VEM SER

Módulo 01 - Java + OO

Aula 07 - Enum e Exceptions

Conteúdo da aula

- **Enums;**
- **Exceptions:**
  - Erros;
  - Exceções.

Enums

Enums

- Em Java, os enums são uma forma de definir um **conjunto de constantes**;
- Eles são usados para representar um valor que pode ser um de um número finito de valores.

Sintaxe básica

```
public enum TipoEndereco {
  RESIDENCIAL,
  COMERCIAL;
}
```

Atributos

```
public enum TipoEndereco {
  RESIDENCIAL(1),
  COMERCIAL(2);

  // não é estático pois o valor vai estar relacionado ao objeto instanciado e não à classe
  // pode ser público, privado ou protegido
  private final int VALOR;

  TipoEndereco(int valor) {
    this.VALOR = valor;
  }

// lembrar do get caso seja protegido/privado
  public int getVALOR() {
    return VALOR;
  }
}
```

Método

```
public enum TipoPagamento {
  DEBITO(0.03),
  CREDITO(0.05),
  PIX(0),
  DINHEIRO(0);

private final double TAXA;

  TipoPagamento(double taxa) {
    this.TAXA = taxa;
  }

  public double getTAXA() {
    return this.TAXA;
  }
}
```

Exceptions

Exceptions

- As exceções são usadas para tratar erros;
- Elas são uma forma de **interromper a execução** do código e informar o erro ao usuário.

Unchecked vs Checked

Existem dois tipos de exceções em Java:

- Exceções verificadas (*checked*);
- Exceções não verificadas (*unchecked*).

Unchecked vs Checked

- Exceções verificadas são **exceções** que devem ser tratadas pelo programador. Elas são representadas por classes que derivam da classe ***Exception***;
    - Não são contornáveis em tempo de execução e precisam interromper o fluxo do programa para que o erro seja tratado.
- Exceções não verificadas são **erros** que não precisam ser tratadas pelo programador. Elas são representadas por classes que derivam da classe ***RuntimeException*** ou ***Error***;

Exception Hierarchy

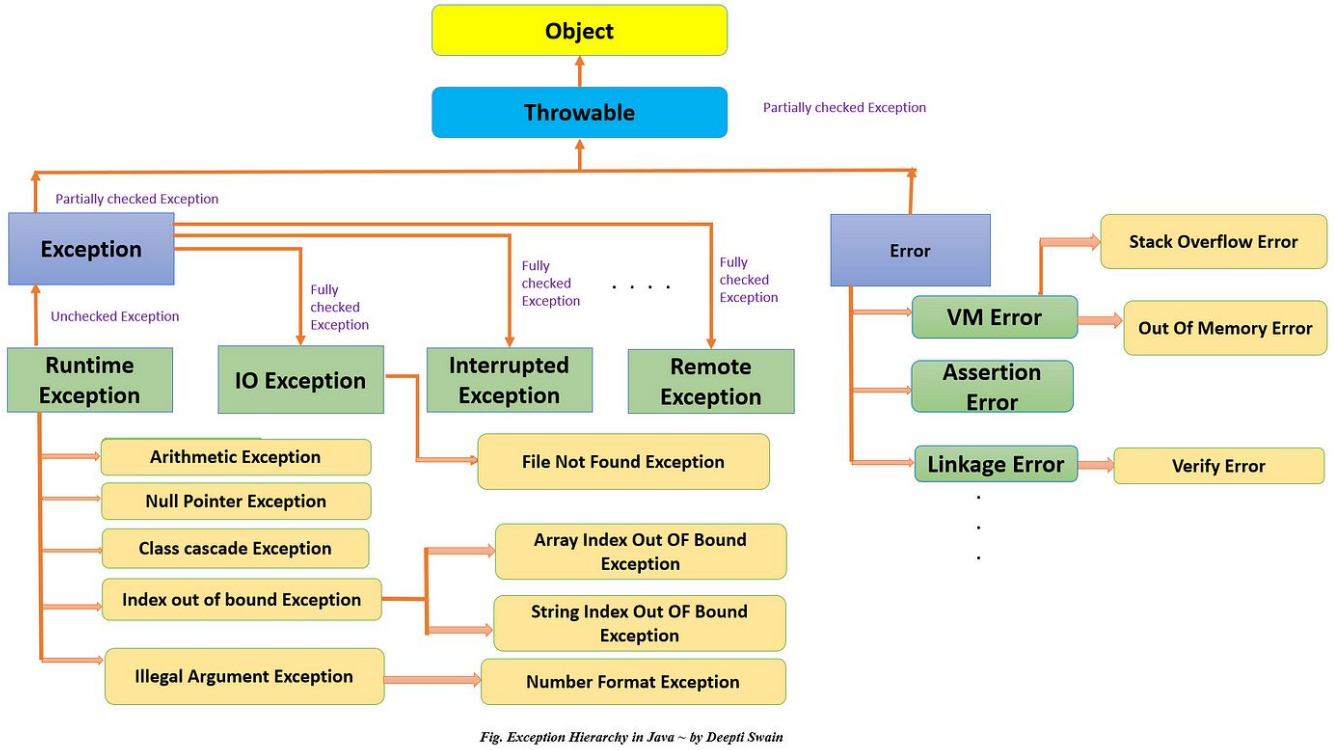

Fig. Exception Hierarchy in Java ~ by Deepti Swain

Crédito da imagem: https://interviewnoodle.com/exception-in-java-89a0b41e0c45

Exception

- Todas as subclasses de ***Exception*** (exceto as subclasses *RuntimeException*) são **exceções e devem ser tratadas;**
- Os erros da classe ***Error*** ou ***RuntimeException*** são **erros** e não precisam de tratamento.

Tratando erros com try/catch/finally

```
try {
  int x = 1 / 0;
} catch (NomeDaExceção e) {
  System.out.println("Ocorreu um erro de divisão por zero: " + e.getMessage());
} finally {
  System.out.println("Este código sempre será executado.");
}
```

Referências

https://docs.oracle.com/javase/8/docs/api/java/lang/Enum.html
https://docs.oracle.com/javase/8/docs/api/?java/lang/Error.html
https://docs.oracle.com/javase/8/docs/api/java/lang/RuntimeException.html
https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html


Let's Tech Up Together