Contents

[**Java Overview** 2](#_Toc507923340)

[**Programming Basics** 2](#_Toc507923341)

[**Java Introduction** 5](#_Toc507923342)

[**Javac(compiler), JVM, Java(interpreter) and JIT(compiler)** 6](#_Toc507923343)

[**Java Features** 11](#_Toc507923344)

[**A Java Program** 14](#_Toc507923345)

[**Simple Class and Object** 19](#_Toc507923346)

[**Data Types** 21](#_Toc507923347)

[**Literals and Escape Sequence** 23](#_Toc507923348)

[**Variables - (Static,Instance,Local)** 24](#_Toc507923349)

[**Access Modifiers** 28](#_Toc507923350)

[**Non-Access Modifiers - (static, final , abstract….)** 31](#_Toc507923351)

[**Operators** 36](#_Toc507923352)

[**Loops** 38](#_Toc507923353)

[**Conditional Structure** 41](#_Toc507923354)

[**Array** 43](#_Toc507923355)

[**Date/Time – old** 56](#_Toc507923356)

[**Regular Expression** 64](#_Toc507923357)

[**Methods and Constructors** 72](#_Toc507923358)

[**Nested Class** 78](#_Toc507923359)

[**Inheritance - (Single , Multilevel, Hierarchical)** 82](#_Toc507923360)

[**Polymorphism** 86](#_Toc507923361)

[**Abstraction** 88](#_Toc507923362)

[**Encapsulation** 90](#_Toc507923363)

[**Interface** 91](#_Toc507923364)

[**Package** 96](#_Toc507923365)

[**Collections** 98](#_Toc507923366)

|  |
| --- |
| **Java Overview** |
|  |
|  |
|  |
|  |
|  |
| **Date/Time – old** |
|  |
|  |
| What are the classes Date Time API available in Java 7? |
| Java 7 has the following date and time classes and methods.  System.currentTimeMillis()  java.util.Date  java.sql.Date  java.sql.Timestamp  java.util.Calendar  java.util.GregorianCalendar  java.util.TimeZone  java.text.SimpleDateFormat |
| What is the use of System.currentTimeMillis() ? |
| System.currentTimeMillis() is A static method that returns the current date and time as milliseconds since January 1st 1970. |
| What is the use of java.util.Date ? |
| Java's java.util.Date class represents date and time  Today most of the methods in the class are deprecated in favor of the java.util.Calendar class.  You can still use the java.util.Date class to represent a date though.  It has two constructor :  Date() : To initializes the object with the current date and time.  Date(long millisec) : This constructor accepts an argument that equals the number of milliseconds that have elapsed since midnight, January 1, 1970. |
| What are the methods java.util.Date supports? |
| |  |  | | --- | --- | |  |  | | boolean after(Date date) | Returns true if the invoking Date object is later than the one specified by date | | boolean before(Date date) | Returns true if the invoking Date object is earlier than the one specified by date | | Object clone( ) | Duplicates the invoking Date object. | | 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. | | int compareTo(Object obj) | Operates identically to compareTo(Date) if obj is of class Date. Otherwise, it throws a ClassCastException. | | boolean equals(Object date) | Returns true if the invoking Date object contains the same time and date as the one specified by date | | long getTime( ) | Returns the number of milliseconds that have elapsed since January 1, 1970. | | int hashCode( ) | Returns a hash code for the invoking object. | | 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 | | String toString( ) | Converts the invoking Date object into a string and returns the result. | |
| What is the use of java.text.DateFormat class ? |
| There are two classes for formatting date in java:  DateFormat  SimpleDateFormat.  The DateFormat class is an abstract class. Provides various methods to format and parse date and time in java in language independent manner.  java.text.Format is the parent class  java.text.SimpleDateFormat is the subclass |
| List down some of the methods of DateFormat class ? |
| |  |  | | --- | --- | | Public Method | Description | | Date parse(String source)  throws ParseException | converts string into Date object. | | final String format(Date date) | converts given Date object into string. | | static final DateFormat getTimeInstance() | returns time formatter with default formatting style for the default locale. | | static final DateFormat getDateInstance() | returns date formatter with default formatting style for the default locale. | | static final DateFormat getDateTimeInstance() | returns date/time formatter with default formatting style for the default locale. | | static final DateFormat getInstance() | returns date/time formatter with short formatting style for date and time. | | static Locale[] getAvailableLocales() | returns an array of available locales. | |
| What does java.text.SimpleDateFormat class does? |
| The java.text.SimpleDateFormat class provides methods to format and parse date and time in java.  The SimpleDateFormat is a concrete class which inherits java.text.DateFormat class. |
| What is java.util.GregorianCalendar class does ? |
| The java.util.GregorianCalendar class is a concrete subclass of Calendar abstract class and provides the standard calendar system used by most of the world.  It is a hybrid calendar that supports both the Julian and Gregorian calendar systems.  The Julian calendar specifies leap years every four years, whereas the Gregorian calendar omits century years which are not divisible by 400.  Field: Following are the fields for java.util.GregorianCalendar class −  static int AD − This is the value of the ERA field indicating the common era (Anno Domini), also known as CE.  static int BC − This is the value of the ERA field indicating the period before the common era (before Christ), also known as BCE.  List of popular constructors :  GregorianCalendar():  This constructs a default GregorianCalendar using the current time in the default time zone with the default locale.  GregorianCalendar(int year, int month, int dayOfMonth):  This constructs a GregorianCalendar with the given date set in the default time zone with the default locale.  GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second):  This constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale.  GregorianCalendar(TimeZone zone, Locale aLocale):  This constructs a GregorianCalendar based on the current time in the given time zone with the given locale. |
| List out some popular methods of GregorianCalendar class ? |
| |  | | --- | | [**void add(int field, int amount)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_add.htm)  This method adds the specified (signed) amount of time to the given calendar field, based on the calendar's rules. | | [**int getActualMaximum(int field)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_getactualmaximum.htm)  This method returns the maximum value that this calendar field could have, taking into consideration the given time value and the current values of the getFirstDayOfWeek, getMinimalDaysInFirstWeek, getGregorianChange and getTimeZone methods. | | [**int getActualMinimum(int field)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_getactualminimum.htm)  This method returns the minimum value that this calendar field could have, taking into consideration the given time value and the current values of the getFirstDayOfWeek, getMinimalDaysInFirstWeek, getGregorianChange and getTimeZone methods. | | [**boolean isLeapYear(int year)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_isleapyear.htm)  This method determines if the given year is a leap year. | | [**void roll(int field, int amount)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_roll_amount.htm)  This method adds a signed amount to the specified calendar field without changing larger fields. | | [**setTimeZone(TimeZone zone)**](https://www.tutorialspoint.com/java/util/gregoriancalendar_settimezone.htm)  This method sets the time zone with the given time zone value. | |
| What does java.util.TimeZone do ? |
| Java TimeZone class represents a time zone offset, and also figures out daylight savings. It inherits the Object class.     |  |  | | --- | --- | | Method | Description | | static String[] getAvailableIDs() | It is used to get all the available IDs supported. | | static TimeZone getDefault() | It is used to get the default TimeZone for this host. | | String getDisplayName() | It is used to return a name of this time zone suitable for presentation to the user in the default locale. | | String getID() | It is used to get the ID of this time zone | | int getOffset(long date) | It is used to return the offset of this time zone from UTC at the specified date. | | void setID(String ID) | It is used to set the time zone ID | |
| **Regular Expression** |
|  |
| Which pakage contains Regular Expressions related methods? |
| * A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. * Java provides the java.util.regex package for pattern matching with regular expressions. * The java.util.regex package primarily consists of the following three classes −  1. Pattern Class –   Pattern class provides no public constructors.  To create a pattern, you must first invoke one of its public static compile() methods, which will then return a Pattern object.   1. Matcher Class –   Like the Pattern class, Matcher defines no public constructors. You obtain a Matcher object by invoking the matcher() method on a Pattern object.  A Matcher object is the engine that interprets the pattern and performs match operations against an input string.   1. PatternSyntaxException –   A PatternSyntaxException object is an unchecked exception that indicates a syntax error in a regular expression pattern. |
| List some of the Subexpression ? |
| |  |  | | --- | --- | | **Subexpression** | **Matches** | | **^** | **Matches the beginning of the line.** | | **$** | **Matches the end of the line.** | | **.** | **Matches any single character except newline. Using m option allows it to match the newline as well.** | | **[...]** | **Matches any single character in brackets.** | | **[^...]** | **Matches any single character not in brackets.** | | \A | Beginning of the entire string. | | \z | End of the entire string. | | \Z | End of the entire string except allowable final line terminator. | | **re\*** | **Matches 0 or more occurrences of the preceding expression.** | | **re+** | **Matches 1 or more of the previous thing.** | | **re?** | **Matches 0 or 1 occurrence of the preceding expression.** | | **re{ n}** | **Matches exactly n number of occurrences of the preceding expression.** | | **re{ n,}** | **Matches n or more occurrences of the preceding expression.** | | **re{ n, m}** | **Matches at least n and at most m occurrences of the preceding expression.** | | a| b | Matches either a or b. | | (re) | Groups regular expressions and remembers the matched text. | | (?: re) | Groups regular expressions without remembering the matched text. | | (?> re) | Matches the independent pattern without backtracking. | | **\w** | **Matches the word characters.** | | **\W** | **Matches the nonword characters.** | | **\s** | **Matches the whitespace. Equivalent to [\t\n\r\f].** | | **\S** | **Matches the nonwhitespace.** | | **\d** | **Matches the digits. Equivalent to [0-9].** | | **\D** | **Matches the nondigits.** | | \A | Matches the beginning of the string. | | \n | Back-reference to capture group number "n". | | \b | Matches the word boundaries when outside the brackets. Matches the backspace (0x08) when inside the brackets. | | \B | Matches the nonword boundaries. | | \n, \t, etc. | Matches newlines, carriage returns, tabs, etc. | | \Q | Escape (quote) all characters up to \E. | | \E | Ends quoting begun with \Q. | |
| Methods of the Matcher Class |
| 1. Index Methods : Index methods provide useful index values that show precisely where the match was found in the input string        1. Study Methods: Study methods review the input string and return a Boolean indicating whether or not the pattern is found          1. Replacement Methods :Replacement methods are useful methods for replacing text in an input string |
| List PatternSyntaxException Class Methods ? |
| * A PatternSyntaxException is an unchecked exception that indicates a syntax error in a regular expression pattern. * The PatternSyntaxException class provides the following methods to help you determine what went wrong |