrucoes
} catch (subclasse_1 de Exception e) {
    // bloco_de_instrucoes
} catch (subclasse_2 de Exception e) {
    // bloco_de_instrucoes
} catch (subclasse_n de Exception e) {
    // bloco_de_instrucoes
} finally {
    // bloco_de_instrucoes
}
```

ESTRUTURA TRY–CATCH–FINALLY

- Exemplo: Solução dos problemas 1 e 2

modulo2 - Problemas1e2Tratados.java

```java
import java.util.ArrayList;
import java.util.List;

public class Problemas1e2Tratados {
    public static void main(String[] args) {
        try {
            int[] arranjo = {1, 2, 3};
            System.out.println(arranjo[2]);
            System.out.println(arranjo[3]);     // Gera a exceção 'ArrayIndexOutOfBoundsException'

            List<Integer> numeros = new ArrayList<>();
            numeros.add(10);
            numeros.add(20);
            numeros.add(30);
            System.out.println(numeros.get(2));
            System.out.println(numeros.get(3));  // Lança a exceção 'IndexOutOfBoundsException'
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Posição inválida do array");
            System.out.println(e.getMessage());
        } catch (IndexOutOfBoundsException e) {
            System.out.println("Posição inválida da lista");
        }

        System.out.println("Programa finalizado");
    }
}
```

ESTRUTURA `TRY-CATCH-FINALLY`

- Exercício: Fazer o tratamento das outras 5 exceções que estão sendo geradas no programa do *slide* 3 (motivação)
  - O nome das exceções aparecem nos comentários das respectivas linhas que as geram
  - Cada exceção deve ser tratada em um bloco `catch` separado
  - Use o método `getMessage()` para mostrar o código de erro gerado pelo Java

ESTRUTURA TRY–

- Exemplo

modulo2 - ProblemasTratados.java

```java
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;

public class ProblemasTratados {
    public static void main(String[] args) {
        try {
            int[] arranjo = {1, 2, 3};
            System.out.println(arranjo[2]);
            System.out.println(arranjo[3]);     // Gera a exceção 'ArrayIndexOutOfBoundsException'

            List<Integer> numeros = new ArrayList<>();
            numeros.add(10);
            numeros.add(20);
            numeros.add(30);
            System.out.println(numeros.get(2));
            System.out.println(numeros.get(3));  // Lança a exceção 'IndexOutOfBoundsException'

            Locale.setDefault(Locale.US);
            Scanner ler = new Scanner(System.in);
            System.out.print("Digite um número com parte decimal: ");
            double numero = ler.nextDouble();    // Dependendo do Locale dará problema com , ou .
                                                 // gerando a exceção InputMismatchException
            System.out.println(numero);

            numero = Double.parseDouble("10,5"); // Levanta a exceção NumberFormatException

            System.out.println(10/0);            // Gera a exceção ArithmeticException

            String frase = null;
            System.out.println(frase.toUpperCase());  // Lança a exceção NullPointerException
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Posição inválida do array");
        } catch (IndexOutOfBoundsException e) {
            System.out.println("Posição inválida da lista");
        } catch (InputMismatchException e) {
            System.out.println("Entrada inválida");
        } catch (NumberFormatException e) {
            System.out.println("Formato de número inválido");
        } catch (ArithmeticException e) {
            System.out.println("Erro de divisão por zero");
        } catch (NullPointerException e) {
            System.out.println("Erro de ponteiro nulo");
        }

        System.out.println("Programa finalizado");
    }
}
```

UNOESC

ESTRUTURA TRY–CATCH–FINALLY

- Blocos `catch`
  - A ordem em que os blocos `catch` aparecem é muito importante – eles devem ser dispostos da mais específica (acima) para a mais genérica (abaixo)

modulo2 - bloco try-catch.java

```java
try {
    comando(s);
} catch (Exception e) {
    comando(s);
} catch (FileNotFoundException f) {
    comando(s);
}
```

Errado

modulo2 - bloco try-catch.java

```java
try {
    comando(s);
} catch (FileNotFoundException f) {
    comando(s);
} catch (Exception e) {
    comando(s);
}
```

Certo

ESTRUTURA TRY–CATCH–FINALLY

- Bloco `finally`
  - Esse bloco é opcional, mas quando presente é sempre executado, em outras palavras, ele encerra a execução, tanto para o bloco `try` quanto para o bloco `catch`, mesmo havendo uma instrução `return` neles
  - Só pode haver um bloco `finally` para cada bloco `try`
  - São utilizados geralmente para liberar recursos que tenham sido alocados durante o processamento do bloco `try` e que podem ser liberados, independentemente de a execução ter encerrado com sucesso ou ter sido interrompida por uma condição de exceção, ou seja, é um código de limpeza

```
try {
   // bloco_de_instrucoes
} catch (subclasse_1 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_2 de Exception e) {
   // bloco_de_instrucoes
} catch (subclasse_n de Exception e) {
   // bloco_de_instrucoes
} finally {
   // bloco_de_instrucoes
}
```

ESTRUTURA TRY–CATCH–FINALLY

- Código com problema, não desaloca o *scanner* em caso de erro

modulo2 - ProblemaRecursos.java

```java
import java.util.Scanner;

public class ProblemaRecursos {

    public static void main(String[] args) {
        final int CONSTANTE = 42;
        Scanner ler = new Scanner(System.in);

        try {
            System.out.print("Digite o denominador: ");
            int numero = ler.nextInt();

            int resultado = CONSTANTE / numero;
            System.out.printf("\nResultado de 42 / %d = %d\n\n", numero, resultado);

            System.out.println("Fechando o Scanner...");
            ler.close();
        } catch (Exception e) {
            System.out.println("\n" + e.getMessage());
            System.out.println("Não fechou o Scanner!\n");
        }

        System.out.println("Fim do programa!");
    }

}
```

ESTRUTURA TRY–CATCH–FINALLY

- Correção do problema com o emprego de `finally`

modulo2 - SolucaoFinally.java

```java
import java.util.Scanner;

public class SolucaoFinally {

    public static void main(String[] args) {
        final int CONSTANTE = 42;
        Scanner ler = new Scanner(System.in);

        try {
            System.out.print("Digite o denominador: ");
            int numero = ler.nextInt();

            int resultado = CONSTANTE / numero;
            System.out.printf("\nResultado de 42 / %d = %d\n\n", numero, resultado);
        } catch (Exception e) {
            System.out.println("\n" + e.getMessage() + "\n");
        } finally {
            System.out.println("Fechando o Scanner...");
            ler.close();
        }

        System.out.println("Fim do programa!");
    }

}
```

UNOESC

CENTRO DE INOVAÇÃO VIDEIRA

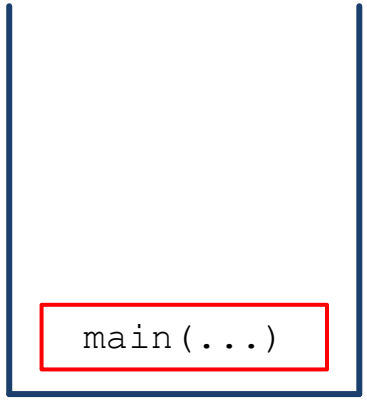
SANTA CATARINA

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`

modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

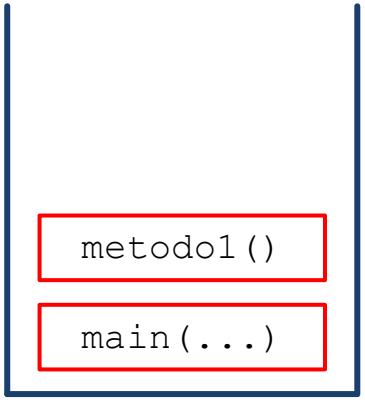


Pilha (*Stack*)

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`



Pilha (*Stack*)

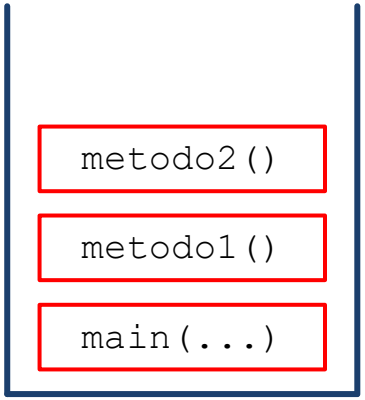
modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`



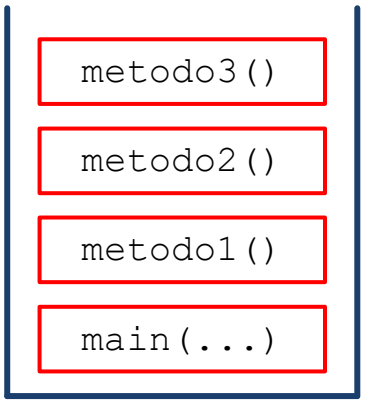
Pilha (*Stack*)

modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`



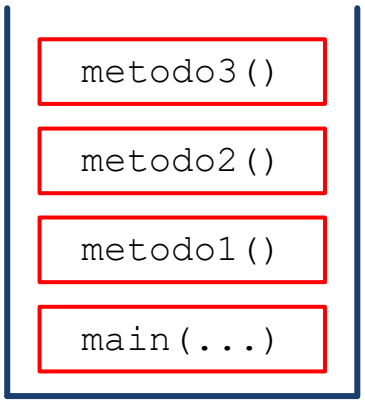
Pilha (*Stack*)

modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`
  - Fim do `metodo3`



Pilha (*Stack*)

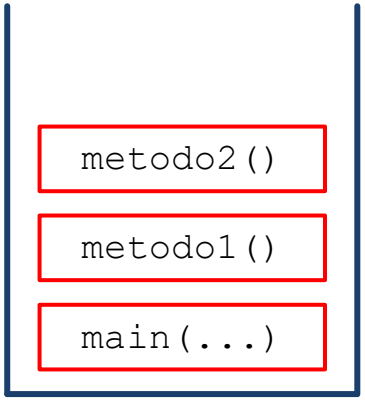
modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`
  - Fim do `metodo3`
  - Fim do `metodo2`



Pilha (*Stack*)

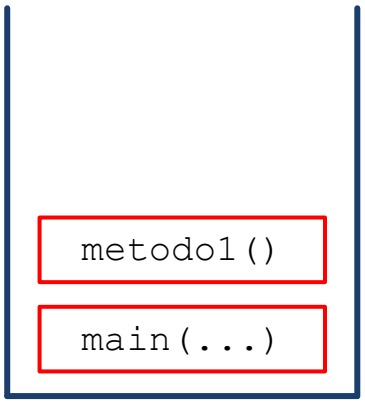
modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`
  - Fim do `metodo3`
  - Fim do `metodo2`
  - Fim do `metodo1`



Pilha (*Stack*)

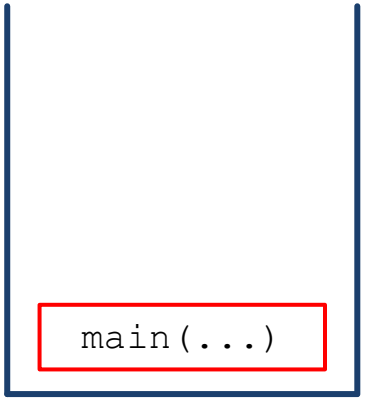
modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`
  - Fim do `metodo3`
  - Fim do `metodo2`
  - Fim do `metodo1`
  - Fim método `main`



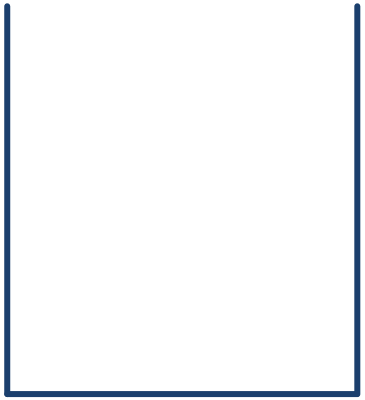
Pilha (*Stack*)

modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

Pilha de Chamada de Métodos (*Stack Trace*)

- Praticar depuração
  - Início método `main`
  - Início do `metodo1`
  - Início do `metodo2`
  - Início do `metodo3`
  - Fim do `metodo3`
  - Fim do `metodo2`
  - Fim do `metodo1`
  - Fim método `main`
  - Programa encerrado



Pilha (*Stack*)

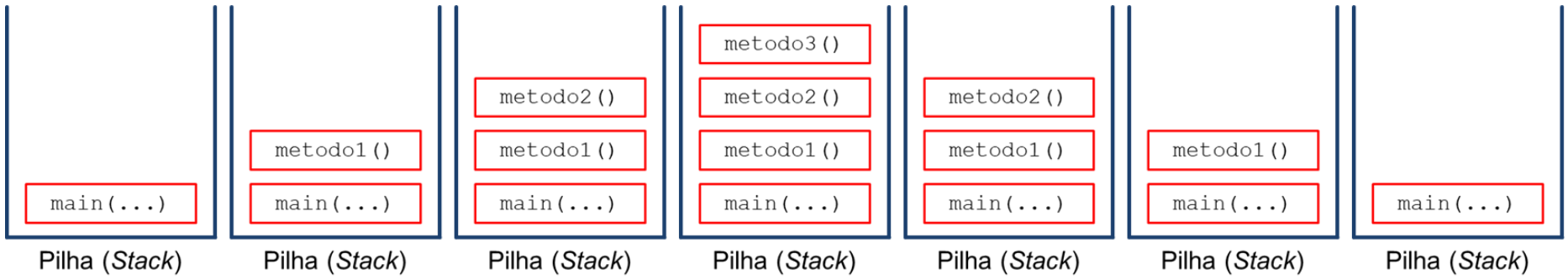
modulo2 - Ex2Solucao.java

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Visão geral da pilha



Pilha de Chamada

- Visualização da pilha de execução de chamadas de métodos (*stack trace*)

```java
public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");
        Double.parseDouble("10,5");
        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

Properties | Console | Problems | Error Log

<terminated> TestaPilhaChamadas [Java Application] C:\sts-4.14.1\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (7 de ago. de 2022 20:55:1

```
*** Início do método main() ***
        *** Início do metodo1() ***
                *** Início do metodo2() ***
                        *** Início do metodo3() ***
Exception in thread "main" java.lang.NumberFormatException: For input string: "10,5"
        at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
        at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.base/java.lang.Double.parseDouble(Double.java:651)
        at pilha.TestaPilhaChamadas.metodo3(TestaPilhaChamadas.java:24)
        at pilha.TestaPilhaChamadas.metodo2(TestaPilhaChamadas.java:18)
        at pilha.TestaPilhaChamadas.metodo1(TestaPilhaChamadas.java:12)
        at pilha.TestaPilhaChamadas.main(TestaPilhaChamadas.java:6)
```

UNOESC

Pilha de Chamada de Métodos (*Stack Trace*)

- Tratando a exceção e imprimindo o *stack trace*

```java
package pilha;

public class TestaPilhaChamadas {
    public static void main(String[] args) {
        System.out.println("*** Início do método main() ***");
        metodo1();
        System.out.println("*** Fim do método main() ***");
    }

    public static void metodo1() {
        System.out.println("\t*** Início do metodo1() ***");
        metodo2();
        System.out.println("\t*** Fim do metodo1() ***");
    }

    public static void metodo2() {
        System.out.println("\t\t*** Início do metodo2() ***");
        metodo3();
        System.out.println("\t\t*** Fim do metodo2() ***");
    }

    public static void metodo3() {
        System.out.println("\t\t\t*** Início do metodo3() ***");

        try {
            Double.parseDouble("10,5");
        } catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println("\t\t\t*** Fim do metodo3() ***");
    }
}
```

Properties | Console | Problems | Error Log

<terminated> TestaPilhaChamadas [Java Application] C:\sts-4.14.1\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (7 de ago. de 2022 21:07:

```
*** Início do método main() ***
        *** Início do metodo1() ***
                *** Início do metodo2() ***
                        *** Início do metodo3() ***
java.lang.NumberFormatException: For input string: "10,5"
        at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
        at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.base/java.lang.Double.parseDouble(Double.java:651)
        at pilha.TestaPilhaChamadas.metodo3(TestaPilhaChamadas.java:26)
        at pilha.TestaPilhaChamadas.metodo2(TestaPilhaChamadas.java:18)
        at pilha.TestaPilhaChamadas.metodo1(TestaPilhaChamadas.java:12)
        at pilha.TestaPilhaChamadas.main(TestaPilhaChamadas.java:6)
                        *** Fim do metodo3() ***
                *** Fim do metodo2() ***
        *** Fim do metodo1() ***
*** Fim do método main() ***
```

UNOESC

Apoiadores: UNOESC, CITEOSC, INOVALE, CENTRO DE INOVAÇÃO VIDEIRA, DEV-TI, SC TEC, fapesc, SANTA CATARINA