Regular Expression: A group of string objects in particular pattern is regex

* Write regex to represent some pattern

To see which pattern create the pattern object.

Pattern object is the compiled version of regex

Pattern p=Pattern.compile(“ab”);

**Public static pattern compile(string regularexp)**

In which target we need to match the pattern:

Matcher will match the pattern from the target

Matcher m=p.matcher(“abbababba”);

Public matcher matcher(String target)

Int count=0;

While(m.find()){

Count++;

Sysout(m.start()+”…”+m.end()+”…”+m.group());

}

Sysout(count);

m.start()---starting index where the pattern is matched

m.end()---end index+1 when the pattern ends

m.group()---what got matched in the target pattern

regex:

[abc]— a or b or c

[^abc]---except a or b or c

[a-z]—any lower case alphabets

[A-Z]---any upper case alphabets

[a-zA-Z]---any alphabet

[0-9]—any number between 0-9

[^0-9]—any alphabets n spcl char

[^a-zA-Z0-9]—any spcl characters

**Pre-defined character classes:**

\s—space

\S—except Space

\d-any digit from 0-9

\D—except digit any character

\w—Any word character(a-zA-Z0-9)

\W—except word character,spcl character

.—any character

Exception when wegive single ‘\’ illegal escape character. So have to take \\ in java

**Quantifiers**: This is to specify number of occurences to match.

Pattern p=pattern.compile(“x”);

Matcher m=p.matcher(“abbaabbbaaa”);

While(m.find()){

Sysout(m.start()+”…”+m.group());

}

If x=a

Here we get the output wherever a is present.

o/p: 0—a 3—a 4—a 8—a 9—a 10—a

if x=a+

here atleast one a is required to match.

o/p: 0—a 3—aa 8—aaa

if x=a\*

here any nmber of a’s including zero number

o/p: 0—a 1—2—3—aa 5—6—7—8—aaa 9--

if x=a?

atmost one a. one a or zero number of a

o/p: 0—a 1—2—3—a 4—a 5—6—7—8—a 9—a 10—a 11--

In regex, cursor can move end+1 location

**Pattern class Split method:**

Pattern p=pattern.compile(“\\s”);

String[] s=p.split(“Divya Polaki”);

For(String s1:s){

Sysout(s1)

}

o/p: Divya

Polaki

Pattern p=pattern.compile(“a”);

String[] s=p.split(“Divya Polaki”);

For(String s1:s){

Sysout(s1)

}

o/p:

Divy

Pol

Ki

Pattern p=pattern.compile(“.”);

String[] s=p.split(“Divya.Polaki”);

For(String s1:s){

Sysout(s1)

}

Here, except all charcaters the split has to happen. So no o/p. because “.” Is considered as all char in regex.

Pattern p=pattern.compile(“\\.”);//we can also use [.] instead of \\.

String[] s=p.split(“Divya.Polaki”);

For(String s1:s){

Sysout(s1)

}

o/p:

Divya

Polaki

**String class split method:**

String a=”Divya Polaki”;

String[] s=a.split(“\\s”);

For(String s1:s){

Sysout(s1)}

o/p:

Divya

Polaki

In pattern class split method, target string is used as argument whereas in string class split method, pattern is used as argument.

StringTokenier s=new StringTokenizer(“Divya Polaki”);

While(s.hasmoreTokens()){

Sysout(s.nextToken())

}

o/p: Divya

Polaki

Here, the default pattern in stringTokenzer is space

StringTokenier s=new StringTokenizer(“Divya-Polaki”,”-”);here we can specify tarhet string and also the delimiter or pattern

While(s.hasmoreTokens()){

Sysout(s.nextToken())

}

o/p: Divya

Polaki

Phone number regex:

(0|91)?[789][0-9]{9}

First digit either 0 or 91

1st digit in phn number ie either 7 or 8 or 9

Next 9 digits from 0-9

Email Regex:

[a-zA-Z0-9][a-zA-Z0-9\_\*][\*@[0-9a-zA-Z]+([.][0-9a-zA-Z]+)](mailto:*@[0-9a-zA-Z]+([.][0-9a-zA-Z]+))+

For gmail id

[a-zA-Z0-9][a-zA-Z0-9\_\*][\*@gmail[.]com](mailto:*@gmail[.]com)

For yahoo  
[a-zA-Z0-9][a-zA-Z0-9\_\*][\*@yahpp[.]co[.]in](mailto:*@yahpp[.]co[.]in)