Regular Expression:-By using regular expression to represent group of String according to some Patten.

Like mobile number must be start with 9, 8 or 7, email id must end with @XXX.com.

The main purpose of Regular Expression performed validation operation.

Example:-

```
import java.util.regex.*;
class TestRegExp
{
public static void main(String[] args)
{
int count=0;
Pattern p=Pattern.compile("gla");
Matcher m=p.matcher("glaglatgfhhglaglarttrgla");
while(m.find())
{
count++;
System.out.println(m.start()+"------"+m.end()+"------"+m.group());
}
System.out.println("The no of occurences:"+count);
}
}
```

Result:-

```
0------3------gla

3------6------gla

11------14------gla

14------17------gla

21------24------gla

The no of occurences:5
```

To implement Regular Expressions in Java applications we use import "java.util.regex".

java.util.regex package contain two most important class Pattern and Matcher.

Pattern class:-

Pattern class object hold Regular-Expression.

A Pattern object represents "compiled version of Regular Expression". We create a Pattern object by using compile() method of Pattern class.

```
public static Pattern compile(String regularexpression);
```

Example:-

```
Pattern p=Pattern.compile("gla");
```

Matcher class:-

A Matcher object can be used to match character sequences on the given a Regular-Expression.

We create a Matcher object by using matcher() method of Pattern class.

```
public Matcher matcher(String target);
```

Example:-

```
Matcher m=p.matcher("glaglatgfhhglaglarttrgla");
```

To Get some more information about Matched Strings by using following method of Matcher class.

`public boolean find() :-`By using this method we find next match and returns true if it is available otherwise returns false.

`public int start() :-`This method return the start index of the match.

`public int end():-`This method return index of the character just after the end of the matching section.

`public String group():-`This method returns the matched Pattern.

Character Classes:

Character classes are used to specify alphabets and digits in Regular Expressions.

[abc]-----------------Either 'a' or 'b' or 'c'
[^abc] ---------------Except 'a' and 'b' and 'c'
[a-z] -----------------Any lower case alphabet symbol
[A-Z] ----------------Any upper case alphabet symbol
[a-z A-Z] ------------Any alphabet symbol
[0-9] ----------------Any digit from 0 to 9
[a-zA-Z0-9] --------Any alphanumeric character
[^a-zA-Z0-9] ------Any special character

Predefined Character Classes:

\s----space character
\d----Any digit from o to 9 [o-9]
\w----Any word character [a-zA-Z0-9]
. ----Any character including special characters.
\S----any character except space character
\D----any character except digit
\W----any character except word character (special character)

Example:- Write a program in Regular-Expression to check given mobile numbers is valid or not.
Mobile Number must be containing exactly 10 digits and 1st digit should be 7 to 9.

```
import java.util.regex.*;
class MobileValidApp
{
public static void main(String[] args)
{
Pattern p = Pattern.compile("[7-9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9]");
//Pattern p = Pattern.compile("[7-9][0-9]{9}");
Matcher m = p.matcher(args[0]);
if(m.find() && m.group().equals(args[0]) )
{
System.out.println("valid number");
}
else
{
System.out.println("invalid number");
}
}
}
```

Example:- Write a program in Regular-Expression to check given Email-ID is valid or not.

```
import java.util.regex.*;
class CheckEmailID
{
public static void main(String[] args)
{
Pattern p=Pattern.compile("[a-zA-Z][a-zA-Z0-9-.]*@[a-zA-Z0-
9]+([.][a-zA-Z]+)+");

Matcher m=p.matcher(args[0]);
if(m.find()&&m.group().equals(args[0]))
{
System.out.println("valid mail id");
}
else
{
System.out.println("invalid mail id");
}
}
}
```