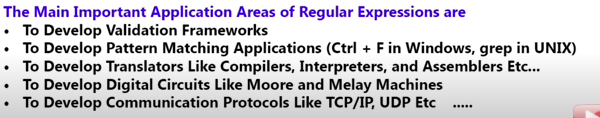
**Regular Expressions**

It is a representation to represent a group of string objects according to a particular pattern.

If we want to represent a group of strings according to a particular pattern, then we should go for Regular Expression.

Ex: We can write a Regular Expression to Represent all mob numbers.

Ex: We can write a Regular Expression to Represent all Mail ID’s.



Ex:

import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
public class Main {  
 public static void main(String[] args) {  
 int count = 0;  
 Pattern p = Pattern.*compile*("ab");  
 Matcher m = p.matcher("ababbaba");  
 while(m.find()){  
 count++;  
 System.*out*.println("Start index: "+m.start()+" "+"End index: "+m.end()+ "...." +m.group());  
 }  
 System.*out*.println("The total number of occurrence : "+count);  
 }  
}

O/P:

Start index: 0 End index: 2....ab

Start index: 2 End index: 4....ab

Start index: 5 End index: 7....ab

The total number of occurrence : 3

**Pattern:**

* A pattern object is a compiled version of regular expression.
* It is equivalent java object of regular expression.
* We can create a pattern object by using compile() of Pattern class.

Syntax: public static Pattern compile(String regularExpression)

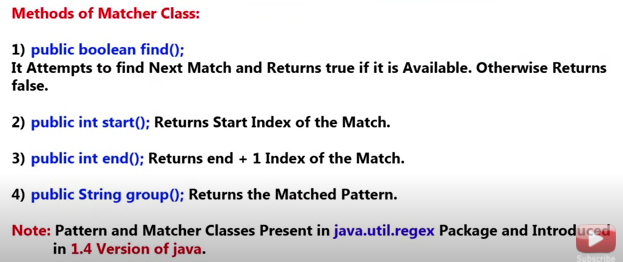
Ex: Pattern p = Pattern.compile(“ab”);

**Note: Static factory method:** By using class name if are calling a method and that method returns the same class type such type method.

Ex: Pattern.compile()

**Matcher:**

* We can use Macher object to match the given pattern in the target string.
* We can Matcher object by using matcher method of Pattern class.
  + public Matcher matcher(String target);
* Ex: Matcher m = p.matcher(“ababbaba”);



**Character class:**

If we want to go for multiple characters at a time then we have to go for character classes.

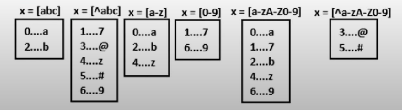
Several character classes are given below:

|  |  |
| --- | --- |
| [abc] | Either a OR b OR c |
| [^abc] | Except a, b, c |
| [a-z] | Any lower case alphabet symbol. |
| [A-Z] | Any upper case alphabet symbol. |
| [0-9] | Any Digit from 0 to 9 |
| [a-zA-Z0-9] | Any Alpha Numeric Symbol |
| [^a-zA-Z0-9] | Except Alpha Numeric symbol(special Character) |

Ex: we will see the output for different values of x;

public static void main(String[] args) {  
Pattern p = Pattern.*compile*("x");  
Matcher m = p.matcher("a7b@z#9");  
 while (m.find()){  
 System.*out*.println(m.start()+ "...." + m.group());  
 }  
 }

O/P:



**Pre-defined character classes:**

There are some predefined character classes we can directly use.

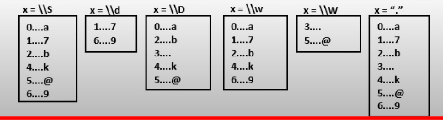
|  |  |
| --- | --- |
| **\s** | Space character |
| **\S** | Any character except space |
| **\d** | Any digit form [0-9] |
| **\D** | Any character except digit |
| **\w** | Any word character (Any alphanumeric character) [a-zA-Z0-9] |
| **\W** | Except word Character (Special Character) |
| **.** | Any symbol including Special character also |

**Note:** if we will use above classes with one backslash then we will get compile time error because \s or \d are not recognized by compiler as symbols so we have to use **double backslash (Ex:** [**\\s**](file:///\\s)**,** [**\\S**](file:///\\S)**,** [**\\d**](file:///\\d)**, etc).**

Code: for different values of x o/p are given below.

public static void main(String[] args) {  
Pattern p = Pattern.*compile*("x");  
Matcher m = p.matcher("a7b k@9");  
 while (m.find()){  
 System.*out*.println(m.start()+ " ... " + m.group());  
 }  
 }

O/P:

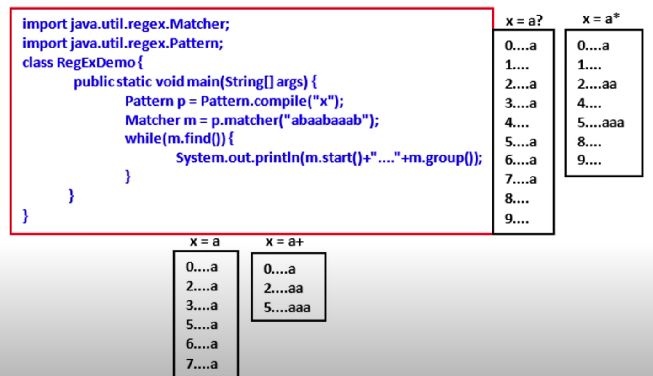


**Quantifiers:**

We can use to quantifiers to specify number of occurrences of a symbol to match.

|  |  |
| --- | --- |
| A | Exactly one a |
| a+ | At least one a |
| a\* | Any number of a’s including zero number also |
| a? |  |

Note : in case of \* quantifier we are going to search for lastindex+1 position which is empty .



**Pattern class split() method:**

It is used to split the target string according a particular pattern we have to go for pattern class split method.

Ex:

public static void main(String[] args) {  
 Pattern p = Pattern.*compile*("\\s");  
 String[] s = p.split("Durga soft solution");  
 for (String s1:s){  
 System.*out*.println(s1);  
 }  
}

O/P: it will split in the space 🡪 \\s

Durga

soft

solution

Ex: 2

public static void main(String[] args) {  
 Pattern p = Pattern.*compile*("\\."); // the . is represented by \\. or [.]  
 String[] s = p.split("www.google.com");  
 for (String s1:s){  
 System.*out*.println(s1);  
 }  
}

o/p:

www

google

com

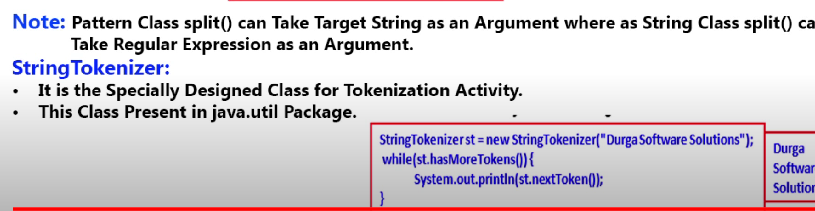
**string class split() method:**

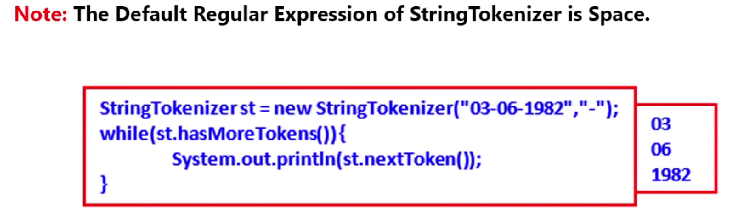
We can do the above thing by string class split() method also.

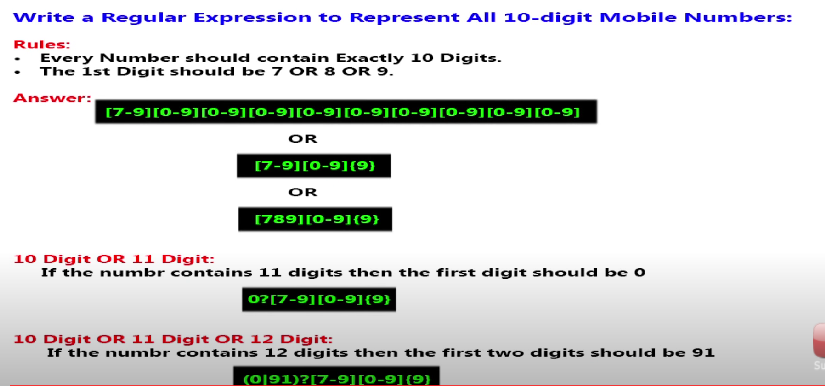
It take pattern as argument.

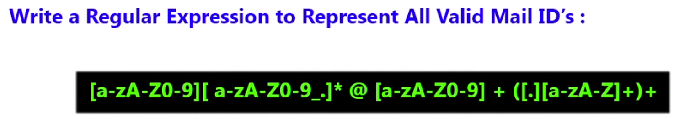
Ex:

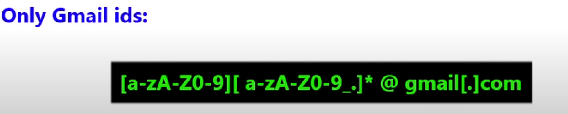
String S = "Durga soft solution";  
String[] S2 = S.split("\\s");  
for (String s1:S2){  
 System.*out*.println(s1);  
}

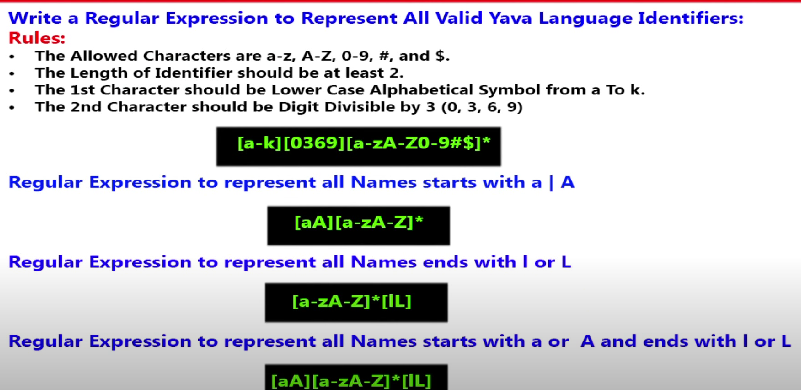












Q. WAP to check whether the provided email id and mobile number is valid or not.

package com.company;  
  
import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Demo\_RegEx {  
 //WAP to check whether provided number and email id are valid or not.  
 //Regular expression to represent a mobile number according to India  
 // mobile no - 10 digit, first digit 6 or 7 or 8 or 9 Ex: 9182736450  
 // Regular Expression = [6-9][0-9]{9}  
  
 // if length = 11 then first number must be 0 Ex: 07372942863  
 //Regular Expression = 0?[6-9][0-9]{9}  
  
 // if length = 12 then first two digit should be 91 Ex: 918972942863  
 //Regular Expression = (0/91)?[6789][0-9]{9}  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter mobile number: ");  
 String mobile = sc.nextLine();  
 System.*out*.print("\nEnter Email Id: ");  
 String email = sc.nextLine();  
  
 Pattern p1 = Pattern.*compile*("(0/91)?[6**-**9][0**-**9]{9}");  
 Matcher m1 = p1.matcher(mobile);  
  
 Pattern p2 = Pattern.*compile*("[a**-**zA**-**Z0**-**9][a**-**zA**-**Z0**-**9\_.]\*@[a**-**zA**-**Z0**-**9]+([.][a**-**zA**-**Z]+)+");  
 Matcher m2 = p2.matcher(email);  
  
 if (m1.find() && m1.group().equals(mobile)){  
 System.*out*.println("Valid Mobile number!!!");  
 }else  
 System.*out*.println("Invalid Mobile number!!!");  
  
 if (m2.find() && m2.group().equals(email)){  
 System.*out*.println("Valid Mail Id!!!");  
 }else  
 System.*out*.println("Invalid Mail Id!!!");  
  
 }  
}

o/p:

Enter mobile number: 6372942863

Enter Email Id: mzzhimansu@gmail.com

Valid Mobile number!!!

Valid Mail Id!!!

Q. WAP to extract all the mobile number from input.txt to output.txt!

package com.company;  
  
import java.io.\*;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class ExtractMobileNumbers {  
 //WAP to extract mobile numbers present in input.txt files  
 // where mobiles number are mixed with normal text data.  
 public static void main(String[] args) throws IOException {  
 Pattern p = Pattern.*compile*("(0/91)?[6**-**9][0**-**9]{9}");  
 PrintWriter pw = new PrintWriter("output.txt");  
 BufferedReader br = new BufferedReader(new FileReader("input.txt"));  
 String line = br.readLine();  
 while(line != null){  
 Matcher m = p.matcher(line);  
 while(m.find()){  
 pw.println(m.group());  
 }  
 line = br.readLine();  
 }  
 pw.flush();  
 System.*out*.println("Task Done!");  
 }  
}

Q. WAP to display all .txt filename present in C:\Users\mzzhi classes.

package com.company;  
  
import java.io.File;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class FindTxtFile {  
 public static void main(String[] args) {  
 int count=0;  
 Pattern p = Pattern.*compile*("[a**-**zA**-**Z0**-**9][a**-**zA**-**Z0**-**9\_$.]\*[.]txt");  
 File f = new File("C:\\Users\\mzzhi");  
 String[] s = f.list();  
  
 for(String s1:s){  
 Matcher m = p.matcher(s1);  
 if (m.find() && m.group().equals(s1)){  
 count++;  
 System.*out*.println(s1);  
 }  
 }  
 System.*out*.println(count);  
 }  
}