Regular Expressions, Date and Time

Regular Expressions

Regular Expressions or Regex (in short) is an API for defining String patterns that can be used for searching, manipulating, and editing a string in Java. Email validation and passwords are a few areas of strings where Regex is widely used to define the constraints. Regular Expressions are provided under java.util.regex package. This consists of 3 classes and 1 interface. The java.util.regex package primarily consists of the following three classes.

Class	Description
util.regex.Pattern	Used for defining patterns
util.regex.Matcher	Used for performing match operations on text using patterns
PatternSyntaxException	Used for indicating syntax error in a regular expression pattern

Class 1: Pattern Class
This class is a compilation of regular expressions that can be used to define various types of patterns, providing no public constructors. This can be created by invoking the compile() method which accepts a regular expression as the first argument, thus returns a pattern after execution.

Class 2: Matcher class
This object is used to perform match operations for an input string in java, thus interpreting the previously explained patterns. This too defines no public constructors. This can be implemented by invoking a matcher() on any pattern object.

Date and Time

Java provides the Date class available in java.util package, this class encapsulates the current date and time.

S. No.	Constructor & Description
1	**Date( )** This constructor initializes the object with the current date and time.
2	**Date(long millisec)** This constructor accepts an argument that equals the number of milliseconds that have elapsed since midnight, January 1, 1970.

S. No.	Method & Description
1	**boolean after(Date date)** Returns true if the invoking Date object contains a date that is later than the one specified by date, otherwise, it returns false.
2	**boolean before(Date date)** Returns true if the invoking Date object contains a date that is earlier than the one specified by date, otherwise, it returns false.
3	**Object clone( )** Duplicates the invoking Date object.
4	**int compareTo(Date date)** Compares the value of the invoking object with that of date. Returns 0 if the values are equal. Returns a negative value if the invoking object is earlier than date. Returns a positive value if the invoking object is later than date.

5	**int compareTo(Object obj)** Operates identically to compareTo(Date) if obj is of class Date. Otherwise, it throws a ClassCastException.
6	**boolean equals(Object date)** Returns true if the invoking Date object contains the same time and date as the one specified by date, otherwise, it returns false.
7	**long getTime( )** Returns the number of milliseconds that have elapsed since January 1, 1970.
8	**int hashCode( )** Returns a hash code for the invoking object.
9	**void setTime(long time)** Sets the time and date as specified by time, which represents an elapsed time in milliseconds from midnight, January 1, 1970.
10	**String toString( )** Converts the invoking Date object into a string and returns the result.