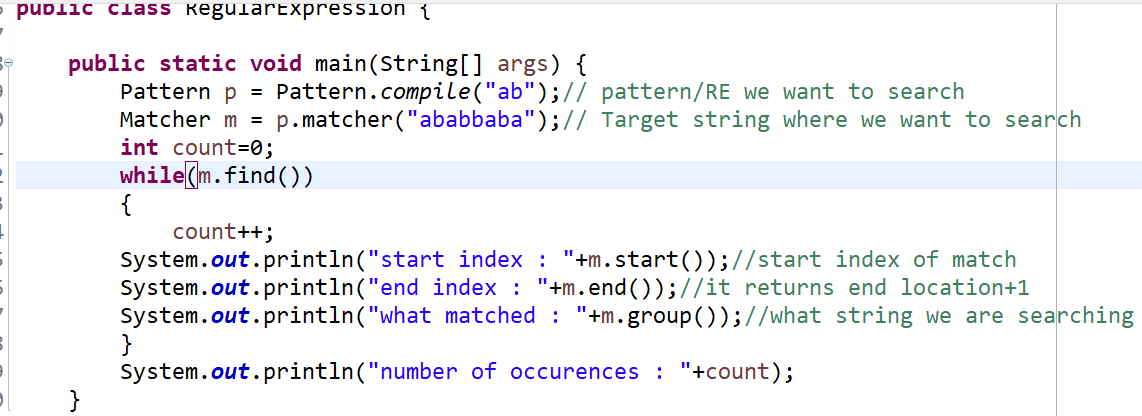
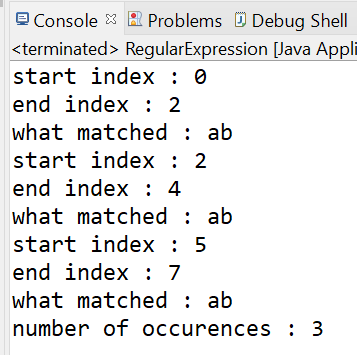
**Regular expression** : It came in 1.4 version of java. It is an expression to represent a group of string objects according to a particular pattern. Ex. Expression to represent all mobile numbers / email-Ids. Where we can use reg ex. : To develop validations, to perform pattern matching (search operation). 1.Create pattern object and then create matcher object.





java.util.regex package

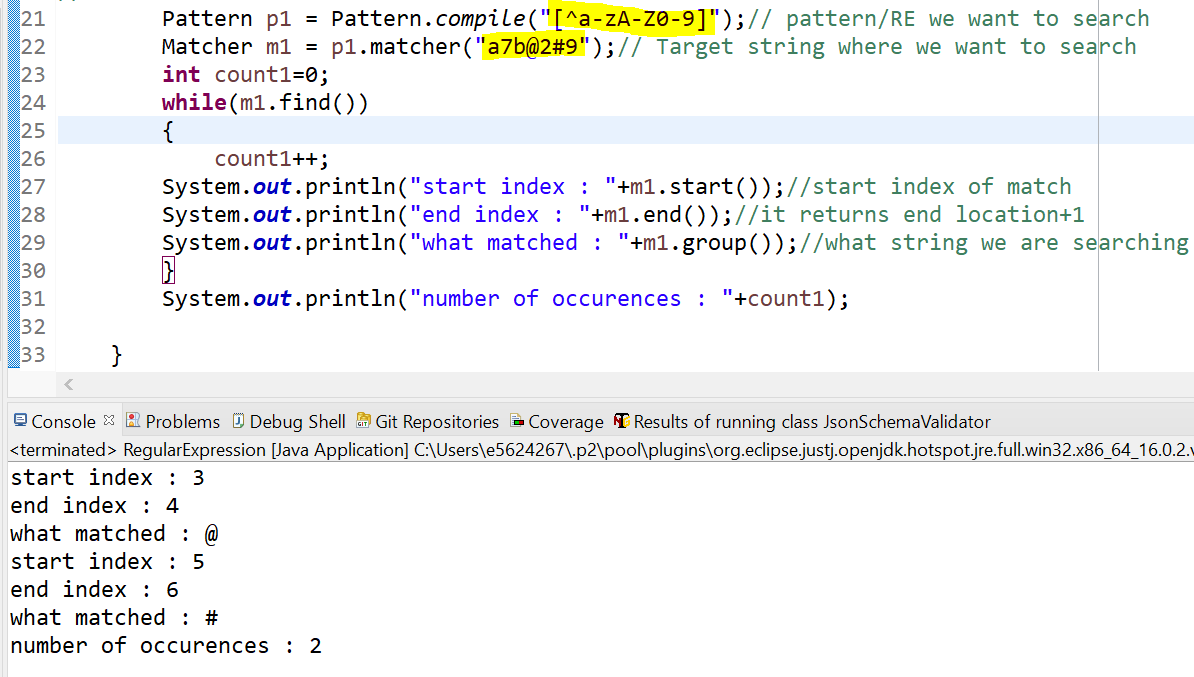
The Matcher and Pattern classes provide the facility of Java regular expression. The java.util.regex package provides following classes and interfaces for regular expressions.

1. MatchResult interface
2. util.regex.Pattern class - Used for defining patterns
3. util.regex.Matcher class - Used for performing match operations on text using patterns
4. PatternSyntaxException class - Used for indicating syntax error in a regular expression pattern

**Methods of Matcher** **class** :

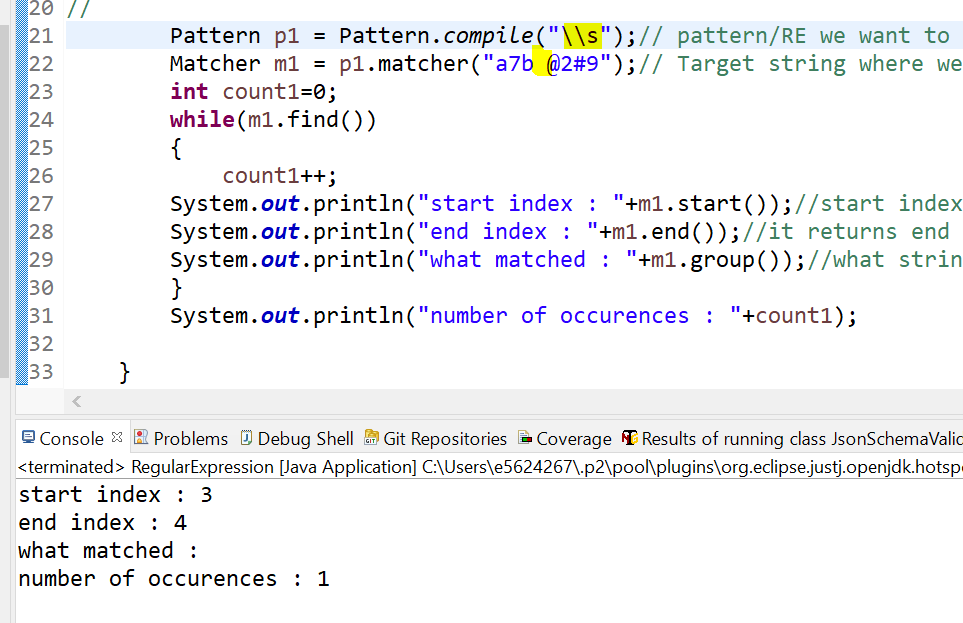
1. boolean find() : finds the expression that matches the pattern.
2. int start() : returns the starting index of the matched subsequence.
3. Int end() : returns the ending index+1 of the matched subsequence.
4. String group() : returns the matched subsequence.

|  |  |  |
| --- | --- | --- |
| 1 | [abc] | a, b, or c (simple class) |
| 2 | [^abc] | Any character **except** a, b, or c (negation) |
| 3 | [a-z] | a through z |
| 4 | [A-Z] | A through Z |
| 5 | [a-zA-Z] | a through z or A through Z, inclusive (range) |
| 6 | [0-9] | Any digit from 0-9 |
| 7 | [a-zA-Z0-9] | Any alphanumeric |
| 8 | [^a-zA-Z0-9] | Except alphanumeric (means all special characters) |
|  |



**Predefined character classes** :

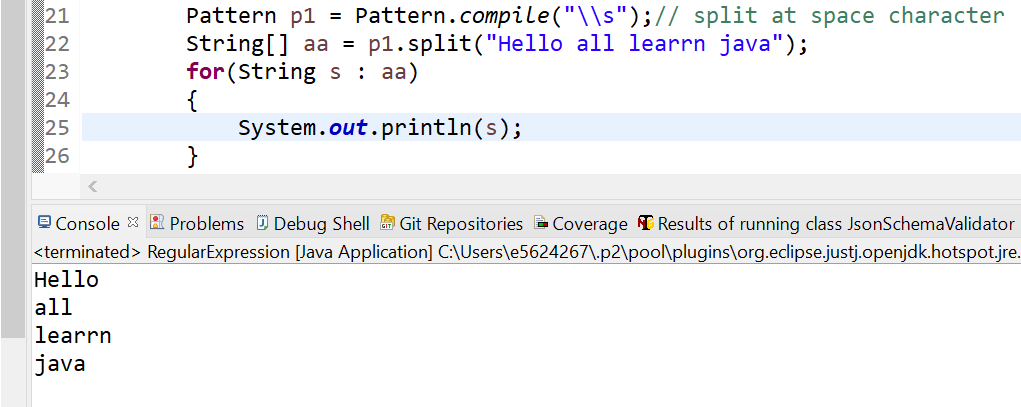
|  |  |
| --- | --- |
| . | Any character |
| \d | Any digits, means any digit from 0-9: [0-9] |
| \D | Any non-digit, any character |
| \s | Any whitespace character |
| \S | Except whitespace, any char. |
| \w | Any word character, means any alphanumeric character [a-zA-Z\_0-9] |
| \W | Any non-word character i.e. special character |
| \b | A word boundary |
| \B | A non word boundary |



**Quantifiers** : We can use to specify number of occurrences to match.

|  |  |
| --- | --- |
| X? | X occurs once or not at all |
| X | X occurs exactly once |
| X+ | X occurs once or more times |
| X\* | X occurs zero or more times |
| X{n} | X occurs n times only |
| X{n,} | X occurs n or more times |
| X{y,z} | X occurs at least y times but less than z times |

**Pattern class split method** : Divide target string according to a particular token / pattern then we should use split method.



**Practice programs** :

1. Validate mobile number : should be 10 digit long, first digit should be 7,8 or 9

Ans : [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}** //means [0-9] 9 times

1. 10 digit OR 11 digit number and 1st digit should be 0 : 0?[7-9][0-9]{9}
2. Represent all mail-Ids : [a-zA-Z0-9][a-zA-Z0-9\_.]\*@[a-zA-Z0-9]+([.][a-zA-Z]+)+

**Split at any character :**

