Regular Expression :
==========

If we want to represent a group of a strings according to particular pattern, then we should go for regular expression.
ex., 1) to represent all mobile numbers.
2) to represent all mail id's

The main important application areas :
1. to develop validation frameworks.
2. to develop pattern matching applications (Ctrl+F in windows, grep in UNIX)
3. to develop translators like compilers, interpreters and assembles etc.
4. to devvelop digital circuits like Moore and Melay machines.
5. to develop communication protocols like TCP/IP, UDP etc.,

Pattern :-

- A pattern object represents a compiled version 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");

Matcher :-

- We can use Matcher object to match the given pattern in the target string.
- We can create Matcher Object by using matcher() of pattern class.
- Syntax :- public Matcher matcher(String target);
- Ex., Matcher m = p.matcher("ababbab");

Methods of Matcher class :
****
1. boolean find() :- It attempts to find next match and returns true if it is available, otherwise returns false.
2. int start() :- returns start index of the match.
3. int end() :- returns end +1 index of the match.
4. String group() :- returns the matched pattern.

Note:- Pattern and Matcher classes present in java.util.reges package and introduced in 1.4 version of java.

Character classes :-
****
1. [abc] - Either a OR b OR c
2. [^abc] - Except a, b and c
3. [a-z] - Any lower case alphabet symbol.
4. [A-Z] - Any upper case alphabet symbol.
5. [a-zA-Z] - Any alphabet symbol.

6. [0-9] - Any digit from 0 to 9
7. [a-zA-Z0-9] - Any alpha numeric symbol
8. [^a-zA-Z0-9] - Except alpha numeric symbol (special characters)

Pre-defined character classes :-
****
1. \\s - Space character
2. \\S - Any character expect space
3. \\d - Any digit from [0-9]
4. \\D - Any character expect digit
5. \\w - Any word character [any alpha numeric character][a-zA-Z0-9]
6. \\W - Except word character (special character)
7. . - Any symbol including special character also.

Quantifiers :-
****

- We can use quantifiers to specify number of occurences to match
1. a - Exactly one a
2. a+ - Atleast one a
3. a* - Any number of a's including zero number also
4. a? - Atmost one a

String Class split() :-
****

- String class also contains split() to Split the given target string according to particular pattern.
Note : In pattern class - If u want to split string from where . character preset for that use "\\." or "[.]"
Pattern class split() can take target string as an argument where as string class split() can take regular expression as an argument.

StringTokenizer :-
****

- It is the specially designmed class for tokenization activity.
- this class present in java.util package.
- Note : the default regular expressiong of stringTokenizer is space .
- Example :

```
StringTokenizer st = new StringTokenizer(11-06-2000", "-");
while(st.hasMoreTokens()) {
     print(st.nextToken());
}
```

Write a regular expression to represent all 10-digit mobile numbers
****
****

Rules :
- Every number should contain exactly 10 digits
- the 1st digit should be 7 or 8 or 9

Answer :

```
[7-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
                            OR
[7-9][0-9]{9}
                            OR
[789][0-9]{9}
```

10 Digit or 11 digit :
==========

```
0?[7-9][0-9]{9}
```

here ? is quantifier : 0 is 1 time or 0 time

10 digit or 11 digit or 12 digit :
==========

```
(0|91)?[7-9][0-9]{9}
```

here ? is quantifier : o or 91 is 1 time or 0 time

Write a regular expression to represent all valid mail id's :
****

```
[a-zA-Z0-9][a-zA-Z0-9_.]*@[a-zA-Z0-9]+([.][a-zA-Z]+)+
```

only gamil id's :
==========

```
[a-zA-Z0-9][a-zA-Z0-9_.]*@gmail[.]com
```

Write a regular expression to represent all valid yava(any languare or own) language identifiers :
****
****

Rules :
- the allowed character are a-z, A-Z, 0-9, #, and $.
- length of identifier should be at least 2.

- 1st character should be lower case alphabetical symbol from a To k.
- 2nd character should be digit divisible by 3 (0, 3, 6, 9)

```
[a-k][0369][a-zA-Z0-9#$]*
```

represent all names starts with a or A
==========

```
[aA][a-zA-Z]*
```

represemt all names ends with l or L
==========

```
[a-zA-Z]*[lL]
```

represent all names starts with a or A and ends with l or L
==========

```
[aA][a-zA-Z]*[lL]
```

1) WAP to check whether the given number is a vlaid mobile number or not
2) WAP to check whether the given mail id is valid or not?
3) WAP to extract all valid mobile numbers present in the given txt file where numbers are mixed with normal text data?
4) WAP to extract all valid mail id's present in the given text file where mail id's are mixed with normal text data
5) WAP to display all .txt file names present in given path