Regular Expressions

Topics Covered Under This Chapter:

- Introduction
- Pattern & Matcher
- Character Classes
- Quantifiers
- split() Method
- StringTokenizer Class

1.. Introduction

- If we want to represent a group of String According to a particular pattern then we should go for Regular Expression.
- We can write a Regular Expression to write all Valid Mobile Numbers, or to write a Valid Email IDs etc.
- All the Validation Frameworks like Form Validation is internally implemented by using Regular Expression.
- Pattern Matching application like Find Application is internally implemented by using Regular Expression.
- While Designing Network Protocols also internally Regular Expressions are used.

Some Use Cases of Regular Expressions

- Regular Expression for Valid Mobile Number

  Every Mobile Number should be of 10 Digits.
  First Digit should be either 7,8, or 9
  Hence, Regular Expression Will be – [7-9][0-9]{9}

- Regular Expression for Valid Email IDs

  Regular Expression Will be – [a-zA-Z0-9][a-zA-Z0-9_.]*@[a-zA-Z0-9]+([.][a-zA-Z]+)+

2.. Pattern and Matcher

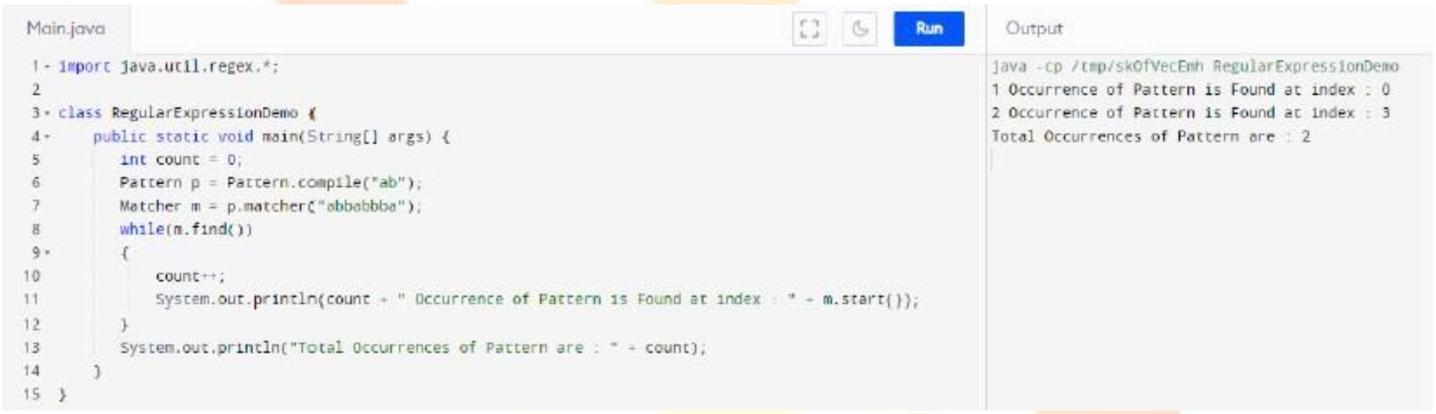
- Pattern and Matcher are Java Classes Present in java.util.regex package.
- Pattern Object is the compiled Version of Regular Expression. i.e., it is Java equivalent Object of Regular Expression.
- The Pattern based on which Our String Should be constructed is defined with the help of Pattern Class Object. In Pattern class we have a static method say compile in which we need to pass the argument string which is nothing but a Pattern String.

  Pattern p = Pattern.compile(String pattern);

- The Matcher Object should be created which will indicates the Target String in which we have to apply the created Pattern. It matches weather given pattern is present in the target string or not. We can created Matcher class Object by using matcher() method of Pattern class in which we have to pass the Target String as an argument.

  Matcher m = p.matcher(String target);

Sample Program Snippet



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

class RegularExpressionDemo {
    public static void main(String[] args) {
        int count = 0;
        Pattern p = Pattern.compile("ab");
        Matcher m = p.matcher("abbabbba");
        while(m.find())
        {
            count++;
            System.out.println(count + " Occurrence of Pattern is Found at index : " + m.start());
        }
        System.out.println("Total Occurrences of Pattern are : " + count);
    }
}
```

```
java -cp /tmp/skOfVecEmh RegularExpressionDemo
1 Occurrence of Pattern is Found at index : 0
2 Occurrence of Pattern is Found at index : 3
Total Occurrences of Pattern are : 2
```

Important Methods which can be applied on Matcher Object

- **boolean find()**
  Returns true if next Match is found in the Target String for the given Pattern else returns false.

- **int start()**
  Returns the start index of the matched string in the target string.

- **int end()**
  Returns the end Index + 1 of the matched string in the target string.

- **String group()**
  Returns the matched String in the Target String for the given Pattern.

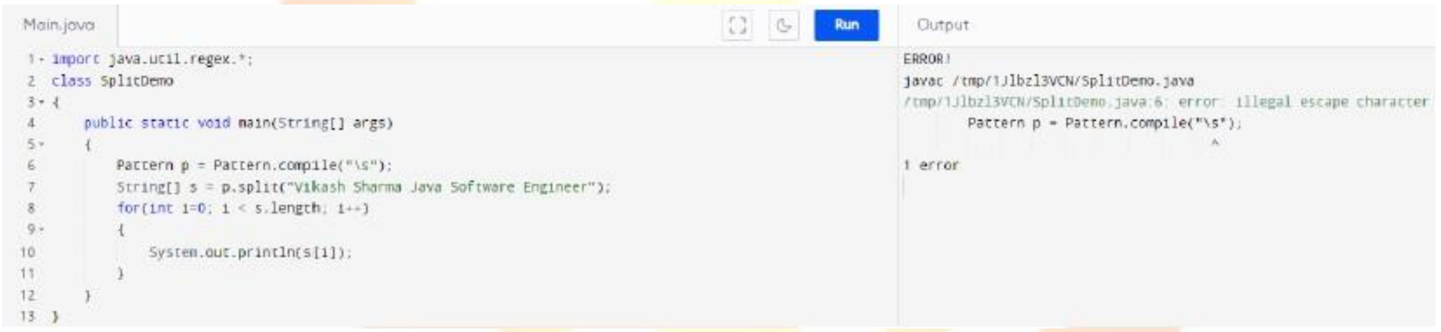
3.. Character Classes

- [abc] : Either ‘a’ or ‘b’ or ‘c’
- [^abc] : Except ‘a’ and ‘b’ and ‘c’
- [a-z] : Any Lower Case Alphabet Symbol from a-z
- [A-Z] : Any Upper Case Alphabet Symbol from A-Z
- [a-zA-Z] : Any Alphabet Symbol
- [0-9] : Any Digit from 0-9
- [A-Za-z0-9] : Any Alphanumeric Symbol
- [^A-Za-z0-9]: Except Alphanumeric Symbols Anything.

3.1 Predefined Character Classes

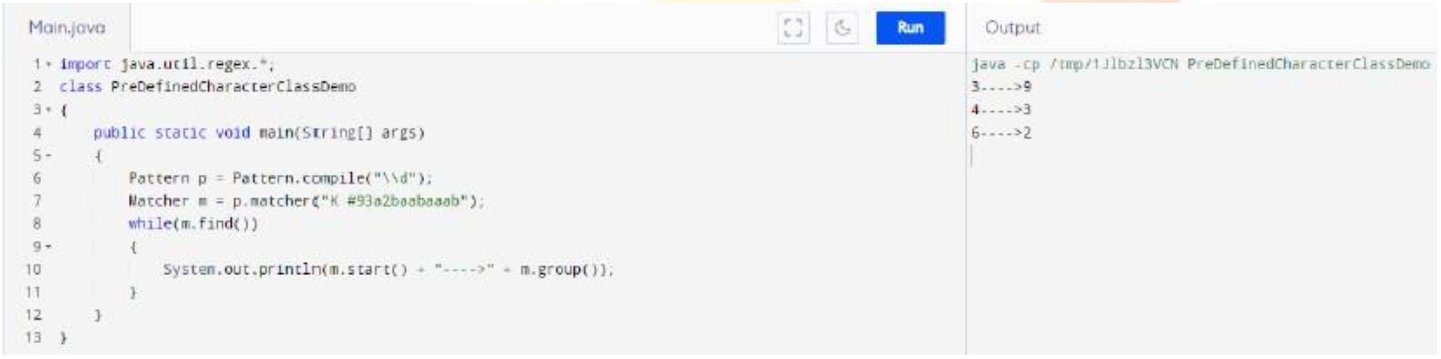
- \s : Space Character
- \S : Anything except Space Character
- \d : Any Digits. Equivalent to [0-9] Character Class.
- \D : Except Digit any Character. Equivalent to [^0-9] Character Class.
- \w : Any Alpha Numeric Symbols. Equivalent to [A-Za-z0-9] Character Class.
- \W : Except Alpha Numeric Symbols. Equivalent to [^A-Za-z0-9] Character Class. Represents Special Characters Class
- . : Any Characters

If we give \s or \S or \d or \D and so on compiler will give error as it will consider it as an Invalid Escape Sequence. So, we have to write the pattern followed by \\.



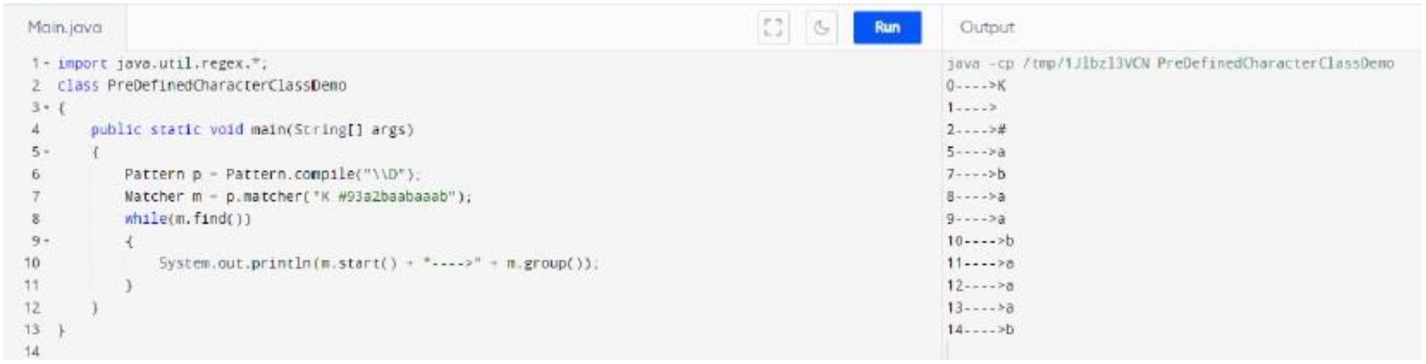
```
import java.util.regex.*;
class SplitDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\s");
        String[] s = p.split("Vikash Sharma Java Software Engineer");
        for(int i=0; i < s.length; i++)
        {
            System.out.println(s[i]);
        }
    }
}
```

```
ERROR!
javac /tmp/1Jlbzl3VCN/SplitDemo.java
/tmp/1Jlbzl3VCN/SplitDemo.java:6: error: illegal escape character
        Pattern p = Pattern.compile("\s");
                                      ^
1 error
```



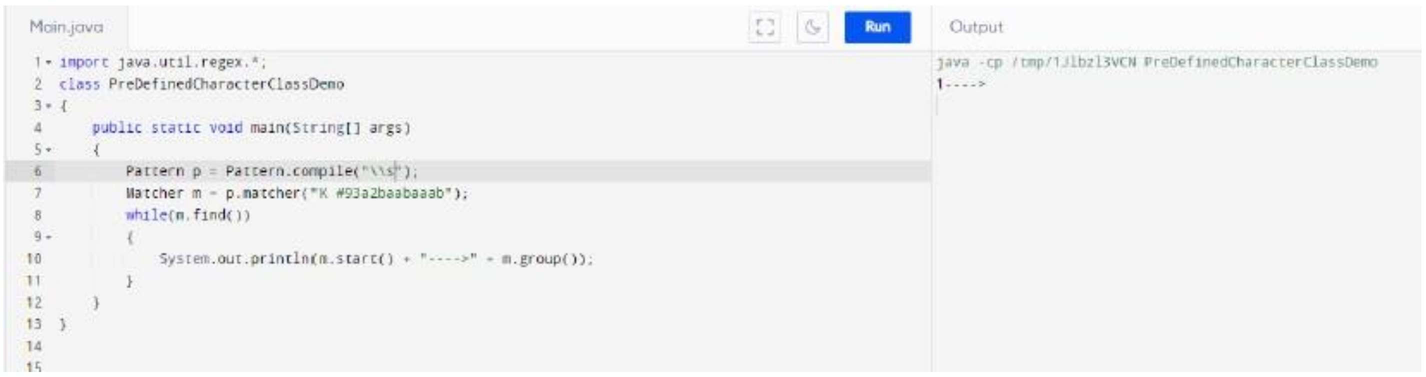
```
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\d");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
3---->9
4---->3
6---->2
```



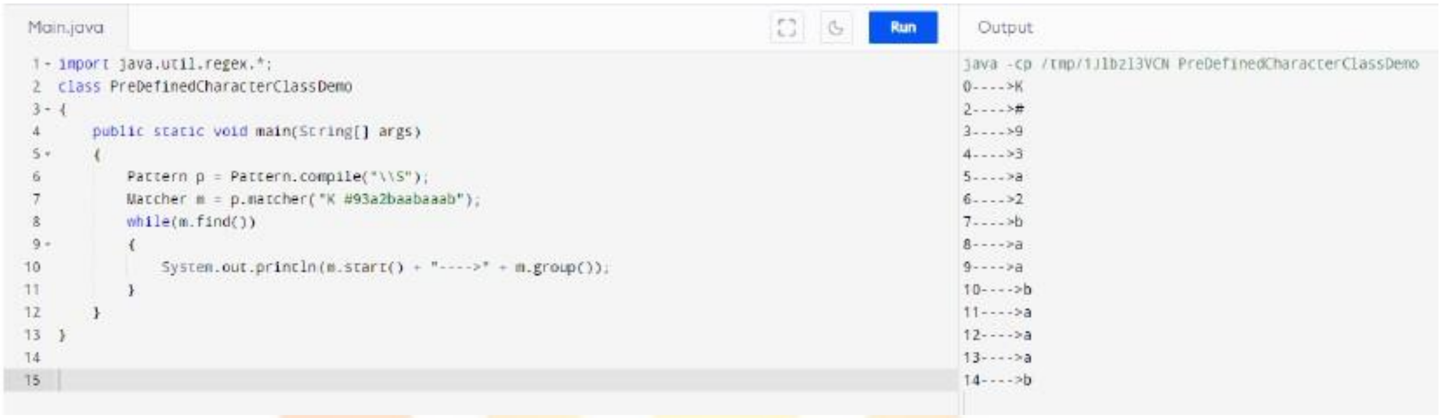
```
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\D");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
0---->K
1---->
2---->#
5---->a
7---->b
8---->a
9---->a
10---->b
11---->a
12---->a
13---->a
14---->b
```



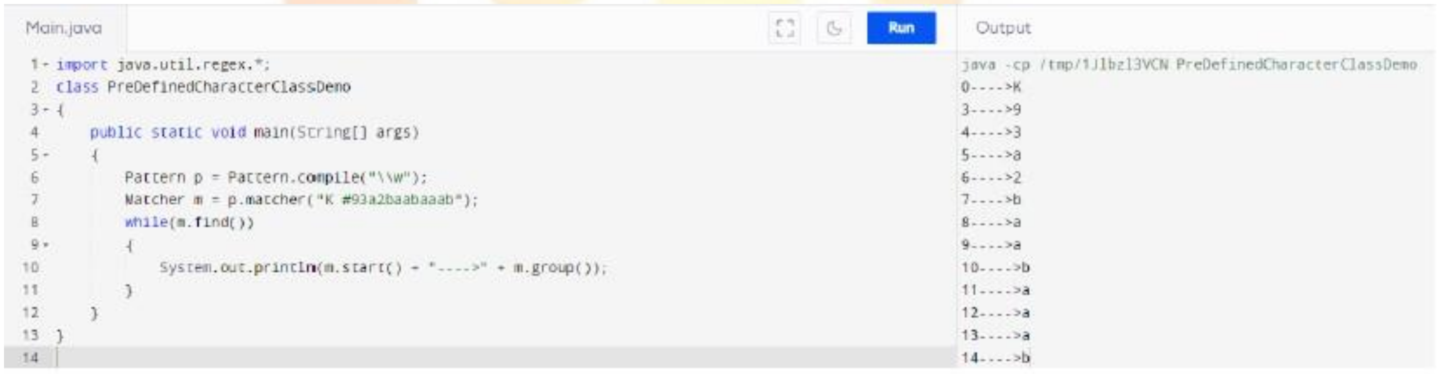
```java
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\s");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
1---->
```



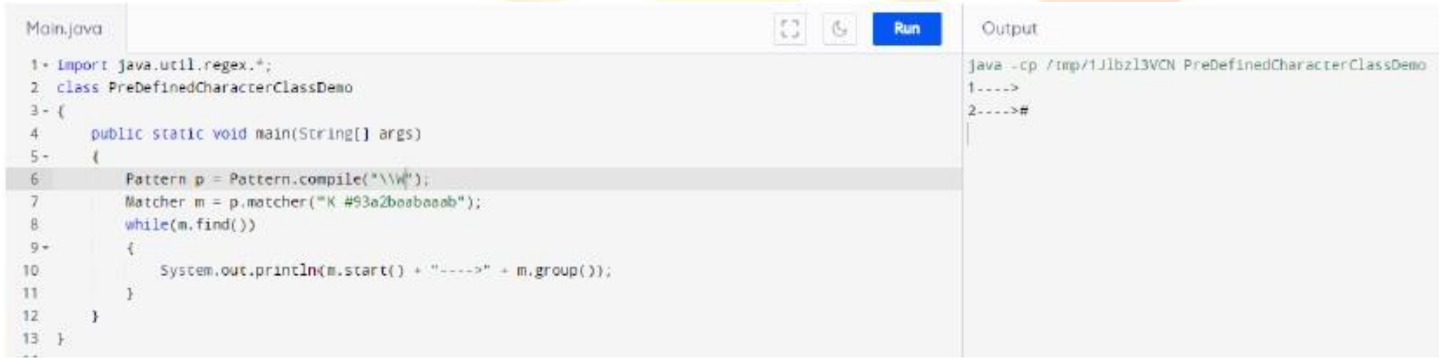
```java
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\S");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
0---->K
2---->#
3---->9
4---->3
5---->a
6---->2
7---->b
8---->a
9---->a
10---->b
11---->a
12---->a
13---->a
14---->b
```



```java
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\w");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
0---->K
3---->9
4---->3
5---->a
6---->2
7---->b
8---->a
9---->a
10---->b
11---->a
12---->a
13---->a
14---->b
```



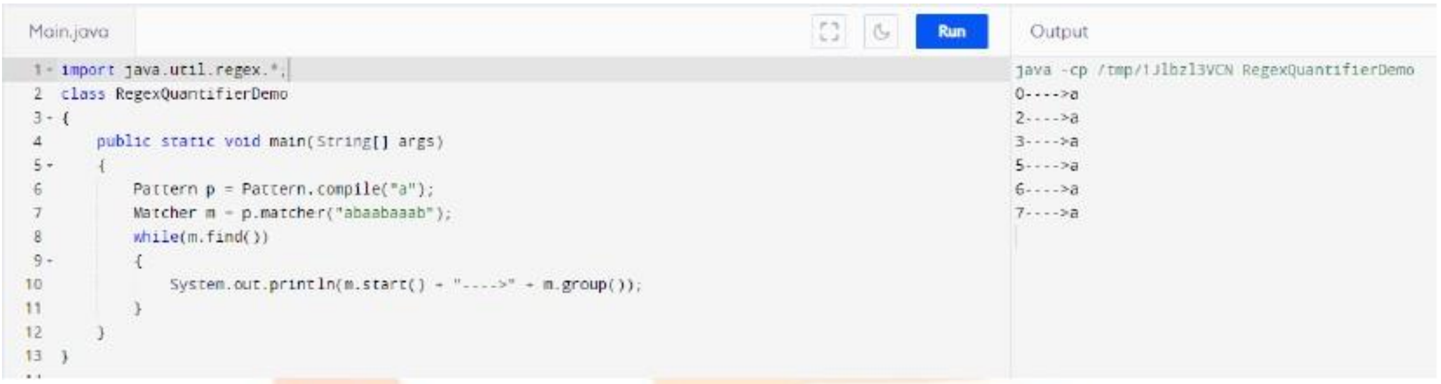
```java
import java.util.regex.*;
class PreDefinedCharacterClassDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\W");
        Matcher m = p.matcher("K #93a2baabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN PreDefinedCharacterClassDemo
1---->
2---->#
```

4.. Quantifiers

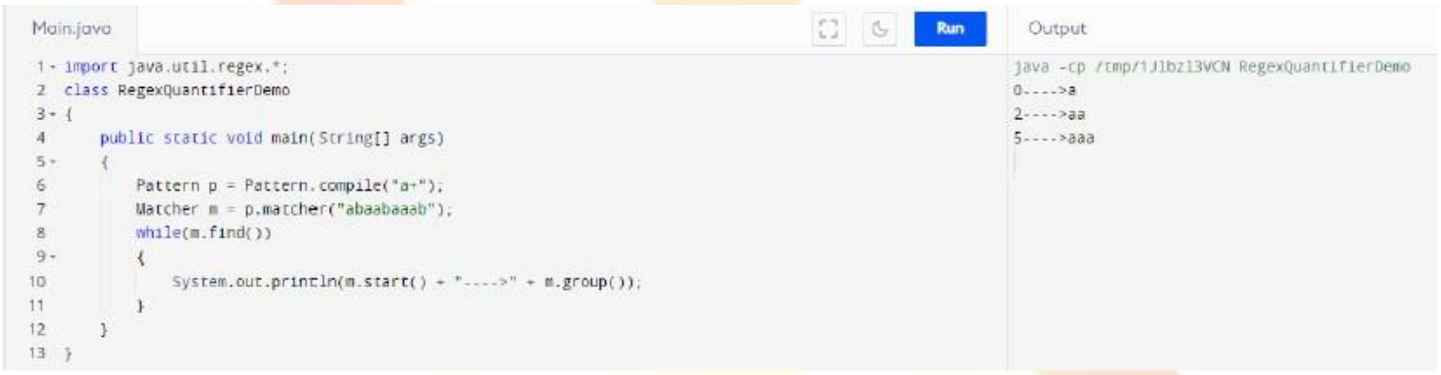
We can use Quantifiers to specify the number of Occurrences to Match.

- a : Exactly One Occurrence of a.
- a+ : At least one Occurrence of a. i.e., One or More Occurrences.
- a* : Any number of Occurrences of a. i.e., 0 or More Occurrences.
- a? : At most one Occurrence of a. i.e., Either 0 or 1 Occurrence.



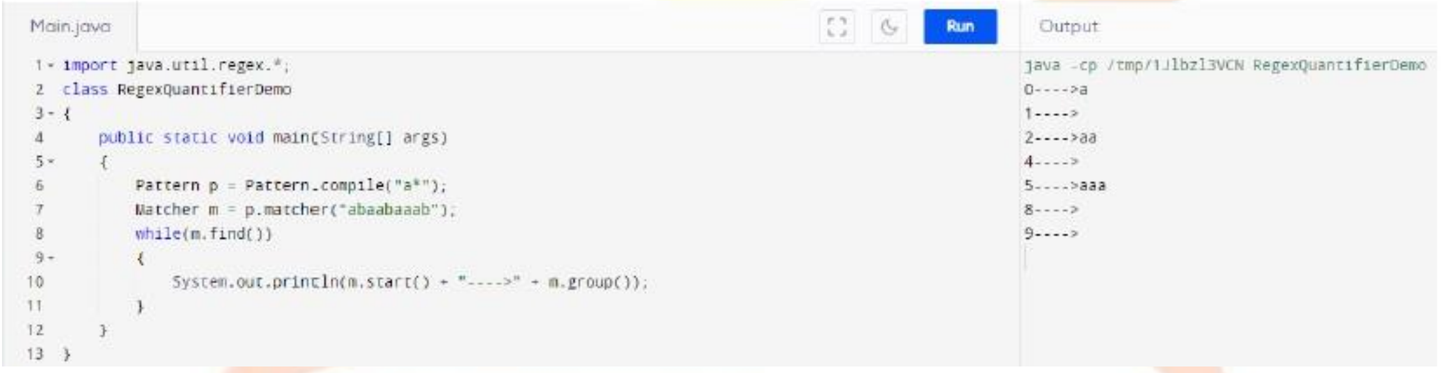
```
import java.util.regex.*;
class RegexQuantifierDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("a");
        Matcher m = p.matcher("abaabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbz13VCN RegexQuantifierDemo
0---->a
2---->a
3---->a
5---->a
6---->a
7---->a
```



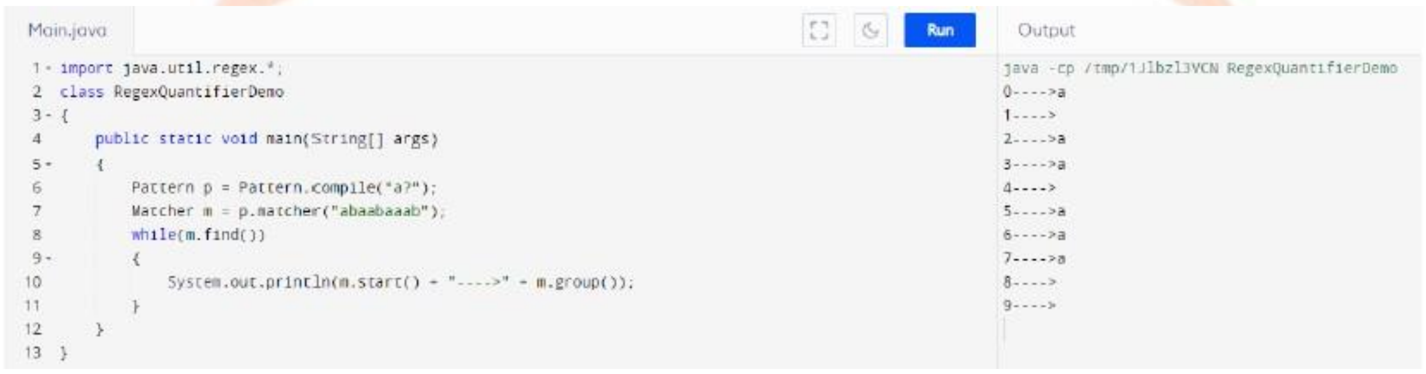
```
import java.util.regex.*;
class RegexQuantifierDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("a+");
        Matcher m = p.matcher("abaabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbz13VCN RegexQuantifierDemo
0---->a
2---->aa
5---->aaa
```



```
import java.util.regex.*;
class RegexQuantifierDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("a*");
        Matcher m = p.matcher("abaabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbz13VCN RegexQuantifierDemo
0---->a
1---->
2---->aa
4---->
5---->aaa
8---->
9---->
```



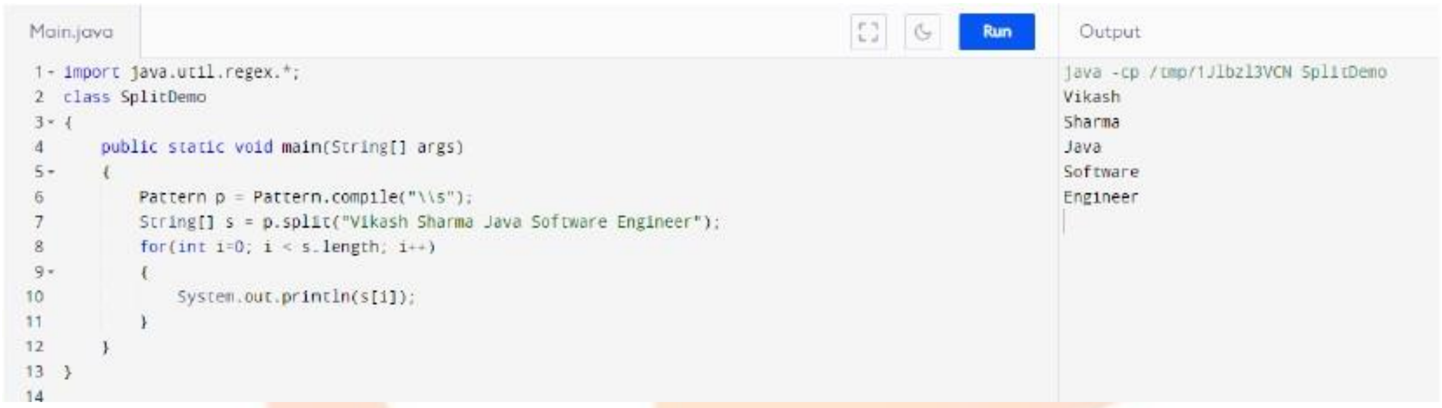
```
import java.util.regex.*;
class RegexQuantifierDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("a?");
        Matcher m = p.matcher("abaabaaab");
        while(m.find())
        {
            System.out.println(m.start() + "---->" + m.group());
        }
    }
}
```

```
java -cp /tmp/1Jlbz13VCN RegexQuantifierDemo
0---->a
1---->
2---->a
3---->a
4---->
5---->a
6---->a
7---->a
8---->
9---->
```

5.. split() method

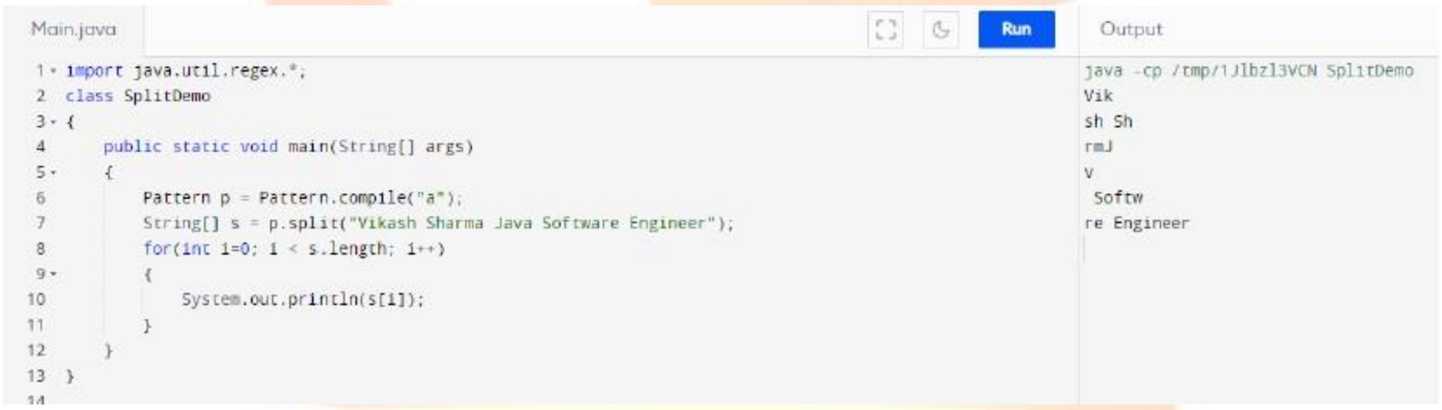
Pattern Class split() method

This method is used to split the given target string as per given pattern p ad returns a String array of Splitted Strings.



```java
import java.util.regex.*;
class SplitDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("\\s");
        String[] s = p.split("Vikash Sharma Java Software Engineer");
        for(int i=0; i < s.length; i++)
        {
            System.out.println(s[i]);
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN SplitDemo
Vikash
Sharma
Java
Software
Engineer
```

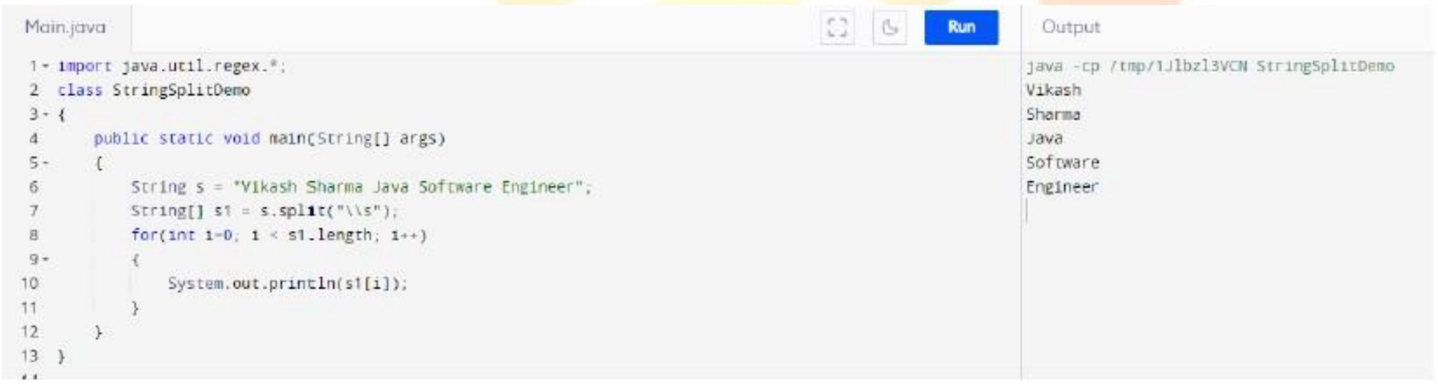


```java
import java.util.regex.*;
class SplitDemo
{
    public static void main(String[] args)
    {
        Pattern p = Pattern.compile("a");
        String[] s = p.split("Vikash Sharma Java Software Engineer");
        for(int i=0; i < s.length; i++)
        {
            System.out.println(s[i]);
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN SplitDemo
Vik
sh Sh
rmJ
v
 Softw
re Engineer
```

String Class split() method

This method is also used to split the String according to some pattern.



```java
import java.util.regex.*;
class StringSplitDemo
{
    public static void main(String[] args)
    {
        String s = "Vikash Sharma Java Software Engineer";
        String[] s1 = s.split("\\s");
        for(int i=0; i < s1.length; i++)
        {
            System.out.println(s1[i]);
        }
    }
}
```

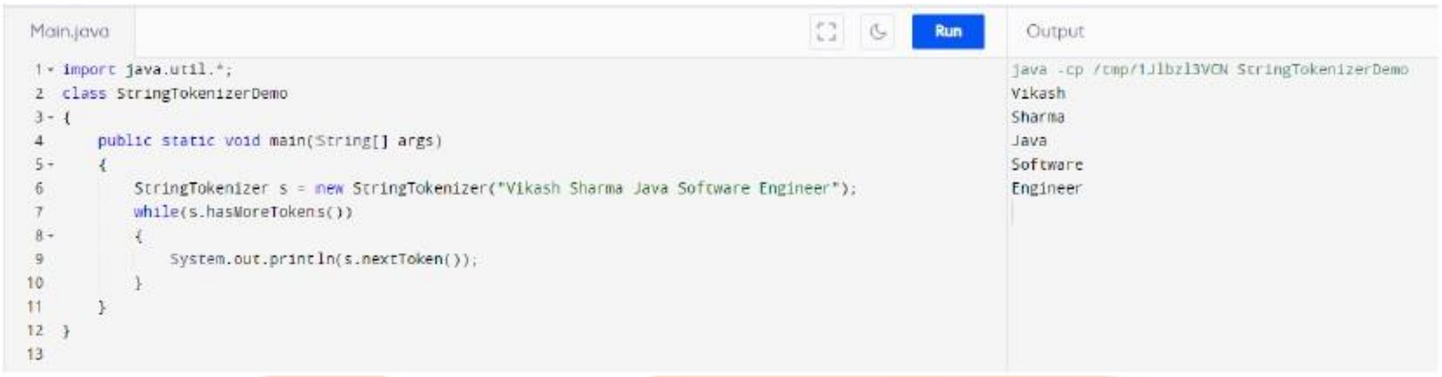
```
java -cp /tmp/1Jlbzl3VCN StringSplitDemo
Vikash
Sharma
Java
Software
Engineer
```

The Only difference between Pattern class split() method and String class split() method is that in Pattern Class Split method we pass the Target String which we wants to split as an argument while in String class split() method we pass pattern string as an argument according to which we wants to split our Target String.

6.. StringTokenizer Class

- StringTokenizer Class is present in java.util Package.
- It is a Specially designed class for Tokenization activity.
- Default regular expression for this class is whitespace.

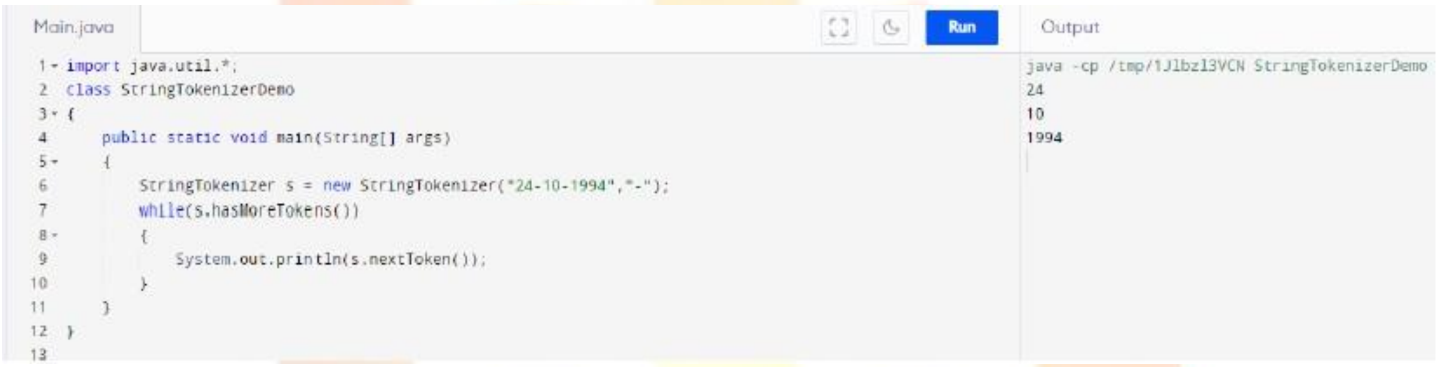
Example



```java
import java.util.*;
class StringTokenizerDemo
{
    public static void main(String[] args)
    {
        StringTokenizer s = new StringTokenizer("Vikash Sharma Java Software Engineer");
        while(s.hasMoreTokens())
        {
            System.out.println(s.nextToken());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN StringTokenizerDemo
Vikash
Sharma
Java
Software
Engineer
```

If we wants to provide some regular expressions explicitly we want to provide Delimiter or Regular Expression in the Constructor of StringTokenizer Class.



```java
import java.util.*;
class StringTokenizerDemo
{
    public static void main(String[] args)
    {
        StringTokenizer s = new StringTokenizer("24-10-1994","-");
        while(s.hasMoreTokens())
        {
            System.out.println(s.nextToken());
        }
    }
}
```

```
java -cp /tmp/1Jlbzl3VCN StringTokenizerDemo
24
10
1994
```