

## Ficha de Trabalho nº 3

### Expressões Regulares, *Wrappers*

#### 1. Bibliografia

<http://www.regular-expressions.info/java.html>

<http://www.regular-expressions.info/reference.html>

#### 2. Introdução às ER em Java

Neste ficha de trabalho pretende-se que os alunos explorem a utilização de expressões regulares (ER) em Java. Além das ERs abordadas nas aulas teóricas, a tabela seguinte sintetiza algumas expressões que podem ser usadas num programa em Java:

Expressão regular	Descrição
<code>\d</code>	Qualquer dígito, equivalente a <code>[0-9]</code>
<code>\D</code>	Um não-dígito, equivalente a <code>[^0-9]</code>
<code>\s</code>	Um carácter de espaçamento, equivalente a <code>[\t\n\r\f]</code>
<code>\S</code>	Um carácter que não é de espaçamento, equivalente a <code>[^\s]</code>
<code>\w</code>	Caracter alfanumérico, equivalente a <code>[a-zA-Z_0-9]</code>
<code>\W</code>	Um carácter não alfanumérico, equivalente a <code>[^\w]</code>

NOTA 1: Colocar **(?i)** no início de uma ER torna-a ‘case insensitive’

NOTA 2: O carácter `\` é um carácter de escape usado nas strings em Java. Se se pretender usar este carácter numa ER, este tem aparecer duplicado. Por exemplo, usar `"\"` define um único carácter `"`. Para usar `\w` numa ER, tem de se escrever `\\w`, para usar `\d`, escreva `\\d`, etc.

##### 2.1.Classes Pattern and Matcher

Para uso avançado de ER devem ser usadas as classes `java.util.regex.Pattern` e `java.util.regex.Matcher`

Deve usar-se primeiro a classe `Pattern` para definir a expressão regular. O objeto criado com a classe `Pattern` permite criar um objeto `Matcher` para uma dada string. Este objeto `Matcher` permite executar métodos regex na string.

A descrição dos métodos destas classes pode ser consultada em:

<http://docs.oracle.com/javase/tutorial/essential/regex/matcher.html>)

As etapas para a criação da ER e a procura do respetivo padrão numa fonte de dados são as seguinte:

- 1) Criar a expressão regular numa String:  
`String er = "[a-zA-Z]+";`
- 2) Ler o texto onde procurar os padrões da ER:  
`String texto = "A Maria tem 12 anos";`
- 3) Criar variável da classe Pattern para compilar a ER:  
`Pattern p = Pattern.compile(er);`
- 4) Criar variável do tipo Matcher para ligar a ER ao texto:  
`Matcher procura = p.matcher(texto);`
- 5) Percorrer o Matcher e verificar as ocorrências encontradas:  

```
while (procura.find()){  
    System.out.println(procura.group());  
}
```

### Exemplo:

```
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class TestarER {  
    public static void main(String[] args) {  
  
        String linha = "A linha 1 vem antes da linha 2 no meio de 100 linhas";  
  
        //ER para encontrar os números na String linha  
        String er="[0-9]+"; // ou \\d+  
        Pattern p = Pattern.compile(er);  
        Matcher m = p.matcher(linha);  
        while(m.find()){  
            System.out.println("Numero : " + m.group() + "\\n");  
        }  
    }  
}
```

## 3. Tarefa 1

Crie um projeto Java de nome **Ficha3**.

- a) Implemente uma função **static void ficha3a()** que:
  - inicialize uma variável da classe String com vários números de telefone (móveis e fixos)  
`String telef="919191919 929992221 9111111111 239494582 9199999999 967779999";`
  - inicialize uma String com uma ER que valide apenas números de telemóveis
    - começam por 91, 92, 93, 96 e têm 9 dígitos de comprimento
    - apenas os números marcados a azul devem ser encontrados pela ER
  - Utilize as Classes **Pattern** e **Matcher** para imprimir na consola a lista de telemóveis válidos encontrados.
  - Chame a função no *main* e verifique se está correta.

b) Implemente uma função **static void ficha3b(String fileIn, String fileOut)** que:

- Inicialize uma String com uma ER que valide datas no formato **dd-mm-aaaa** ou **dd/mm/aaaa**
- Utilize as Classes *Pattern* e *Matcher*
- Leia o ficheiro **fileIn** linha a linha
- Em cada linha lida, verifique se encontra uma data válida. Se encontrar, escreva no ficheiro **fileOut** a lista de datas válidas encontradas:
  - 31-03-2002 - Data Valida
  - 04/12/2007 - Data Valida
  - 01-01-2005 - Data Valida
  - 01/03/2007 - Data Valida
- Chame a função no **main** com os argumentos “datas.txt” e “out.txt” e verifique o conteúdo de **out1.txt**

c) Implemente uma função **static void ficha3c(String fileIn, String fileOut)** que:

- Escreva uma ER que encontre palavras que tenham a sequência de caracteres **ch**
- Leia o ficheiro **fileIn** linha a linha
  - Substitua nessas palavras **ch** por **X** >> *achou* ficará *aXou*
  - Escreva as linhas alteradas para **fileOut**
  - Conte o número de substituições feitas
  - No final da leitura escreva no ficheiro de saída: “*Foram efetuadas ..... substituições*”
- Utilize as Classes *Pattern* e *Matcher*
- Use o método **replace** da classe String e o método **group** para substituir apenas o **ch** mantendo o resto da palavra. Deve criar um grupo na ER de forma a indexá-lo corretamente.
- Chame a função no **main** com os argumentos “ficheiro3.txt” e “out2.txt” e verifique o conteúdo de **out2.txt**

## 4. Tarefa 2

**Wrappers:** funções que recebem uma expressão de pesquisa, procuram no ficheiro fonte usando ER e devolvem o resultado pretendido.

Crie um novo ficheiro java **Wrappers.java** implemente os *wrappers* para extrair informação do ficheiro **peessoas.html**. O objetivo desta ficha é construir os *wrappers* que procuram informação isolada nas fontes de dados, conseguindo descobrir toda a informação de uma determinada pessoa (nome, morada, profissão, ...).

Deve analisar a estrutura do ficheiro html e implementar as ERs adequadas a encontrar a informação pretendida.

a) Implemente uma função **static String procura\_nome\_por\_id(String procura)**

que receba do utilizador um identificador de pessoa e **procura o nome dessa pessoa** no ficheiro **peessoas.html**. A função devolve uma String com o nome completo encontrado, ou no caso de não existir, devolve **null**

*Teste a função no main: peça a palavra de pesquisa ao utilizador, chame a função e mostre o resultado*

- b) Implemente uma função **static String procura\_cidade\_por\_id(String procura)**

que receba um identificador de pessoa e procura a cidade onde vive essa pessoa no ficheiro `peessoas.html`. A função devolve uma String com a morada encontrada, ou no caso de não existir, devolve **null**

*Teste a função no main: peça a palavra de pesquisa ao utilizador, chame a função e mostre o resultado*

- c) Implemente uma função **static String procura\_profissao\_por\_id(String procura)**

que receba um identificador de pessoa e procura a profissão dessa pessoa no ficheiro `peessoas.html`. A função devolve uma String com a profissão encontrada, ou no caso de não existir, devolve **null**

*Teste a função no main: peça a palavra de pesquisa ao utilizador, chame a função e mostre o resultado*

- d) Implemente uma função **ArrayList procura\_nomes(String procura)**

que receba uma palavra de pesquisa (nome ou parte de um nome) e procure esse nome no ficheiro **peessoas.html**. A função devolve a lista de nomes completos onde a expressão de pesquisa foi encontrada. A palavra de pesquisa deve ser solicitada ao utilizador.

Exemplo: Pesquisar “**Ana**” deve devolver *Ana Martins* e *Anabela Lopes*

*Teste a função no main: peça a palavra de pesquisa ao utilizador, chame a função e mostre o resultado*

```
Scanner ler = new Scanner(System.in);
String palavra;
System.out.println("Palavra a procurar: ");
palavra = ler.nextLine();
ArrayList res = Wrappers.procura_nomes(palavra);
System.out.println(res);
```

- e) Crie uma Classe **Pessoa** com os campos  
**String id, nome, morada, profissão;**

Crie o construtor completo, os *getters* e os *setters*. Use a opção **Source – Insert Code - ...**

Implemente um método **imprime** para imprimir na consola os atributos de uma Pessoa.

- f) Implemente uma função **static Pessoa procura\_dados\_pessoa(String id)** que devolva uma instância da classe Pessoa com o **id** introduzido. Deve chamar algumas das funções anteriores para procurar a informação nos ficheiros correspondentes e enviar essa informação para o construtor.

## 5. Tarefa 3

### Wrappers: extrair dados da web

Suponha que o ficheiro `peessoas.html` não está local no seu disco, mas sim na web.

Precisa de fazer um `HttpRequest` ao link para obter a informação desejada. No Moodle encontra um ficheiro com a implementação desta função:

- **HttpRequestFunctions.java**: para aceder aos dados/source de um determinado site

Existem duas funções, onde a única diferença é o separador usado na codificação do espaço em branco (+ ou \_).

Quando existe uma palavra de pesquisa para a criação do URL a aceder, use a função com duas strings como argumento. Caso não haja palavra de pesquisa, envie a String "" no 2º argumento. Assim garante que a codificação de todos os caracteres é feita corretamente e é gerado um URL válido que pode ser acedido pela função.

Copie o ficheiro **HttpRequestFunctions.java** para a pasta **src/ficha3** do Projeto da Ficha3 desenvolvido nas aulas anteriores. Corrija eventuais erros (nome da *package*).

Elimine o ficheiro **peessoas.html** da pasta do projeto. O ficheiro **peessoas.html** usado nos wrappers implementados passou agora a estar disponibilizado no site: <https://eden.dei.uc.pt/~abs/ID/peessoas.html>

Aos Wrappers implementados acrescente a chamada à função **HttpRequest1** para que o *source* do ficheiro seja temporariamente armazenado em disco antes da chamada dos *Wrappers*. Quando gravar o *source* em disco, mantenha o nome do ficheiro antigo (**peessoas.html**)

```
String link = "https://eden.dei.uc.pt/~abs/ID/peessoas.html";  
String pesquisa = "";  
HttpRequestFunctions.httpRequest1(link, pesquisa, "peessoas.html");
```

Teste os *wrappers* e verifique se a informação é extraída da Web corretamente.