SENAI CIMATEC

SISTEMA FIEB

TECNOLOGIA E INOVAÇÃO EM PROL DA INDÚSTRIA

SENAI CIMATEC

SISTEMA FIEB

Curso Técnico em Informática

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Exceções, Tratamento de erros

Prof: Diego Corrêa

Conceito Exceções

- São eventos que ocorrem durante a execução de um programa e quebram o fluxo normal de execução das instruções

- Indicam a ocorrência de erros ou condições excepcionais no programa

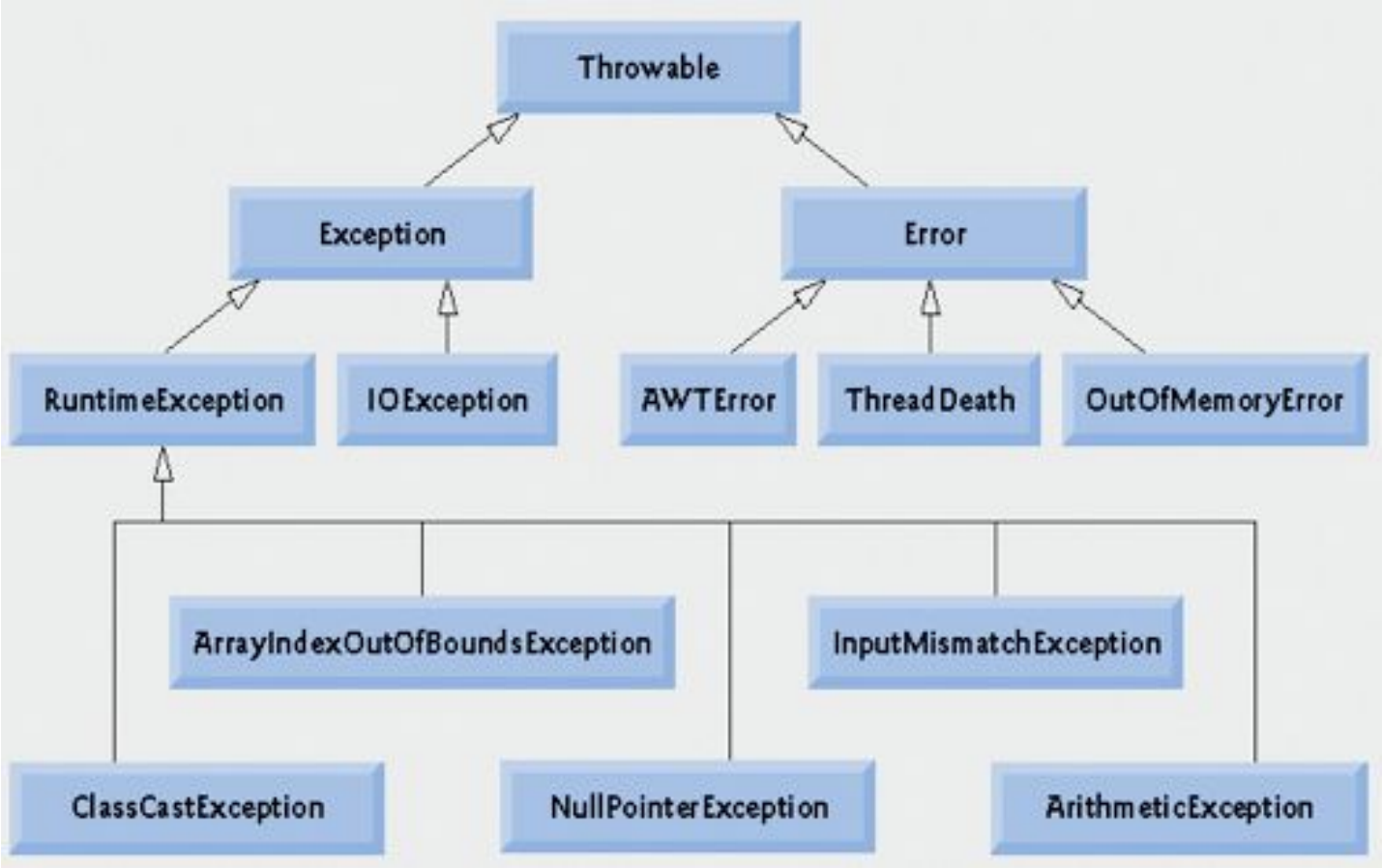
Tratamento

- Impedir o fechamento repentino da aplicação

- Avisar ao usuário/sistema externo que um determinado procedimento não pode ser concluído
  - Erros nos dados
  - Recurso não disponível

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Divisão



Como Gerar um Erro

Implementem os dois próximos programas e verifiquem os erros que aparecem

Como Gerar um Erro

```java
package tratamentoerros;

public class TratamentoErros {

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

        for(int i = 0; i <= 10; i++) {
            numeros_vetor[i] = i;
            System.out.println(i);
        }
    }

}
```

```
run:
0
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
1
2
3
4
        at tratamentoerros.TratamentoErros.main(TratamentoErros.java:9)
/home/lowe/.cache/netbeans/8.2/executor-snippets/run.xml:53: Java returned: 1
FALHA NA CONSTRUÇÃO (tempo total: 0 segundos)
```

Como Gerar um Erro

```java
package tratamentoerroteclado;

import java.util.Scanner;

public class TratamentoErroTeclado {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Digite a idade: ");

        int idade = sc.nextInt();
        System.out.println(idade);
    }
}
```

```
run:
Digite a idade: TESTE
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at tratamentoerroteclado.TratamentoErroTeclado.main(TratamentoErroTeclado.java:11)
/home/lowe/.cache/netbeans/8.2/executor-snippets/run.xml:53: Java returned: 1
FALHA NA CONSTRUÇÃO (tempo total: 2 segundos)
```

Tipos

- Erros aritméticos
- Estouro de limite de array
- Entrada de dados inválidos
- Erros na manipulação de arquivos
- Erros na comunicação com bancos de dados
- Falhas de comunicação entre programas distribuídos

Tipos

A classe Throwable tem duas subclasses:

Tipos

- Exception (java.lang.Exception) – É a raiz das classes originárias da classe Throwable, onde mostra as situações em que a aplicação pode querer capturar e realizar um tratamento para conseguir realizar o processamento

Tipos

- Error (java.lang.Error) – Também é raiz das classes originárias da classe Throwable, indicando as situações em que a aplicação não deve tentar tratar, como ocorrências que não deveriam acontecer

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Classificação

Classe	Objetivo	Eventos
Error	Classificada como exceção que não pode ser tratada pela aplicação.	
Exception	Trata todas as exceções da aplicação que podem ser tratadas e capturadas.	I/O Exception ou aplicar uma divisão por zero resultando na exceção **ArithmeticException**
RuntimeException	Resgata as exceções lançadas pela máquina virtual (JVM).	Referência nula (**NullPointerException**)

Conceito

- O uso das exceções em um sistema é de extrema importância
- Pois ajuda a detectar e tratar possíveis erros que possam acontecer
- Entretanto, na linguagem **Java** existem dois tipos de exceções, que são:

Conceito

- **Implícitas**: Exceções que não precisam de tratamento e demonstram serem contornáveis. Esse tipo origina-se da subclasse **Error** ou **RunTimeException**
- **Explícitas**: Exceções que precisam ser tratadas e que apresentam condições incontornáveis. Esse tipo origina do modelo throw e necessita ser declarado pelos métodos. É originado da subclasse **Exception** ou **IOException**

Conceito

Existe também a formação de erros dos tipos **throwables** que são:

Conceito

- **Checked Exception**: Erros que acontecem fora do controle do programa, mas que devem ser tratados pelo desenvolvedor para o programa funcionar

Conceito

- **Unchecked (Runtime)**: Erros que podem ser evitados se forem tratados e analisados pelo desenvolvedor. Caso haja um tratamento para esse tipo de erro, o programa acaba parando em tempo de execução (Runtime)

Conceito

- **Error**: Usado pela JVM que serve para indicar se existe algum problema de recurso do programa, tornando a execução impossível de continuar

Palavra Reservada

- Em **Java**:
  - **try**, **catch** e **finally**
    - Define um bloco de tratamento de exceção
  - **throws**
    - Declara que um método pode lançar uma exceção ou mais exceções.
  - **throw**
    - Lança uma exceção

try

- O bloco **try** tenta processar o código que está dentro, sendo que se ocorrer uma exceção, a execução do código pula para a primeira captura do erro no bloco **catch**

- O uso do **try** serve para indicar que o código está tentando realizar algo arriscado no sistema

catch

- O bloco catch trata a exceção lançada
- Caso a exceção não seja esperada, a execução do código pula para o próximo catch, se existir
- Portanto, se nenhum do bloco catch conseguir capturar a exceção, dependendo o tipo que for, é causada a interrupção ao sistema, lançando a exceção do erro

finally

- A bloco **finally** sempre finaliza a sequência de comandos do sistema, independente de ocasionar algum erro no sistema

- Esse bloco é opcional, não sendo obrigatório sua inserção na sequência try/catch

- É usado em ações que sempre precisam ser executadas independente se gerar erro

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Código

```
try {
        //Código que pode gerar exceção

} catch (Exception1 e) {
        //Tratamento das exceções
} catch (Exception2 e) {
        //Tratamento das exceções
}
(..)
} catch (ExceptionN e) {
        //Tratamento das exceções
} finally {
        //Executa esse trecho mesmo que uma
        //exceção ocorra.
}
```

Capturando Mensagens

- A classe **Throwable** oferece alguns métodos que podem verificar os erros reproduzidos, quando gerados para dentro das classes

- Esse tipo de verificação é visualizado no rastro da pilha (stracktrace), que mostra em qual linha foi gerada a exceção

Capturando Mensagens

- **printStrackTrace** – Imprime uma mensagem da pilha de erro encontrada em um exceção
- **getStrackTrace** – Recupera informações do stracktrace que podem ser impressas através do método printStrackTrace
- **getMessage** – Retorna uma mensagem contendo a lista de erros armazenadas em um exceção no formato String

Capturando Mensagens

Alterem os exemplos do início da aula para que os erros sejam capturados

Capturando Mensagens

```
package tratamentoerros;

public class TratamentoErros {

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

            for(int i = 0; i <= 10; i++) {
                numeros_vetor[i] = i;
                System.out.println(i);
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.err.println("Um Erro foi encontrado!");
            System.err.println("Array fora do índice: " + e.getMessage());
        }
    }

}
```

```
run:
0
Um Erro foi encontrado!
1
2
3
Array fora do índice: 5
4
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Capturando Mensagens

```java
package tratamentoerroteclado;

import java.util.InputMismatchException;
import java.util.Scanner;

public class TratamentoErroTeclado {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try{
            System.out.print("Digite a idade: ");

            int idade = sc.nextInt();
            System.out.println(idade);
        } catch (InputMismatchException e) {
            System.out.println("Um Erro ocorreu!");
            System.out.println("A seguir encontra-se a arvore de execução.");
            e.printStackTrace();
        }
    }

}
```

```
run:
Digite a idade: TESTE
Um Erro ocorreu!
A seguir encontra-se a arvore de execução.
java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at tratamentoerroteclado.TratamentoErroTeclado.main(TratamentoErroTeclado.java:13)
CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos)
```

Lançamento de Exceções

- As cláusulas throw e throws podem ser entendidas como ações que propagam exceções

- Ou seja, em alguns momentos existem exceções que não podem ser tratadas no mesmo método que gerou a exceção

- Nesses casos, é necessário propagar a exceção para um nível acima na pilha

Lançamento de Exceções

- Indicar na assinatura dos métodos que exceções podem ser lançadas
  - Utiliza-se a diretiva “**throws**”

- Indicar no corpo do método os locais nos quais as exceções são lançadas
  - Utiliza-se a diretiva “**throw**”

Lançamento de Exceções

```
public void metodo( ) throws Excecao1, Excecao2 {
        ...
}
```

```java
package exemplodivisao;

import java.util.InputMismatchException;
import java.util.Scanner;

public class ExemploDivisao {

     public static int calculaQuociente(int numerador, int denominador) throws ArithmeticException{
        return numerador / denominador;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean continua = true;

        do{
            try{
                System.out.print("Numerador: ");
                int numerador = sc.nextInt();

                System.out.print("Denominador: ");
                int denominador = sc.nextInt();

                int resultador = calculaQuociente(numerador, denominador);
                System.out.println("Resultado: "+resultador);

                continua = false;

            }catch (InputMismatchException erro1) {
                System.err.println("Não é permitido inserir letras, informe apenas números inteiros!");
                sc.nextLine(); //descarta a entrada errada do usuário
            }catch(ArithmeticException erro2){
                System.err.println("O número do divisor deve ser diferente de 0!");
            }finally{
                System.out.println("Execução do Finally!");
            }
        }while(continua);
    }

}
```

Lançamento de Exceções

```
run:
Numerador: 1
Denominador: 0
O número do divisor deve ser diferente de 0!
Execução do Finally!
Numerador: 1
Denominador: 5
Resultado: 0
Execução do Finally!
CONSTRUÍDO COM SUCESSO (tempo total: 22 segundos)
```

Definindo as Próprias Exceções

Para criar a própria definição de erro é necessário criar uma classe que **extenda** da classe **Exception**

Definindo as Próprias Exceções

- Usando **throw**
  - Palavra reservada utilizada para lançar uma exceção

Exemplo 1:

```
// Instanciando e lançando o objeto Exception
throw new Exception("Mensagem de ERRO!");
```

Definindo as Próprias Exceções

Exemplo 2:

```
// Instanciação do objeto Exception
Exception e = new Exception("Mensagem de ERRO!");
// Lançando a exceção
throw e;
```

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Definindo as Próprias Exceções

```
package lancandothrow;

public class ExcecaoTextoInvalido extends Exception {
    public ExcecaoTextoInvalido(String mensagem) {
        super(mensagem);
    }
}
```

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Definindo as Próprias Exceções

```
package lancandothrow;

public class LancandoThrow {

    public static void main(String[] args) {
        try {
            verificaDigitacao("Texto massa pra imprimir");
            verificaDigitacao("Mais um texto sem erro");
            verificaDigitacao("");
        } catch (ExcecaoTextoInvalido e) {
            System.err.println(e.getMessage());
        }
    }

    public static void verificaDigitacao(String texto) throws ExcecaoTextoInvalido {
       if (texto.equals("") || texto == null || texto.equals(" ")) {
           throw new ExcecaoTextoInvalido("Texto inválido. Não pode ser vazio!");
       } else {
           System.out.println("O Texto é: " + texto);
       }
    }
}
```