String Manipulations with RegEx

Session Goals

By the end of this session, you should be able to demonstrate how to:

- **use StringBuffer and StringBuilder class**
- **apply Regular Expressions**
- **execute Matcher class and methods**
- **implement Pattern class and methods**

Let Us Try to Find Out

- What is a mutable String?
- Why to use StringBuffer and StringBuilder?
- How does a mutable String work?
- What are Regular Expressions?
- How Regular Expression works?

StringBuffer and StringBuilder

StringBuffer	StringBuilder
Syntax: StringBuilder var = new StringBuilder(str);	Syntax: StringBuilder var = new StringBuilder(str);

StringBuffer and StringBuilder

Method	Description	Example	Output
StringBuilder append(String obj)	Appends the argument to the string builder.	`StringBuilder sb = new StringBuilder("Fruits "); sb.append("are good for health"); System.out.println(sb);`	Fruits are good for health
StringBuilder delete(int start, int end)	Deletes the sequence from start to end in the char sequence.	`StringBuilder str = new StringBuilder("fruits are very good"); str.delete(10, 15); System.out.println("After deletion = " + str);`	After deletion = fruits are good

StringBuffer and StringBuilder

Method	Description	Example	Output
StringBuilder insert(int offset, String obj)	Inserts the second argument into the string builder. The first argument indicates the index before which the data is to be inserted.	*StringBuilder str = new StringBuilder(“fruitsgood”); str.insert(6, “ are “); System.out.print(“After insertion = “);* System.out.println(str.toString());	After insertion = fruits are good
StringBuilder reverse()	The sequence of characters in the string builder is reversed.	*StringBuilder str = new StringBuilder(“fruits”); System.out.println(“reverse = “ + str.reverse());*	reverse = stiurf

String vs StringBuffer vs StringBuilder

	String	StringBuffer	StringBuilder
Storage	String Pool	Heap	Heap
Modifiable	It is not modifiable because it is immutable	It is modifiable because it is mutable	It is modifiable because it is mutable
Thread Safe	It is Thread safe, multiple thread can't access simultaneously	It is Thread safe, multiple thread can't access simultaneously	It is not Thread safe, multiple thread can access simultaneously
Synchronized	Its methods are Synchronized	Its methods are Synchronized	Its methods are not Synchronized
Performance	Peformance is high	Peformance is low	Peformance is high
Usage	If contents are fixed then we can use String	If contents are not fixed and we need thread safety then we can use StringBuffer	If contents are not fixed and we don't need thread safety is then we can use StringBuilder

Java Regex

MatchResult

Interface

Matcher Class

Pattern Class

PatternSyntaxException Class

Matcher Class Methods

Method	Description
boolean matches()	test whether the regular expression matches the pattern.
boolean find()	finds the next expression that matches the pattern.
boolean find(int start)	finds the next expression that matches the pattern from the given start number.
replaceAll()	replaces all matches of the regular expression
int start()	returns the starting index of the matched subsequence.
int end()	returns the ending index of the matched subsequence.
replaceFirst()	Replaces only the first match.

Matcher Methods Demo

find(), start() , end()

```
package com.stackroute.regularexpressions.matchermethods;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherMethodsDemo {
	public static void main(String[] args) {
	String text = "This is the text which is to be searched " +
			"for occurrences of the word 'is'.";
	String patternString = "is";

	 Pattern pattern = Pattern.compile(patternString);
	 Matcher matcher = pattern.matcher(text);

		while (matcher.find()) {
		  System.out.println("found: " + " : "
				+ matcher.start() + " - " + matcher.end());
	 }}}
```

Matcher Methods Demo

replaceAll(), replace

```
package com.stackroute.regularexpressions.matchermethods;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherMethodsDemo {
	public static void main(String[] args) {
	String text = "This is the text which is to be searched " +
			"for occurrences of the word 'is'.";
	String patternString = "is";

	 Pattern pattern = Pattern.compile(patternString);
	 Matcher matcher = pattern.matcher(text);

	 String s = matcher.replaceAll("by");
	 System.out.println(s);

	 String s1 = matcher.replaceFirst("rep");
	 System.out.println(s1);
```

Pattern Class Methods

Method	Description
static Pattern compile(String regex)	compiles the given regex and returns the instance of the Pattern.
Matcher matcher(CharSequence input)	creates a matcher that matches the given input with the pattern.
static boolean matches(String regex, CharSequence input)	It works as the combination of compile and matcher methods. It compiles the regular expression and matches the given input with the pattern.
String[] split(CharSequence input)	splits the given input string around matches of given pattern.
String pattern()	returns the regex pattern.

Pattern Class Methods Demo

```
package com.stackroute.regularexpressions.patternmethods;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PatternMatchesDemo {
    public static void main(String[] args) {

        System.out.println(
                Pattern.matches("tom", "Tom"));

        System.out.println(
                Pattern.matches("[tT]im|[jJ]in", "Tim"));

        String text = "Text to be searched";

        String patternString = ".*be.*";

        Pattern pattern = Pattern.compile(patternString);

        Matcher matcher = pattern.matcher(text);
        boolean matches = matcher.matches();

        System.out.println(matches);}
}
```

Pattern Class Methods Demo Contd.

```
package com.stackroute.regularexpressions.patternmethods;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PatternMatchesDemo {
    public static void main(String[] args) {


            String str = "welcometostackroute";
            String[] arrOfStr = str.split("s");
            for (String a : arrOfStr)
               System.out.println(a);
    }
}
```

PatternSyntaxException Class Methods

Method	Description
public String getDescription()	Retrieves the description of the error.
public int getIndex()	Retrieves the error index.
public String getPattern()	Retrieves the erroneous regular expression pattern.
public String getMessage()	Returns a multi-line string containing the description of the syntax error and its index, the erroneous regular expression pattern, and a visual indication of the error index within the pattern.

Validate Phone Number

```
package com.stackroute.regularexpressions;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class ValidatePhoneNumber {

    public static boolean isPhoneNumberValid(String s) {
        Pattern pattern = Pattern.compile("(0/91)?[7-9][0-9]{9}");
        Matcher matcher = pattern.matcher(s);
        return (matcher.find() && matcher.group().equals(s));
    }

    public static void main(String[] args) {
        String s = "787878787878";
        if (isPhoneNumberValid(s))
            System.out.println("Valid Number");
        else
            System.out.println("Invalid Number");
    }
}
```

Validate Email Address

```
package com.stackroute.regularexpressions;
import java.util.regex.Pattern;
public class ValidatEmailAddress {
    public static boolean isEmailValid(String email) {
        String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\." +
                "[a-zA-Z0-9_+&*-]+)*@" +
                "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
                "A-Z]{2,7}$";
        Pattern pattern = Pattern.compile(emailRegex);
        if (email == null)
            return false;
        return pattern.matcher(email).matches();
    }
    public static void main(String[] args) {
        String email = "demo@gmail.com";
        if (isEmailValid(email))
            System.out.print("Yes");
        else
            System.out.print("No");
    }
}
```

Key TakeAways

At the end of this session, you should be able to demonstrate how to:

- **Use StringBuffer and StringBuilder class**
- **Apply Regular Expressions**
- **Implement Matcher class and methods**
- **Implement Pattern class and methods**

Thank You!