Regular Expression Java (Imp)

A regular Exp. is a sequence of char that create a Search Pattern.

When you search data into a text, you can use this search pattern to describe what you are looking for.

eg

^[a-z0-9_-]{3,15}$ → end of line

start

letter, no, underscore, hyphens.

Java Regex

API that is used to define a pattern for searching/manipulating Strings. It is used to define constraints on String such as password or Email Validation.

Matcher class

boolean matches() → Test whether given regular exp matches or not

boolean find() → used to find next expression that matches the pattern.

boolean find(int start) → Search next exp from the given start no.

String group() → return matched Sequence.

int groupCount → return total no of matched Seq.

Pattern class

Static Pattern compile(String regex) → Compile given regex and return instance of a pattern

Matcher matcher(CharSeq input) → used to create the matcher the given input with pattern.

Ex Pattern pattern = Pattern.compile(".xx.");
Matcher matcher = Pattern.matcher("AxxB.");
[boolean result = matcher.matches()]

Regex classes

[abc] →	a, b, or c
[^abc] →	Any class except a, b or c
[a-z A-Z] →	A through z or a-z (Range)
[a-d[m-p]] →	a through d or m through p (union)
[a-z && [def]] →	d, e or f (intersection)
[a-z && [^bc]] →	A through z except b or c (Substraction)

Ex
Pattern matcher

- Pattern.matches("[xyz]", "wbcd")) → false (Not x or y or z
- Pattern.matches("[xyz]", "x")) → true (among x, y or z
- Pattern.matches("[xyz]", "xxyyyyz") → false
(x and y come more than once)

Regex quantifier

x? →	x occur one or zero time
x+ →	x occur at least one
x* →	x occur zero or more than one
x{n} →	x occur n times only
x{n,} →	x occur n or more time.
x{y,z} →	x occour at least y time But less than z times.

Regular Expression Java ([illegible])

eg Pattern.matches("[ayz]?", "a") → True

("[ayz]?", "aaa") → false (a more then 1)

("[ayz]?", "ayyyyzz") → false

("[ayz]?", "amuta") → false

("[ayz]?", "ay") → false

("[ayz]+", "a") → True

("[ayz]+", "aaa") → True

("[amn]+", "aayyyzz") → True

("[ayz]+", "aammta") → false

("[ayz]*", "ayyyza") → True.

Reg Meta Char

. any char

d → Repⁿ any Digit

s → White space

w → Word Char

b → Word boundry

D → Not a digit

S → Not White Space

W → Not Word Char

B → Not Word Boundry.

eg Pattern.matches("d", "abc") → false

("d", "1") → True

("d", "2345") → F

("d", "3a") → F

("D", "abc") → F (more then 1)

("D", "1") → F

("D", "32abc") → F

("D", "m") → T

("D*", "abc") → T

matcher

Patten

Email Validation

```
Pattern p = Pattern.compile("________");
  ~~boolean result = Macher m = p.compile(~~
  Macher m = p.compile("String to check");
  Boolean result = m.matches();
```

Eg

```
Int count = 0;
Pattern p = Pattern.compile("ab");
Macher m = p.matcher("ababbaba");
while (m.find()) {
    count++;
    Sop(m.start() + ":" + m.end() + ":" + m.group())
}
sop(count);
```

"ab" → Pattern
"ababbaba" → target

String Split   String s = "I love India";
    String [] tokens = s.split("\\s");   11.

StringTokenizer

```
StringTokenizer st = new ST("________");
while (st.hasMoreTokens()) {
}   st.nextToken();
```

Eg StringToken st = new ST("03-06-2010", "-");

Phone No : 10 digit, 1st digit 7,8,9   6,7,8,9

[6-9][0-9]{9} or [789][0-9]{9}   From Drawn

"[6-9]\\d{9}"

Phone start with 0

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

Digit 0/91

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

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

~~Validation~~ Validation.Java

~~[illegible]~~

// only alphabets → No special Char

```
public boolean isValidCandidateName(String name){
    return name.matches("[A-Za-z]+");
}
```

// Id should be of size 5

```
boolean isValidCandidateId(Integer id){
    return id.toString().length()==5;
}
```

// dept must be ECE, CSE, IT, EEE

```
boolean isValidDepartment(String dept){
    return dept.matches("ECE|CSE|IT|EEE");
}
```

// exam Date can't be today or today after

```
return examDate.isBefore(LocalDate.now())
```

// Checking if marks are not zero

```
(CandidateDto.getMarks1()>0 && ———— —— )
```

// checking gender must be M|F

```
return gender.toString().matches("M|F");
```