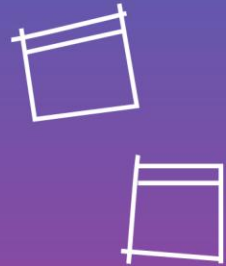




 **VEM SER**
DBC



Enums + Exceptions

Sumário

- Enumeradores
- Exceptions

Enums

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

Enums

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

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

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;
```

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".

Enums

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

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.

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.

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.

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)

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.

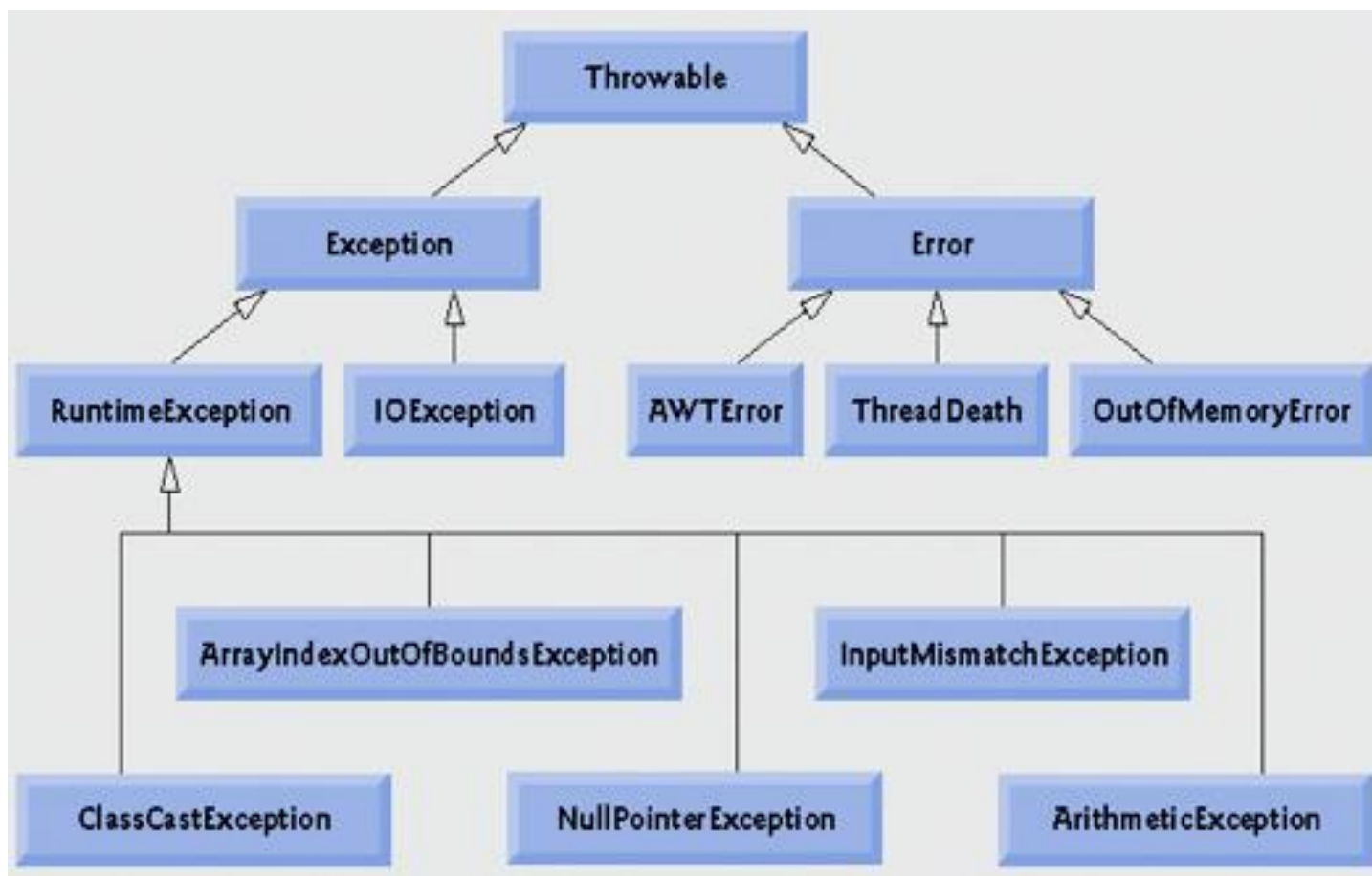
Trabalhando com Exceções

- 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.

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.

Hierarquia de Exceções



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();
}
  
```


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;
}
```

Exercício #2

- Criar um programa que peça dois números inteiros ao usuário e faça uma divisão dos valores
- Caso ocorrer um `NumberFormatException`, 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!”