[Regex - Expressões Regulares]

```
package expressoes_regulares;

public class regex1 {

	public static void main(String[] args) {

		// aceita letras maiúsculas e minúscula em qualquer quantidade (0 ou muitos)
		// [A-Za-z] é a lista de caracteres aceitos
		// * é o quantificador 0 ou muitos
		String letras = "AbracaDabra";
		if (letras.matches("[A-Za-z]*"))
			System.out.println("letras válidas");
		else
			System.out.println("letras inválidas");

		// aceita apenas dígitos (pelo menos um ou muitos)
		// [0-9] é lista de dígitos aceitoas
		// + é o quantificador 1 ou muitos
		String digitos = "1927409377";
		if (digitos.matches("[0-9]+"))
			System.out.println("digitos válidos");
		else
			System.out.println("digitos inválidos");

		// aceita caracteres alfanuméricos (com 6 dígitos)
		String senha = "A7b3c8";
		if (senha.matches("[A-Za-z0-9]{6}"))
			System.out.println("senha válida");
		else
			System.out.println("Senha inválida");

		// OU |
		String animal1 = "gato";
		if (animal1.matches("gato|pato|rato"))    // gato, rato ou pato
			System.out.println("animal1 válido");
		else
			System.out.println("animal1 inválido");

		// lista + caracteres
		String animal2 = "gato";    // gato, rato ou pato
		if (animal2.matches("[grp]ato"))
			System.out.println("animal2 válido");
		else
			System.out.println("animal2 inválido");

		// [grp] = lista de caracteres aceitos na 1a posição
		// [a-z] = faixa de caracteres aceitos na 2a posição
		// {3} quantificador de [a-z] repetir para 3 posições
		String animal3 = "gato";   // g/p/r/xxx
		if (animal3.matches("[grp][a-z]{3}"))
			System.out.println("animal3 válido");
```

```
else
	System.out.println("animal3 inválido");

// ( )  grupo
// ? opcional
// minimercado, supermercado, hipermercado ou mercado
String mercado = "mercado";
// "(su|hi)permercado" ou "((su|hi)per)?mercado" ou "(mini|(su|hi)per)?mercado"
if (mercado.matches("(mini|super|hiper)?mercado"))
	System.out.println("mercado válido");
else
	System.out.println("mercado inválido");

// ( ) grupo com quantificador
String ai = "aiaiaiai";
if (ai.matches("(ai)+"))
	System.out.println("ai válido");
else
	System.out.println("ai inválido");

// validação de hora (00:00 até 29:59)
// solução em regex2.java
String hora = "20:22";
if (hora.matches("[012][0-9]:[0-5][0-9]"))
	System.out.println("hora válida");
else
	System.out.println("hora inválida");

// validação de data (00/00/0000 até 39/13/9999)
// solução em regex2.java
String data = "31/03/2014";
if (data.matches("[0-3][0-9]/[01][0-3]/[0-9]{4}"))
	System.out.println("data válida");
else
	System.out.println("data inválida");

// o curinga . (ponto aceita qualquer coisa)
// + aceita 1 ou mais digitos
String valor = "2,5";
if (valor.matches("[0-9]+.[0-9]+"))
	// válidos = 2,5 2.5 2x5 2a5 123,456
	System.out.println("valor válido");
Else
	// inválidos = ,5 2, ,
	System.out.println("valor inválido");

// OU |
String resposta1 = "sim";    // sim ou nao
if (resposta1.matches("sim|nao"))
	System.out.println("resposta1 válida");
else
	System.out.println("resposta1 inválida");

// OU |
String resposta2 = "não";   // sim ou (nao ou não)
if (resposta2.matches("sim|n[aã]o"))
	System.out.println("resposta2 válida");
else
	System.out.println("resposta2 inválida");

String palavra = "acalento";
if (palavra.matches(".*lento"))
	// válidos = lento, xxxlento
	System.out.println("palavra válida");
```

```
            else
                  System.out.println("palavra inválida"); // ento, acaxxxx

            // buscando vicente, aceita "etneciv", "eTneciV", "cevietn", ...
            String rua1 = "Avenida Vicente Machado, 123";
            if (rua1.matches(".*[VICENTEvicente]{7}.*"))
                  System.out.println("rua1 encontrada");
            else
                  System.out.println("rua1 não encontrada");

            // buscando vicente, aceita apenas "VICENTE" ou "vicente" (não aceita "Vicente")
            String rua2 = "Avenida Vicente Machado, 123";
            if (rua2.matches(".*(VICENTE|vicente).*"))
                  System.out.println("rua2 encontrada");
            else
                  System.out.println("rua2 não encontrada");

            String acentos = "água, acentuação e espaço";
            if (acentos.matches("[a-zçãá ,]+"))
                  System.out.println("acentos válidos");
            else
                  System.out.println("acentos inválidos");

            // (?i) ignora maiúscula/minúscula (case insensitive)
            String livro = "Livro";
            if (livro.matches("(?i)([a-z]*)"))
                  System.out.println("livro válido");
            else
                  System.out.println("livro inválido");
      }

}
```

```
package expressoes_regulares;

/* expressoes regulares com metacaracteres tipo barra-letra (posix)
 * \\d = dígito
 * \\D = não dígito
 * \\w = letras
 * \\W = não letras
 * \\s = espaço
 * \\S = não espaço
 *
*/
public class regex2 {

      public static void main(String[] args) {

            // valida somente números (+ = 1 ou muitos)
            String numeros = "123456";
            if (numeros.matches("\\d+"))  // (* = 0 ou muitos)
                  System.out.println("Números válidos");
            else
                  System.out.println("Números inválidos");

            // aceita letras, números e espaço (primeira deve ser maiúscula)
            String endereco = "Rua Santana 820";
            if (endereco.matches("[A-Z][\\w\\s]+"))
                  System.out.println("endereço válido");
            else
                  System.out.println("endereço inválido");
```

```
		// valida estado (apenas 2 letras)
		String uf = "PR";
		if (uf.matches("\\w{2}"))
			System.out.println("UF válido");
		else
			System.out.println("UF inválido");

		// valida o primeiro nome, no formato:
		// [A-Z]    = primeira letra deve ser maiúscula (sem espaço)
		// [a-zA-Z] = segunda letra pode ser minúscula ou maiúscula.
		// * = todos demais caracteres (opcionais), seguem padrão anterior (0 ou muitos)
		// Inválidos: "joao", "Joao da Silva", "Joao23"
		String nome = "Joao da Silva";
		// mínimo de 5 (1+4) e máximo de 20 caracteres
		if (nome.matches("[A-Z][\\w\\s]{4,19}"))
			System.out.println("nome válido");
		else
			System.out.println("nome inválido");


		// valida CEP
		// \\d = qualquer dígito (0..9)
		String cep = "84010-320";
		if (cep.matches("\\d{5}-\\d{3}"))
			System.out.println("cep válido");
		else
			System.out.println("cep inválido");

		// valida fone
		// [1-9] = qualquer número, menos zero.
		// \\d{3} = 3 dígitos numéricos
		String fone = "42-3028-0449";
		if (fone.matches("[1-9]\\d-[1-9]\\d{3}-\\d{4}"))
			System.out.println("fone válido");
		else
			System.out.println("fone inválido");

		// validação de hora (00:00 até 23:59)
		// solução de regex1.java
		String hora = "20:22";
		if (hora.matches("([01][0-9]|2[0-3]):[0-5][0-9]"))
			System.out.println("hora válida");
		else
			System.out.println("hora inválida");

		// validação de data (01/01/1000 até 31/12/2999)
		// solução de regex1.java
		String data = "31/03/2014";
		if (data.matches("(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])/[12][0-9]{3}"))
			System.out.println("data válida");
		else
			System.out.println("data inválida");

		// aceita somente consoantes e ignora maiúscula/minúsculas
		String consoantes = "bCdFghJkLmnPqrstvxywz";
		if (consoantes.matches("(?i)([a-z&&[^aeiou]]*)"))
			System.out.println("consoantes válidas");
		else
			System.out.println("consoantes inválidas");

	}
}
```

```java
package expressoes_regulares;
/*
 * POSIX são grupos de caracteres com padronização internacional
 *   POSIX           JAVA          JAVA barra-letra     DESCRIÇÃO
 * [:alpha:]      \\p{Alpha}                          Alfabeto
 * [:alnum:]      \\p{Alnum}       \\w                Alfanumérico
 * [:digit:]      \\p{Digit}       \\d                Dígitos (números)
 * [:lower:]      \\p{Lower}                          Minúsculas
 * [:upper:]      \\p{Upper}                          Maiúsculas
 * [:space:]      \\p{Space}       \\s                Espaço  (\t\r\n\v\f)
 * [:punct:]      \\p{Punct}                          Pontuação e símbolos
                                                     (.,;<=>!?@#$%&()-+/\\^~)
 */

// OBS.: Embora o padrão POSIX inclua acentuação em português, não está implementado no Java!!
public class posix {

	public static void main(String[] args) {

		String cidade = "Ponta Grossa";
		if (cidade.matches("[\\p{Alpha}\\s]{3,20}"))
			System.out.println("cidade válida");
		else
			System.out.println("cidade inválida");

		String rua = "Rua Santana, 820";
		if (rua.matches("[\\p{Alnum},\\s]{3,20}"))
			System.out.println("rua válida");
		else
			System.out.println("rua inválida");

		String numero = "1234";
		if (numero.matches("\\p{Digit}{1,4}"))
			System.out.println("número válido");
		else
			System.out.println("número inválido");

	}

}
```

Tabela POSIX

POSIX	Description	ASCII	Unicode	Shorthand	Java
`[:alnum:]`	Alphanumeric characters	`[a-zA-Z0-9]`	`[\p{L&}\p{Nd}]`		`\p{Alnum}`
`[:alpha:]`	Alphabetic characters	`[a-zA-Z]`	`\p{L&}`		`\p{Alpha}`
`[:ascii:]`	ASCII characters	`[\x00-\x7F]`	`\p{InBasicLatin}`		`\p{ASCII}`
`[:blank:]`	Space and tab	`[ \t]`	`[\p{Zs}\t]`	`\h`	`\p{Blank}`
`[:cntrl:]`	Control characters	`[\x00-\x1F\x7F]`	`\p{Cc}`		`\p{Cntrl}`
`[:digit:]`	Digits	`[0-9]`	`\p{Nd}`	`\d`	`\p{Digit}`
`[:graph:]`	Visible characters (i.e. anything except spaces, control characters, etc.)	`[\x21-\x7E]`	`[^\p{Z}\p{C}]`		`\p{Graph}`
`[:lower:]`	Lowercase letters	`[a-z]`	`\p{Ll}`		`\p{Lower}`
`[:print:]`	Visible characters and spaces (i.e. anything except control characters, etc.)	`[\x20-\x7E]`	`\P{C}`		`\p{Print}`
`[:punct:]`	Punctuation and symbols.	`[!"#$%&'()*+, \-./:;<=>?@ [\\\]^_`{|}~]`	`[\p{P}\p{S}]`		`\p{Punct}`
`[:space:]`	All whitespace characters, including line breaks	`[ \t\r\n\v\f]`	`[\p{Z}\t\r\n\v\f]`	`\s`	`\p{Space}`
`[:upper:]`	Uppercase letters	`[A-Z]`	`\p{Lu}`		`\p{Upper}`
`[:word:]`	Word characters (letters, numbers and underscores)	`[A-Za-z0-9_]`	`[\p{L}\p{N}\p{Pc}]`	`\w`	
`[:xdigit:]`	Hexadecimal digits	`[A-Fa-f0-9]`	`[A-Fa-f0-9]`		`\p{XDigit}`