Regular Expression

->Any group of string according to a particular pattern is called regular expression

Ex.(1) We can write a regular expression to represent all valid mail-ids and by using that regular expression we can validate whether the given mail-id is valid or not.

(2) We can write a regular expression to represent all valid java identifiers.

->the main application arises of regular expression are

1. We can implements validation mechanism.

2.We can develop pattern matching applications.

3.We can develop translators like compilers interpreters etc

4.we can use for designing digital circuits.

5.we can use to develop communication protocols like TCP by Ip,UDPetc

```
Ex. Import java.util.reger.*;
Class RegDemo
{
p.s.v.main(String args[])
{
Pattern p=pattern.compile("ab");
Master m=p.master("abbbabbcbdab");
While(m.find())
{
s.o.pln(m.start()+"…….”+m.end() +"………"+m.group());
}
}
}
```

o/p-

```
0….2….ab
4….6….ab
10…..12….ab
```

Pattern class:-

- ➔ A pattern objects represents compiled version of regular expression, we can create a pattern object by using compile() of pattern class.
- ➔ Pattern p=pattern.compile(String RegularExpression);

Matches class:-

->A matches object can be used to match character sequence against a regular expression. We can create a matches object by using matches() of pattern class.

Matches m=p.matches(String target);

Important methods of matches class:-

(1)Boolean find():-

->It attempts to find the next match and if it is available returns true otherwise returns false.

(II)int start():-

->Returns start index of the match.

(III)int end():-

->returns end index of the match.

(Iv) String group:-

->returns the match pattern.

Character classes:-

(1)[a-z]:--Any lower case alphabets symbol

(2)[A-Z]:--any upper case alphabet symbol.

(3)[a-z A-Z]:--Any alphabets symbols

(4)[0-9]:--any digit from 0 to 9.

(5)[abc]:--either a or b or c.

(6)[^abc]:--except a or b or c

(7)[0-9 a-z A-Z]:--any alpha numeric character

Ex.

```
        Pattern p=pattern.compile("x");
Matches in=p.matches("a3b@c4z#");
While(m.find())
{
s.o.pln(m.start()+"………"+m.group());
}
```

X=[ab]	x=[a-z]		x=[0-9]	x=[0-9a-z]
0….a	0…a	1….3	0…..a	
2….b	2….b	5….4	1…..3	
	4…c		2….4	
	6….z		4…..c	

Predefined character class:-

Space character -------→\s

[0-9]------------------------→\d

[0-9 a-z A-Z]-------------→\w

Any character-----------→ .

Ex. Pattern p=pattern.compile("x")

```
Matches m=p.matches("a3z4 @   k7#")
                     0123456789
```

While(m.find())

{

s.o.pln(m.start() +"-------"+m.groups());

}

X=\\d	x=\\w	x=\\s	x=
1…3	0…a	5…	0…a
3….4	1….3		1….3
7…..7	2…..z		2….z
	3…..4		3….4
	6….k		4---@
	7…7		5---
			6---k
			7---1
			8---#

Quantifiers:-

->We can use quantifiers to specify no of charactera to match

Ex.

1)a->exactly one a

2) a+->atleast one a

3) a* ->Any no of air

4) a?->atmost one a

```
Ex. pattern p=pattern.compile("x");
Matcher m=p.matcher("abaabaaab");
While(m.find())
{
s.o.pln(m.start() +"……."+m.group());
}
```

X=a	x=a+	a=a*	x=a?
0….a	0…a	0….a	0…..a
2….a	2…aa	1…..	1……
3…..a	5….aaa	2…aa	2….a
5…..a		4…..	3…..a
6…..a		5….aaa	4…….
7……a		8…….	5….a
		9……	6……a
			7……a
			8……..
			9……..

Split Method():-

Pattern class contains split method to split given string according to a given expression.

```
Ex. Pattern p=pattern.compile("\s");
String[] s=p.split("employee");
For(string s1==s)
{
s.o.pln(s1);



}
```

Ex(2).

```
Pattern p=pattern.compile("\\.");          [.]
String[]  s=p.split("example one");
for(string s1=s)
{
s.o.pln(s1)
}
```

o/p

```
example
one
```

String class split() method:-

->String class also contains to split the given string against a regular expression.

Ex.

```
String s=”abc.xy”;
String[] s1=s.split(“\\.”);
For (string s2=s1)
{
S.o.pln(s2);
}
```

o/p:

```
    abc
     xy
```

Note:-

Pattern class split() can take target string as arguments where as string class split() can take regular expression as arguments.

StringTokenizer:-

->We can use StringTokenizer to divide the target string into system of tokens according to the stringTokenizer class presenting in java.util package.

Ex.

```
StringTokenizerst=new StringTokenizer(“durga software solutions”);
While(st.hasMoreTokens())
{
s.o.pln(st.nextToken());
}
```

o/p

durga

software

solutions

Note:- The defaukt regular expression is space.

(2)StringTokenizerst=new StringTokenizer("1,00,000",",");

While(st.hasMoreToken());

}

o/p:

1

00

000

Ex.

Write a regular expression to represent the set of all valid identifiers in java language.

Rules:

(1)The length of each identifiers is atleast 2

(2)The allowed characters are….

a to z

A to Z

0 to 9

(3) The first characters should not digit

Rule.

[a-z A-Z…][a-zA-Z 0-9][a-zA-Z 0-9….]*

[a-z A-Z…][a-zA-Z 0-9][a-zA-Z 0-9….]+

```
importjava.util.regx.*;
class RegExDemo2
{
p.s.v.main(String[] args)
{
Pattern p=pattern.compile[“[a-z A-Z…][a-z A-Z 0-9][a-z A-Z 0-9….]+”];
Matches m=p.mathces(args[0]);
if(m.find()&&m.group().equals(args[0]))
{
s.o.pln(“valid Identifier”);
}
else
{
S.o.pln(“”Invalid identifiers);
}
}
}
```

(2) W.A.P to represent all valid mobile numbers….

Rule:-

(1) Mobile no contains 10 digits
(2) The first digits should be 7 to 9

[7-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-8] [0-9][0-9][0-9][0-9]

(3) W.a. regular expression to represent all valid mail-ids.

rule:-
(1)The set of allowed characters in mail-id are 0 t0 9,a-z,A-Z
(2) Should starts with alphabets symbols.
(3)Should contains atleast one symbols.

RegExp:-
[a-z A-Z][a-z A-Z 0-9--]* (a)[a-z A-Z 0-9]+ ([.][a-z A-Z+]+
RegExp:-
--||-- (a)gmail[.]com
--||-- (a)(gmail|yahoo\hotmail)[.]com

Ex.
```
import java.io.*;
importjava.util.regex.*;
classmobileExtractor
{
p.s.v.main(String [] args) throws IOException
{
Printwriter pw=new printwriter("mobile.txt");
BufferedReaderbr=new BufferedReader(new FileReader(" "));

String line=br.readline();
Pattern p=pattern.compile("[7-9][0-9]{9}");
While(line !=null)
{
Matcher m=p.matchers(line);
While(m.find());
{
Pw.println(m.group());
}
Line=br.readline();
}
Pw.flush();
}
}
```
p) W.a.p to extract mail-ids from the given file where mail-ids are mixed with some row data?

->In the above example replace regularExpression with the fallowing mail-id regular Expression.

[a-z A-Z][a-zA-Z 0-9]* (a)[a-z A-Z 0-9]+([.][a-z A-Z]+)+

p)W.a.p to display all text files present in the given directory and

```
import java.io.*;
importjava.util.regex.*;

classFileNameExtraction
{
Public static void main(String[] args) throws IOException
{
Int count=0;
Pattern p=pattern.compile("[a-z A-Z 0-9--]+[.]txt");
File f=new file("D:\\durga_classes");
String[] s=f.list();
For(string s1=s)
{
Matcher m=p.matcher(s1);
If(m.find() &&m.group().equals(s1))
{
Count ++;
S.o.pln(s1);
}
}
s.o.pln(count);
}
}
```

p)w.a.p to delete all .bak files present in D:\\dugra-class

```
import java.io.*;
importjava.util.regx.*;

classFileNamesDeleter
{
int count=0;
Pattern p=pattern.compile("[a-z A-Z 0-9--]+[.]bak");
File f=new file("D:\\durga-classes");
String[] s=f.list();
For(String s1=s)
{
Matcher m=p.matcher(s1);
If(m.find() &&m.group().equals(s1))
{
Count ++;
s.o.pln(s1);
file f1=new file(f1,s1);
f1.delete();
}
}
s.o.pln(count);
}
}
```