Introdução

Em linhas gerais, nesse módulo estudamos sobre como tratar erros em projetos Java.

Lembrem-se, mesmo quando a aplicação que desenvolvemos gerar algum tipo de erro, esse erro precisará ser tratado para que o usuário entenda o problema que está enfrentando.

Vamos lá?

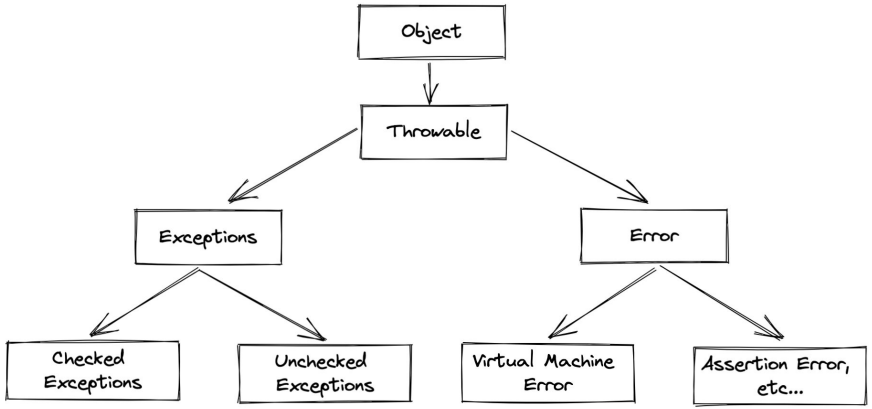
Exceções

Quando trabalhamos com projetos de software, uma coisa é certa, problemas inesperados sempre existirão.

Esses cenários inesperados são conhecidos como **exceções** ou **exceptions** e nossas aplicações devem estar preparadas para tratar as exceções.

Java Exceptions

No Java, as Exceptions são representadas por classes que possuem uma hierarquia para diferentes tipos de exceções:



Checked Exceptions

Checked Exceptions são exceções que podem ser identificadas e tratadas em tempo de compilação. Aqui estão exemplos de exceções do tipo **Checked Exceptions**:

- **IOException**: caracteriza-se por informar que uma operação de entrada/saída (*Input/Output*) falhou ou foi interrompida. Normalmente ocorre quando um arquivo não pode ser lido;
- **SQLException**: pode ocorrer caso haja uma operação inválida em um banco de dados SQL, como falha de conexão, fim do tempo de execução, etc...

Unchecked Exceptions

Unchecked Exceptions são exceções que podem ocorrer durante a execução do programa, potencialmente ocasionando sua parada. As mais comuns são:

- **NullPointerException**: ocorre quando há a tentativa de manipular um objeto nulo (null). É possível evitar essa exceção verificando se o objeto está nulo antes de manipulá-lo;
- **ArrayIndexOutOfBoundsException**: ocorre quando há a tentativa de acessar um índice inexistente de um Array;
- **ArithmeticException**: ocorre quando há a tentativa de executar uma operação aritmética inválida, como por exemplo dividir um número por 0.

Error

Errors são exceções que não podem ser tratadas e ocorrem no nível da Máquina Virtual do Java, JVM. Normalmente essas exceções ocasionam o fim da execução do programa.

- **OutOfMemoryException**: ocorre quando não há memória suficiente disponível para que o programa continue em execução.

Tratamento de exceções

Toda Checked Exception exige que o programa as identifique e, possivelmente, trate as exceções.

Try, Catch, Finally.

Podemos tratar uma exceção usando o *Try*, *Catch*, *Finally*:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class Main {
   public static void main(String[] args) {
       try {

Files.readAllLines(Paths.get("/Users/lucassantos/Documents/alunos.txt"));
       } catch (IOException e) {
           System.out.println("Não é possível ler o arquivo.");
       } finally {
           System.out.println("Finalizando");
       }
   }
}
```

No exemplo anterior, estamos tentando ler um arquivo no caminho "*/Users/lucassantos/Documents/alunos.txt*". No entanto é possível que o arquivo não exista ou esteja inacessível e nesse caso o método "readAllLines" especifica que a exceção "**IOException**" pode ser lançada.

Usamos o *Try/Catch* para tratar essa exceção caso ela aconteça. Se o erro acontecer, o programa não terminará sua execução com um erro. Ao invés disso, mostramos uma mensagem no terminal informando que não é possível ler o arquivo.

Todo o código que estiver dentro do bloco "*try*" indica que alguma exceção pode ocorrer e no bloco "*catch*" especificamos quais exceções são esperadas.

O código dentro do bloco "*finally*" é executado independentemente de ocorrer alguma exceção ou não, sendo esse bloco opcional.

Throws

O tratamento de exceções não é obrigatório no Java, mas é necessário informar na assinatura do método quais exceções podem ser lançadas caso haja a possibilidade de alguma ocorrer.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class Arquivos {
   public static List<String> lerArquivo(String caminho)
throws IOException {
        return Files.readAllLines(Paths.get(caminho));
   }
}

import java.io.IOException;

public class Main {
   public static void main(String[] args)  throws IOException
{

Arquivos.lerArquivo("/Users/lucassantos/Documents/alunos.txt
");
   }
}
```

Na classe "*Arquivos*" é definido o método "*lerArquivo*". Nesse método também chamamos o método "*readAllLines*" da classe "Files". Mas ao invés de tratar a exceção "*IOException*", apenas definimos que na assinatura do método "*lerArquivo*" que a exceção pode ser lançada. Caso a exceção ocorra, também é definido na assinatura do método "main" que a exceção "*IOException*" pode ser lançada.

Se em nenhum momento a exceção for tratada com o *Try/Catch*, o programa é encerrado com um erro.