DIGITAL
INNOVATION
ONE

# Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

DIGITAL INNOVATION ONE

# Objetivo do curso

Ao final deste curso, o Dev será capaz de identificar possíveis exceções de um aplicação Java e interpretar eventuais pilhas de exceção. Assim, você estará preparado para capturar e tratar essas exceções, deixando suas soluções ainda mais robustas.


# Pré-requisitos

1. Java JDK 8 ou superior
2. IDE para desenvolvimento Java
3. Conhecimento BÁSICO em OOP
4. Estar disposto a aprender

DIGITAL INNOVATION ONE

# Percurso

1. Visão Geral
2. Unchecked Exception
3. Checked Exception
4. Exception Personalizada

DIGITAL INNOVATION ONE

# Mais sobre mim

- Estudante de Ciência da Computação
- Comecei no mundo da TI através do excel
- Minha motivação é a vontade de aprender coisas novas
- Nas horas vagas gosto de assistir séries e ler livros

DIGITAL INNOVATION ONE

# Redes Sociais

- https://github.com/cami-la/exceptions-java
- https://www.linkedin.com/in/cami-la/
- https://www.instagram.com/estudant.i/
- https://www.instagram.com/camimi_la/


# Dúvidas durante o curso?

> Fórum do curso
> Comunidade online (discord)

DIGITAL
INNOVATION
ONE

# Visão Geral

## Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

DIGITAL INNOVATION ONE

# Tratamento de Exceções

- Exceção é um evento que interrompe o fluxo normal do processamento de uma classe.
- O uso correto de exceções torna o programa mais robusto e confiável.
- Com o tratamento de exceções, um programa pode continuar executando depois de lidar com um problema.
- Importante: Incorpore sua estratégia de tratamento de exceções no sistema desde o princípio do processo do projeto.
- Pode ser difícil incluir um tratamento de exceções eficiente depois que um sistema foi implementado.


# Tratamento de Exceções

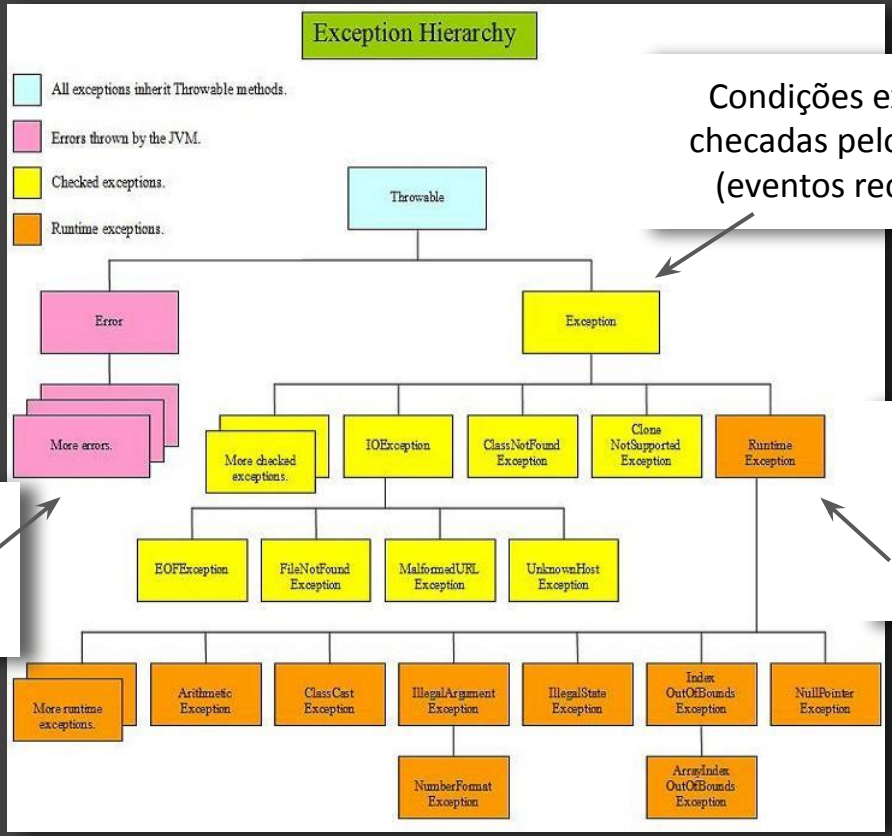
- **Error**: Usado pela JVM que serve para indicar se existe algum problema de recurso do programa, tornando a execução impossível de continuar.

- **Unchecked (Runtime)**: Exceptions que PODEM ser evitados se forem tratados e analisados pelo desenvolvedor.

- **Checked Exception**: Exceptions que DEVEM ser evitados e tratados pelo desenvolvedor para o programa funcionar.

DIGITAL INNOVATION ONE

# Tratamento de Exceções




Condições excepcionais checadas pelo compilador (eventos recuperáveis)

Erros não checados pelo compilador (eventos irrecuperáveis)

Exceções não checadas pelo compilador (erros de lógica)

DIGITAL INNOVATION ONE

# Tratamento de Exceções

- **try, catch, finally**: Cada uma dessas palavras, juntas, definem blocos para o tratamento de exceções.

- **throws**: Declara que um método pode lançar uma ou várias exceções.

- **throw**: Lança explicitamente uma exception.

DIGITAL INNOVATION ONE

# Redes Sociais

- https://github.com/cami-la/exceptions-java
- https://www.linkedin.com/in/cami-la/
- https://www.instagram.com/estudant.i/
- https://www.instagram.com/camimi_la/


# Dúvidas durante o curso?

> Fórum do curso
> Comunidade online (discord)

DIGITAL
INNOVATION
ONE

# try - catch - finally

## Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

DIGITAL INNOVATION ONE

# Tratamento de Exceções

```java
public class EstruturaTryCatchFinally {
    public static void main(String[] args) {
        try {
            // trecho do código que pode gerar exceção
        } catch (Exception ex) {
            //tratamento da exceção
        } finally {
            //bloco que será "sempre" executado
        }
    }
}
```

DIGITAL INNOVATION ONE

# Tratamento de Exceções

Bloco ***try***:

- Região onde se encontra o código que queremos verificar se irá ou não lançar uma exceção.
- Caso ocorra uma exceção em algum ponto, o restante do código contido no bloco *try* não será executado.
- O bloco *try* não pode ser declarado sozinho, por tanto, precisa estar seguido de um ou vários blocos catch e/ou de um bloco finally.

DIGITAL INNOVATION ONE

# Tratamento de Exceções

Bloco ***catch***:

- Região onde se encontra o possível tratamento da exceção. Isso significa que só será executado caso o bloco try apresentar alguma exceção.
- Recebe como argumento a classe ou subclasse da possível exceção.
- No seu escopo ficam as instruções de como tratar essa exceção.
- Pode haver mais de um bloco catch, porém, será executado apenas o primeiro bloco que identificar a exceção.

DIGITAL INNOVATION ONE

# Tratamento de Exceções

Bloco ***finally***:

- Este bloco é opcional, mas caso seja construído, quase sempre será executado. (A menos que seja forçado, por exemplo, com um System.exit(0), no catch).
- Dentro do bloco finally, poderá conter outros blocos try, catch, bem como outro finally.

DIGITAL INNOVATION ONE

# Redes Sociais

- https://github.com/cami-la/exceptions-java
- https://www.linkedin.com/in/cami-la/
- https://www.instagram.com/estudant.i/
- https://www.instagram.com/camimi_la/


# Dúvidas durante o curso?

> Fórum do curso
> Comunidade online (discord)

DIGITAL
INNOVATION
ONE

# throw e throws

## Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

DIGITAL INNOVATION ONE

# Tratamento de Exceções

Cláusula ***throw***:

- Para lançar exceções explicitamente, use a cláusula *throw*.
- Usada principalmente para lançar **exceções personalizadas**.
- Caso um tratador adequado não seja encontrado no bloco onde a exceção foi lançada, ela é propagada para o nível mais externo.
- A propagação contínua até que algum tratador seja encontrado ou até chegar ao nível da JVM.
- Pode ser utilizada tanto para exceções *checked* ou *unchecked.*

DIGITAL INNOVATION ONE

# Tratamento de Exceções

Cláusula ***throws***:

- O *throws* quando declarado no método, servirá apenas para informar ao compilador que estamos cientes da possibilidade de apresentar alguma Exception neste método.
- Para que um método possa disparar uma exceção é necessário colocar a cláusula throws na definição do mesmo, indicando quais tipos de exceção o mesmo pode retornar.
- A responsabilidade de tratar o método lançado fica no código que chamou o método, podendo tratá-la ou lançá-la novamente.

DIGITAL
INNOVATION
ONE

# Unchecked Exception e Checked Exception

## Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

# Tratamento de Exceções

```java
public class Exemplo_1 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        int resultado = dividir(scan.nextInt(), scan.nextInt());
        System.out.println(resultado);
    }

    public static int dividir(int a, int b) {
        return a / b;
    }
}
```

Errors e RuntimeExceptions são considerados *uncheked*, portanto o compilador não obriga que exista tratamento para eles.

**Unchecked Exception**

```
Exemplo_1 ×
/home/cami/programs/jdk-17/bin/java -javaagent:/home/cami/programs/ides/ideaIC-2021.
4
0
Exception in thread "main" java.lang.ArithmeticException Create breakpoint : / by zero
    at br.com.dio.Exemplo_1.dividir(Exemplo_1.java:14)
    at br.com.dio.Exemplo_1.main(Exemplo_1.java:9)

Process finished with exit code 1
```

DIGITAL INNOVATION ONE

# Tratamento de Exceções

**Unchecked Exception**

- Herdam da classe **RuntimeException** ou da classe **Error**.
- O compilador não verifica o código para ver se a exceção foi capturada ou declarada.
- Se uma exceção não-verificada ocorrer e não tiver sido capturada, o programa terminará ou executará com resultados inesperados.
- Em geral, podem ser evitadas com uma codificação adequada.

DIGITAL INNOVATION ONE

# Tratamento de Exceções

```java
public class Exemplo_2 {
    public static void main(String[] args) {
        imprimeArquivoNoConsole( nomeDoArquivo: "romances-blake-crouch.txt");
    }

    private static void imprimeArquivoNoConsole(String nomeDoArquivo) {
        File file = new File(nomeDoArquivo);
        BufferedReader br = new BufferedReader(new FileReader(file.getName()));
        String line = br.readLine();

        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));

        do{
            bw.write(line);
            bw.newLine();
            line=br.readLine();
        } while(line != null);
        bw.flush();
        br.close();
    }
}
```

```
Build:  Build Output ×
exceptions-java: build failed At 9/27/21, 3:33 PM with 7 errors                    996 ms
  Exemplo_2.java src/br/com/dio/exemplos 7 errors
    unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown :12
    unreported exception java.io.IOException; must be caught or declared to be thrown :13
    unreported exception java.io.IOException; must be caught or declared to be thrown :18
    unreported exception java.io.IOException; must be caught or declared to be thrown :19
    unreported exception java.io.IOException; must be caught or declared to be thrown :20
    unreported exception java.io.IOException; must be caught or declared to be thrown :22
    unreported exception java.io.IOException; must be caught or declared to be thrown :23
```

## Checked Exception

Costumam indicar que uma condição necessária para a execução de um programa não está presente.

DIGITAL INNOVATION ONE

# Tratamento de Exceções

**Checked Exception**

- As exceções que são herdadas da classe Exception, mas não de RuntimeException.
- O compilador impõe um requisito do tipo ‘capturar ou declarar’.
- O compilador verifica cada chamada de método e declaração de método para determinar se o método lança (*throws*) exceções verificadas. Se lançar, o compilador assegura que a exceção verificada é capturada ou declarada em uma cláusula *throws*. Caso não capturada nem declarada, ocorre um erro de compilação.

DIGITAL
INNOVATION
ONE

# Exception Personalizada

## Tratamento de Exceções em Java

Camila Cavalcante
Tech Teacher

# Tratamento de Exceções

```java
public class Exemplo_3 {
    public static void main(String[] args) {
        int[] numerador = {4, 2, 5, 8, 10};
        int[] denominador = {2, 0, 4, 0, 2, 8};

        for(int i = 0; i < denominador.length; i++) {
            int resultado = numerador[i] / denominador[i];
            System.out.print(resultado + " ");
        }
    }
}
```

```
Exemplo_3 ×
/home/cami/programs/jdk-17/bin/java -javaagent:/home/cami/programs/ides/
2 Exception in thread "main" java.lang.ArithmeticException: / by zero
    at test.Exemplo_3.main(Exemplo_3.java:9)
```

```
Exemplo_3 ×
/home/cami/programs/jdk-17/bin/java -javaagent:/home/cami/programs/ides/ideaIC-2021.1.3/idea-IC-211.7628.21/lib/ide
2 0 1 2 5 Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
    at test.Exemplo_3.main(Exemplo_3.java:9)

Process finished with exit code 1
```

## Checked Customizada

Assim como qualquer objeto, em Java também é possível criar suas próprias exceções.

DIGITAL INNOVATION ONE

# Tratamento de Exceções

**Exception Personalizada:**

- Programadores podem achar útil declarar suas próprias classes de exceção.
- Essas Exceptions são específicas aos problemas que podem ocorrer quando outro programador empregar suas classes reutilizáveis.
- Uma nova classe de exceção deve estender uma classe de exceção existente que assegura que a classe pode ser utilizada com o mecanismo de tratamento de exceções, logo essas Exceções customizadas são derivadas da classe Exception.

DIGITAL INNOVATION ONE

# Redes Sociais

- https://github.com/cami-la/exceptions-java
- https://www.linkedin.com/in/cami-la/
- https://www.instagram.com/estudant.i/
- https://www.instagram.com/camimi_la/


# Dúvidas durante o curso?

> Fórum do curso
> Comunidade online (discord)