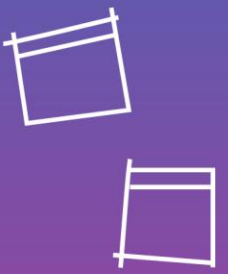
DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC



Enums + Exceptions + Generics + CRUD

Sumário

- Enumeradores
- Exceções
- Generics
- CRUD

Enums

- Uma enum ou enumeração é um tipo no qual declaramos um conjunto de valores constantes pré-definidos.

https://www.devmedia.com.br/enums-no-java/38764

Enums

- Uma enum ou enumeração é um tipo no qual declaramos um conjunto de valores constantes pré-definidos.

```
public enum Turno  {
    MANHA, TARDE, NOITE;
}
```

https://www.devmedia.com.br/enums-no-java/38764

Enums

- Uma enum ou enumeração é um tipo no qual declaramos um conjunto de valores constantes pré-definidos.

```
public enum Turno  {
    MANHA, TARDE, NOITE;
}
```

```
Turno  turno = Turno.MANHA;
```

https://www.devmedia.com.br/enums-no-java/38764

Enums

```
public enum Turno  {
    MANHA, TARDE, NOITE;
}
```

```
Turno  turno = Turno.MANHA;
```

- Perceba que ao **utilizar enums** limitamos os valores que podem ser atribuídos a uma variável. Sendo assim, devemos atribuir ao campo Turno um dos valores pré-definidos na enum ”Turno”.

https://www.devmedia.com.br/enums-no-java/38764

Enums

- Ao declarar uma enum estamos implicitamente estendendo a classe **java.lang.Enum**

https://www.devmedia.com.br/enums-no-java/38764

Enums

- Ao declarar uma enum estamos implicitamente estendendo a classe **java.lang.Enum**
- Isso cria algumas limitações, porque o Java não suporta herança múltipla, o que impede uma classe **enum** de estender outras classes.

https://www.devmedia.com.br/enums-no-java/38764

Enums

- Ao declarar uma enum estamos implicitamente estendendo a classe **java.lang.Enum**
- Isso cria algumas limitações, porque o Java não suporta herança múltipla, o que impede uma classe **enum** de estender outras classes.
- Uma classe enum pode ter propriedades, assim como construtores e métodos.

https://www.devmedia.com.br/enums-no-java/38764

Enums

```
public enum Turno {
    MANHA("manhã"),
    TARDE("tarde"),
    NOITE("noite");

    private String descricao;

    Turno(String descricao) {
        this.descricao = descricao;
    }

    public String getDescricao() {
        return descricao;
    }
}
```

https://www.devmedia.com.br/enums-no-java/38764

Utilização de Enums

```
for (Turno t : Turno.values()) {
    System.out.println(t.getDescricao());
}
```

Utilização de Enums

```
for (Turno t : Turno.values()) {
    System.out.println(t.getDescricao());
}
```

Método	Retorno	Descrição
toString()	String	Retorna uma String com o nome da instância (em maiúsculas).
valueOf(String nome)	static <T extends Enum<T>> T	Retorna o objeto da classe enum cujo nome é a string do argumento.
ordinal()	int	Retorna o número de ordem do objeto na enumeração.

Let’s practice;

Exercício #1

- Crie um enumerador para tipo de comida, defina os seguintes valores:
  - Japonesa: Custa R$50
  - Fast Food: Custa R$30
  - Tradicional: Custa R$20
- Faça um programa que peça ao usuário um tipo de comida desejado e imprima o valor da comida conforme o tipo.

Links Úteis

- https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html
- https://www.devmedia.com.br/enums-no-java/38764
- https://www.w3schools.com/java/java_enums.asp

Trabalhando com Exceções

https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Trabalhando com Exceções

- Exceção é um evento não esperado que ocorre no sistema quando está em tempo de execução (Runtime)

https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Trabalhando com Exceções

- Exceção é um evento não esperado que ocorre no sistema quando está em tempo de execução (Runtime)
- Para conseguir capturar uma **exceção**, é preciso fazer antes o tratamento.

https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Trabalhando com Exceções

- Exceção é um evento não esperado que ocorre no sistema quando está em tempo de execução (Runtime)
- Para conseguir capturar uma **exceção**, é preciso fazer antes o tratamento.
- O uso dos tratamentos é importante nos sistemas porque auxilia em falhas como: comunicação, leitura e escrita de arquivos, entrada de dados inválidos, acesso a elementos fora de índice, entre outros.

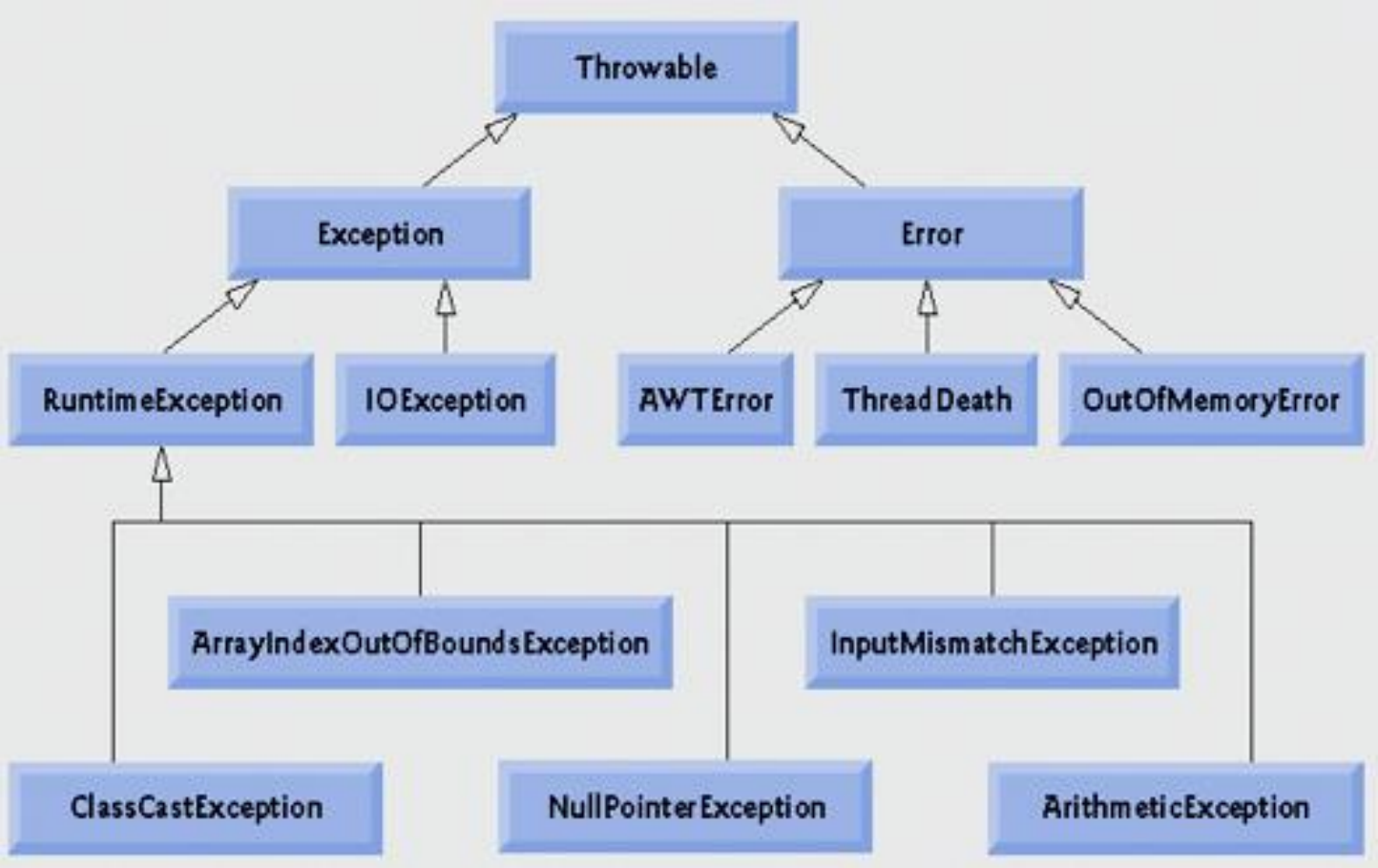
https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Trabalhando com Exceções

- Exceção é um evento não esperado que ocorre no sistema quando está em tempo de execução (Runtime)
- Para conseguir capturar uma **exceção**, é preciso fazer antes o tratamento.
- O uso dos tratamentos é importante nos sistemas porque auxilia em falhas como: comunicação, leitura e escrita de arquivos, entrada de dados inválidos, acesso a elementos fora de índice, entre outros.
- Na linguagem Java existem dois tipos de exceções, que são:
  - **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.

https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Hierarquia de Exceções



https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Tratamento de Exceções

- Existe uma diferença entre “**Erro (Error)”** e “**Exceção (Exception)**”.

Tratamento de Exceções

- Existe uma diferença entre “**Erro (Error)”** e “**Exceção (Exception)**”.
- Todas as subclasses de Exception (menos as subclasses RuntimeException) são exceções e devem ser tratadas.

Tratamento de Exceções

- Existe uma diferença entre “**Erro (Error)”** e “**Exceção (Exception)**”.
- Todas as subclasses de Exception (menos as subclasses RuntimeException) são exceções e devem ser tratadas.
- Os erros da classe Error ou RuntimeException são erros e não precisam de tratamento, por esse motivo é usado o **try/catch** e/ou propagação com **throw/throws**.

Blocos try/catch/finally

```
PreparedStatement stmt;
try {
    stmt = con.prepareStatement(query);
    //...
} catch (SQLException e) {
    throw new AcessoADadosException("Problema na criação do Statement");
} finally {
    stmt.close();
}
```

https://digitalinnovation.one/artigos/tratamento-de-excecoes-em-java

Cláusulas throw/throws

- As cláusulas throw e throws podem ser entendidas como ações que propagam exceções para um nível acima na pilha
- Em alguns momentos existem exceções que não podem ser tratadas no mesmo método que gerou a exceção.

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

Criando uma Exceção

```java
public class DivisaoNegativaException extends Exception {

    1 usage   Maicon Gerardi
    public DivisaoNegativaException(String mensagem){
        super(mensagem);
    }

}
```

Exercício #2

- Criar um programa que peça dois números inteiros ao usuário e faça uma divisão dos valores
- Caso ocorrer um InputMismatchException, informar ao usuário que ele informou valores não numéricos e repetir a operação
- Caso ocorrer um DivideByZeroException, informar ao usuário com a seguinte mensagem: “Não é possível dividir nenhum número por zero, tente novamente” e faça repetir a operação
- Se a divisão ocorrer com sucesso, apresente o resultado e ao final (na clausula finally) fechar o scanner e também apresentar a mensagem: “operação realizada com sucesso!”

Links Úteis

- https://www.oracle.com/br/technical-resources/article/java/erros-java-exceptions.html
- https://www.devmedia.com.br/tratando-excecoes-em-java/25514
- http://www.universidadejava.com.br/java/java-excecoes/
- https://www.alura.com.br/apostila-java-orientacao-objetos/excecoes-e-controle-de-erros

CRUD

- CREATE
- READ
- UPDATE
- DELETE

Homework Em Grupo

- Criar ao menos um enum para o seu projeto
- Validar diagrama de classes com o instrutor
- Criar uma exceção personalizada para o seu projeto e fazer uso de tratamento de exceções
- Criar ao menos 4 CRUDS (um para cada classe do seu projeto)