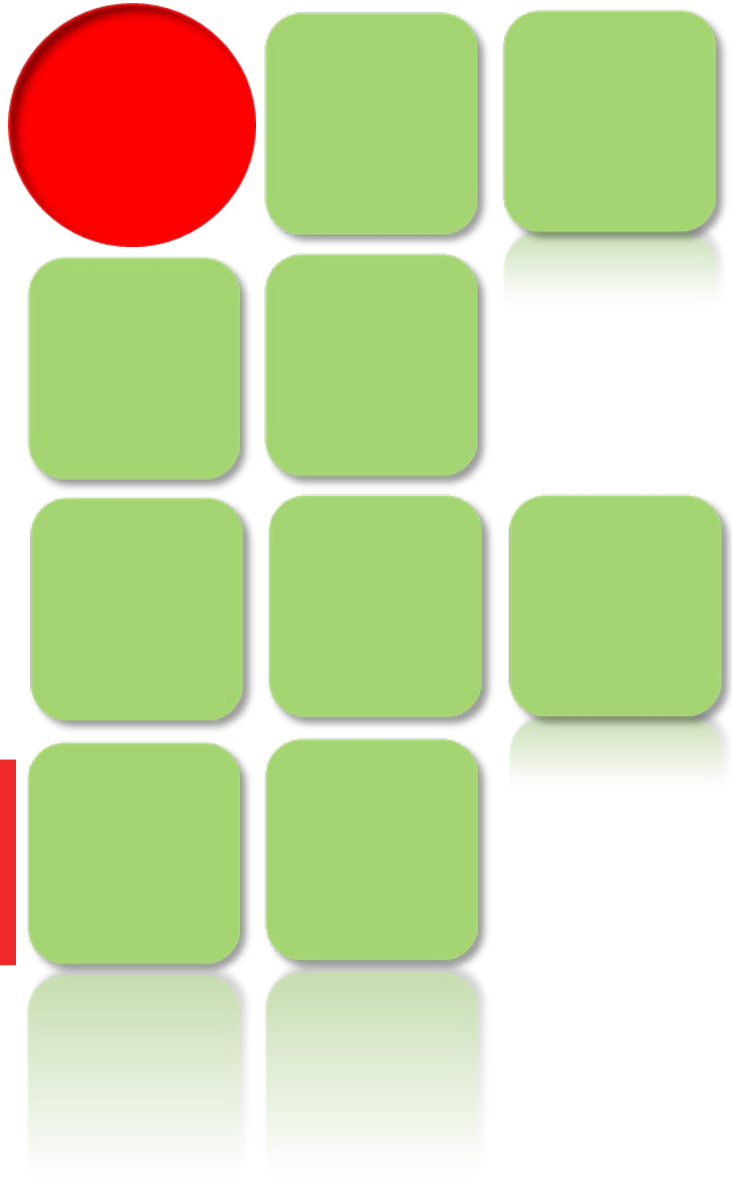

INSTITUTO FEDERAL

Prof. Dr. Bruno Queiroz Pinto

Expressão Regular

- Uma expressão regular é uma String formatada que define um padrão de pesquisa e substituição de textos.
- Permite realizar validações nos dados;
- Verifica se os dados estão em um determinado formato.
- Uma expressão consiste em caracteres e símbolos especiais.

Expressão Regular

- É necessário informar caracteres especiais usados no padrão da expressão.
- Indicam a ocorrência de números, letras entre outros caracteres no texto.

Caractere	Descrição	Metacaractere
.	Busca qualquer caractere	
\d	Busca qualquer número	[0-9]
\D	Busca qualquer caractere que não seja número	[^0-9]
\w	Busca qualquer caractere de letras e números	[a-zA-Z_0-9]
\W	Busca qualquer caractere que não sejam letras e números	[^\w]
\s	Busca qualquer caractere de espaço em branco, tabulações	[\t\n\x0B\f\r]
\S	Busca qualquer caractere sem espaço em branco	[^\s]

Método matches

- O método matches utiliza expressões regulares para verificar se um texto atende um padrão.

```
String texto = "Iftm";
System.out.println(texto.matches("Iftm"));
```

sem caracteres especiais

```
String texto = "Iftm 2022";
System.out.println(texto.matches("Iftm ….");
```

. → contém algum caractere

Aceita:

Iftm 2022
Iftm aaaa

Método matches

- Fazer o exemplo anterior aceitar apenas números

\dd → contém números

```
String texto = "Iftm 2022";
System.out.println(texto.matches("Iftm \\d\\d\\d\\d"));
```

Aceita:

Iftm 2022

~~Iftm aaaa~~

Método matches

- Exemplos de comparações simples

```
boolean email = "@ ".matches(".");
System.out.println("Possui um caractere: " + email);

boolean numero = "a".matches("\\d");
System.out.println("Possui um número: " + numero);

numero = "2".matches("\\d");
System.out.println("Possui um número: " + numero);

boolean letrasNumeros = "A2".matches("\\w\\d");
System.out.println("Possui uma letra e um número? " + letrasNumeros);

boolean espaco = " ".matches("\\s");
System.out.println("Possui um espaço? " + espaco);
```

Validar CPF:

```
String cep = "\\d\\d\\d\\d\\d-\\d\\d\\d";
boolean valida = "99855-000".matches(cep);
System.out.println(valida);
```

Quantificadores

- Fazer com que alguns caracteres se repitam, utilizar algum quantificador.
- O quantificador é um caractere que consegue informar quantas vezes um caractere pode ser repetido. **Onde X é o caractere especial.**

Expressão	Descrição
X{n}	X procura a ocorrência de um caractere n vezes
X{n,}	X pelo menos n vezes
X{n,m}	X pelo menos n mas não mais que m
X?	0 ou 1 vez
X*	0 ou mais vezes
X+	1 ou mais vezes
X{n}	X procura a ocorrência de um caractere n vezes

Quantificadores

- Exemplos simples de quantificadores

```java
// Existe 2 dígitos no texto
boolean valor = "74".matches("\\d{2}");
System.out.println(valor);

// Existe no mínimo 4 dígitos no texto
valor = "211".matches("\\d{4,}");
System.out.println(valor);

// existem entre 2 e 5 caracteres
valor = "2121".matches("\\d{2,5}");
System.out.println(valor);

// Existe 0 ou 1 caractere
valor = "22".matches(".?");
System.out.println(valor);

// existe entre 0 e vários caracteres
valor = "75411".matches(".*");
System.out.println(valor);

// existe entre 1 e vários caracteres
valor = "".matches(".+");
System.out.println(valor);

// Pesquisa se uma palavra existe no texto
String palavra = "Hello World Java";
valor = palavra.matches(".*Java.*");
System.out.println("palavra = " + valor);
```

Quantificadores

- Expressão regular de validação

```
// Cria expressão regular resumida da data
String data = "02/05/1995";
valor = data.matches("\\d{2}/\\d{2}/\\d{4}");
System.out.println("Data: " + valor);

// Cria a expressão regular resumida do cep
cep = "38545-222";
valor = cep.matches("\\d{5}-\\d{3}");
System.out.println("Cep: " + valor);

// Ocorrência de vários caracteres – validar CPF
String email = "iftm@iftm.com.br";
valor = email.matches(".+[@].+[.].+");
System.out.println("Email: " + valor);

//Validação de e-mail 2
boolean email = "java@teste.com.br".matches("\\w+@\\w+\\.\\w{2,3}\\.\\w{2,3}"
System.out.println(email);
```

[.] = ponto
\\. = ponto

Caracteres especiais de fronteira

- Esses caracteres definem se a String começa ou termina com um determinado padrão.

Metacaractere	Objetivo
* ^	Inicia
* $	Finaliza
* |	Ou (condição)

Caracteres especiais de fronteira

- Expressão regular

```
// E-mail finalizando com br
emailT = "iftm@iftm.br";
valor = emailT.matches(".+[@].+[.]br$");
System.out.println("Email: " + valor);

// Texto começando com IFTM
texto = "Iftm udi centro";
valor = texto.matches("^Iftm.*");
System.out.println("Texto: " + valor);

// Pesquisa pela palavra Uberlândia e derivados
palavra = "Iftm uberlândia centro";
valor = palavra.matches(".*Uberlândia.*|.*Uberlandia.*|.*uberlandia.*|.*uberlândia.*");
System.out.println("campus: " + valor);
```

Agrupadores

- Expressão regular (existem outros)

```
// Verifica se a frase começa com letras de a até e.
valor = "Da".matches("^[a-eA-E].*");
System.out.println(valor);

// (União) Verifica se começa com A até Z e contém letras a até z
valor = "Sql".matches("[A-Z][a-z]*");
System.out.println(valor);

// Não permite que comece com as letras a e i
valor = "Oracle".matches("[^aei]racle");
System.out.println(valor);
```

Fim aula 01.....

bruno.queiroz@iftm.edu.br