NAME: VISHWAK SENAN

REGULAR EXPRESSION:

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character,or a more complicated pattern.
- Regular expressions can be used to perform all typese of text search and text replace opeartions.
- Java does not have a built-in regular expression class, but we can import the

Java.util.regex package to work with regular expression.

Java.util.regex package:

- **Pattern** class - Defines a pattern(to be used in a search)
- **Matcher** class – Used to search for the pattern.
- **PatternSyntaxException** class – Indicates synatx error in a regular expression pattern.

EXAMPLE:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class Main
{
 public static void main(String[] args)
{
        Pattern pattern = Pattern.compile("w3schools",
Pattern.CASE_INSENSITIVE);
```

```
        Matcher matcher = pattern.matcher("Visit W3Schools!");     boolean
      matchFound = matcher.find();
    if(matchFound)
 {
     System.out.println("Match found");
    }
else {
      System.out.println("Match not found");
    }
  }}
```

```
OUTPUT
Match found
```

Flags

Flags in the compile() method change how the search is performed. Here are a few of them:

- **Pattern.CASE_INSENSITIVE** - The case of letters will be ignored when performing a search.
- **Pattern. LITERAL** - Special characters in the pattern will not have any special meaning and will be treated as ordinary characters when performing a search.
- **Pattern.UNICODE_CASE** - Use it together with the CASE_INSENSITIVE flag to also ignore the case of letters outside of the English alphabet

Regular Expression Patterns

The first parameter of the Pattern.compile() method is the pattern. It describes what is being searched for.

Brackets are used to find a range of characters:

Expression	Description

[abc]	Find one character from the options between the brackets
[^abc]	Find one character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

LocalDateTime class:

- Java LocalDateTime class is an immutable date-time object that represents a date-time, with the default format as yyyy-MM-dd-HH-mm-ss.zzz.
- It inherits object class and implements the ChronoLocalDateTime interface.

Methods of Java LocalDateTime

Method	Description
String format(DateTimeFormatter formatter)	**It is used to format this date-time using the specified formatter.**
int get(TemporalField field)	**It is used to get the value of the specified field from this date-time as an int.**
LocalDateTime minusDays(long days)	**It is used to return a copy of this LocalDateTime with the specified number of days subtracted.**
static LocalDateTime now()	**It is used to obtain the current date-time from the system clock in the default time-zone.**
static LocalDateTime of(LocalDate date, LocalTime time)	**It is used to obtain an instance of LocalDateTime from a date and time.**
LocalDateTime plusDays(long days)	**It is used to return a copy of this LocalDateTime with the specified number of days added.**
boolean equals(Object obj)	**It is used to check if this date-time is equal to another date-time.**

Syntax:

Class declaration

public final class LocalDateTime

extends Object

implements Temporal, TemporalAdjuster, ChronoLocalDateTime<LocalDate>, Serializable

EXample

```
// Java Program to illustrate LocalDateTime Class by
// Formatting LocalDateTime to string

// Importing all classes from java.time package
import java.time.LocalDateTime;
import java.time.format.*;
import java.util.*;

// Main class
class TimeTester {

// Main driver method
public static void main(String[] args)
{

// Creating an object of DateTimeFormatter class
DateTimeFormatter formatter
= DateTimeFormatter.ofPattern(
"yyyy-MM-dd HH:mm:ss a");
```

```
// Creating an object of LocalDateTime class
// and getting local date and time using now()
// method
LocalDateTime now = LocalDateTime.now();

// Formatting LocalDateTime to string
String dateTimeString = now.format(formatter);

// Print and Display
System.out.println(dateTimeString);
}
}
```

Output

```
2021-10-10 4:30:56 am
```