**Date and Time API**

**Before java 8**

Before Java 8, developers often faced confusion about which package (java.util or java.sql) to import for handling dates and times, as the behavior of classes like Date and Calendar varied between these packages. This led to issues like mismatched output formats, inconsistent APIs, and conversion headaches. The mutability of these classes also caused bugs, especially in multi-threaded applications, while non-thread-safe utilities like SimpleDateFormat added further complexity. The lack of clear time zone handling and 0-based month indexing in Calendar compounded these issues

The java.util package is a general-purpose package that provides utility classes for collections, date and time handling (legacy), random number generation, and more.

## java.sql Package

The java.sql package is specifically designed for database operations. It provides classes and interfaces for interacting with relational databases using JDBC (Java Database Connectivity).

**package** dateandtimeapi;

**import** java.time.Instant;

**import** java.util.Calendar;

**import** java.util.Date;

**public** **class** BeforeJava8 {

**public** **static** **void** main(String[] args) {

Date currentDate = **new** Date(); // it print time date,time

System.***out***.println("Current Date: " + currentDate);

Calendar calendar = Calendar.*getInstance*();

calendar.add(Calendar.***DAY\_OF\_MONTH***, 5);

System.***out***.println("Date After 5 Days: " + calendar.getTime());

// Legacy Date and Time APIs :

/\*

\* The java.util.Date and java.util.Calendar classes are the old APIs.The new

\* java.time API is recommended, but legacy APIs can still be used if needed.

\* You can convert between the two:

\*/

// Convert java.util.Date to Instant

Date date = **new** Date();

Instant instant = date.toInstant();

System.***out***.println("Instant from Date: " + instant);

// Convert Instant to java.util.Date

Date newDate = Date.*from*(instant);

System.***out***.println("Date from Instant: " + newDate);

Calendar calendar1 = Calendar.*getInstance*();

System.***out***.println("Current Date and Time: " + calendar1.getTime());

}

}

**Output**

Date After 5 Days: Fri Nov 22 01:11:30 IST 2024

Instant from Date: 2024-11-16T19:41:30.367Z

Date from Instant: Sun Nov 17 01:11:30 IST 2024

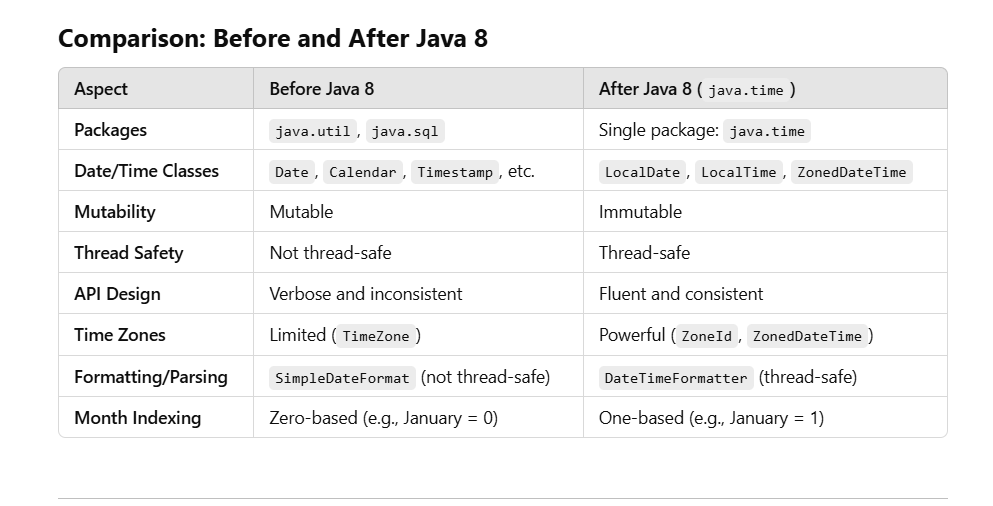
Current Date and Time: Sun Nov 17 01:11:30 IST 2024

**After Java8**

The java.time package introduced in Java 8 as part of the new Date and Time API.

### Key Features of java.time

1. Single Source for Date and Time:
   * All date and time-related functionality is now consolidated under java.time.
   * No more confusion between java.util.Date, java.sql.Date, etc.
2. Clear and Intuitive API:
   * Classes like LocalDate, LocalTime, and ZonedDateTime are easy to use and understand.
3. Immutability:
   * All classes in java.time are immutable, making them thread-safe and reducing bugs.
4. ISO-8601 Standard:
   * Default formats and operations adhere to the widely used ISO-8601 standard.
5. Seamless Interoperability:
   * Conversions between java.time and java.sql are straightforward.
6. Rich Functionality:
   * Built-in support for durations (Duration), periods (Period), time zones (ZoneId), and more.



Ex

1. **LocalDate:**

// 1) LocalDate: Represents a date without a time-zone in the ISO-8601 calendar

// system (e.g.,

// 2024-11-16).

LocalDate currentDate = LocalDate.*now*(); // Current date , format like year-month-date

LocalDate specificDate = LocalDate.*of*(2023, 12, 25); // Specific date , (year, month,date)

LocalDate date = LocalDate.*of*(2024, Month.***DECEMBER***, 20); // we can use (Month.) for adding month

LocalDate futureDate = currentDate.plusDays(5);

System.***out***.println("Current Date: " + currentDate);

System.***out***.println("Specific Date: " + specificDate);

System.***out***.println(" Date: " + date);

System.***out***.println("Date After 5 Days: " + futureDate);

System.***out***.println("");

// Create a LocalDate

LocalDate localDate = LocalDate.*now*();

// Get the month as an enum

Month month = localDate.getMonth();

System.***out***.println("Month (Enum): " + month);

// Get the month as an integer (1-based: January = 1)

**int** monthValue = localDate.getMonthValue();

System.***out***.println("Month (Value): " + monthValue);

System.***out***.println("getYear():" + localDate.getYear());

System.***out***.println("getDayOfMonth():" + localDate.getDayOfMonth());

System.***out***.println("getDayOfYear():" + localDate.getDayOfYear());

System.***out***.println("getChronology():" + localDate.getChronology());

System.***out***.println("getEra():" + localDate.getEra());

System.***out***.println("");

**Output**

Current Date: 2024-11-17

Specific Date: 2023-12-25

Date: 2024-12-20

Date After 5 Days: 2024-11-22

Month (Enum): NOVEMBER

Month (Value): 11

getYear():2024

getDayOfMonth():17

getDayOfYear():322

getChronology():ISO

getEra():CE

1. **LocalTime**

LocalTime: Represents a time without a date or time-zone (e.g., 10:15:30).

package dateandtimeapi;

import java.time.LocalTime;

public class LocalTimeEx {

public static void main(String[] args) {

// 2) LocalTime: Represents a time without a date or time-zone (e.g., 10:15:30).

LocalTime currentTime = LocalTime.*now*(); // Current time

LocalTime specificTime = LocalTime.*of*(14, 30); // Specific time (14:30) hour,minute

LocalTime specificTime1 = LocalTime.*of*(14, 30, 30); // hour,minute,sec

System.*out*.println("Current Time: " + currentTime);

System.*out*.println("Specific Time: " + specificTime);

System.*out*.println("Specific Time using sec: " + specificTime1);

System.*out*.println("getHour():" + currentTime.getHour());

System.*out*.println("getMinute():" + currentTime.getMinute());

System.*out*.println("");

}

}

Output

Current Time: 00:44:53.616755400

Specific Time: 14:30

Specific Time using sec: 14:30:30

getHour():0

getMinute():44

1. **LocalDateTime**

LocalDateTime :Combines date and time without a time-zone

**package** dateandtimeapi;

**import** java.time.LocalDate;

**import** java.time.LocalDateTime;

**import** java.time.LocalTime;

**public** **class** LocalDateTimeEx {

**public** **static** **void** main(String[] args) {

// 3) LocalDateTime :Combines date and time without a time-zone

LocalDateTime currentDateTime = LocalDateTime.*now*(); // Current date and time

LocalDateTime specificDateTime = LocalDateTime.*of*(2023, 12, 25, 14, 30); // Specific date and time

LocalDate currentDate = LocalDate.*now*();

LocalTime currentTime = LocalTime.*now*();

LocalDateTime specificDateTime2 = LocalDateTime.*of*(currentDate, currentTime);

System.***out***.println("Current DateTime: " + currentDateTime);

System.***out***.println("Specific DateTime: " + specificDateTime);

System.***out***.println("Specific DateTime using currentDate,currentTime: " + specificDateTime2);

System.***out***.println("");

}

}

**Output**

Current DateTime: 2024-11-17T00:47:55.086346500

Specific DateTime: 2023-12-25T14:30

Specific DateTime using currentDate,currentTime: 2024-11-17T00:47:55.086346500

1. **ZonedDateTime**

**package** dateandtimeapi;

**import** java.time.ZoneId;

**import** java.time.ZonedDateTime;

**public** **class** ZonedDateTimeEx {

**public** **static** **void** main(String[] args) {

// 4) ZonedDateTime : Represents a date-time with a time-zone (e.g.,

// 2024-11-16T10:15:30+05:30[Asia/Kolkata]).

System.***out***.println(ZoneId.*getAvailableZoneIds*());// it will print all available zone IDs

ZonedDateTime currentDateTim = ZonedDateTime.*now*(); // Current date-time with zone

ZonedDateTime specificZoneTime = ZonedDateTime.*now*(ZoneId.*of*("Asia/Kolkata")); // Time in specific time-zone

System.***out***.println("Current Zoned DateTime: " + currentDateTim);

System.***out***.println("Specific Zoned DateTime: " + specificZoneTime);

System.***out***.println(ZonedDateTime.*now*(ZoneId.*of*("Asia/Calcutta")));

System.***out***.println("");

}

}

**Output**

[Asia/Aden, America/Cuiaba, Etc/GMT+9, Etc/GMT+8, Africa/Nairobi, America/Marigot, Asia/Aqtau, Pacific/Kwajalein, America/El\_Salvador, Asia/Pontianak, Africa/Cairo, Pacific/Pago\_Pago, Africa/Mbabane, Asia/Kuching, Pacific/Honolulu, Pacific/Rarotonga, America/Guatemala, Australia/Hobart, Europe/London, America/Belize, America/Panama, Asia/Chungking, America/Managua, America/Indiana/Petersburg, Asia/Yerevan, Europe/Brussels, GMT, Europe/Warsaw, America/Chicago, Asia/Kashgar, Chile/Continental, Pacific/Yap, CET, Etc/GMT-1, Etc/GMT-0, Europe/Jersey, America/Tegucigalpa, Etc/GMT-5, Europe/Istanbul, America/Eirunepe, Etc/GMT-4, America/Miquelon, Etc/GMT-3, Europe/Luxembourg, Etc/GMT-2, Etc/GMT-9, America/Argentina/Catamarca, Etc/GMT-8, Etc/GMT-7, Etc/GMT-6, Europe/Zaporozhye, Canada/Yukon, Canada/Atlantic, Atlantic/St\_Helena, Australia/Tasmania, Libya, Europe/Guernsey, America/Grand\_Turk, Asia/Samarkand, America/Argentina/Cordoba, Asia/Phnom\_Penh, Africa/Kigali, Asia/Almaty, US/Alaska, Asia/Dubai, Europe/Isle\_of\_Man, America/Araguaina, Cuba, Asia/Novosibirsk, America/Argentina/Salta, Etc/GMT+3, Africa/Tunis, Etc/GMT+2, Etc/GMT+1, Pacific/Fakaofo, Africa/Tripoli, Etc/GMT+0, Israel, Africa/Banjul, Etc/GMT+7, Indian/Comoro, Etc/GMT+6, Etc/GMT+5, Etc/GMT+4, Pacific/Port\_Moresby, US/Arizona, Antarctica/Syowa, Indian/Reunion, Pacific/Palau, Europe/Kaliningrad, America/Montevideo, Africa/Windhoek, Asia/Karachi, Africa/Mogadishu, Australia/Perth, Brazil/East, Etc/GMT, Asia/Chita, Pacific/Easter, Antarctica/Davis, Antarctica/McMurdo, Asia/Macao, America/Manaus, Africa/Freetown, Europe/Bucharest, Asia/Tomsk, America/Argentina/Mendoza, Asia/Macau, Europe/Malta, Mexico/BajaSur, Pacific/Tahiti, Africa/Asmera, Europe/Busingen, America/Argentina/Rio\_Gallegos, Africa/Malabo, Europe/Skopje, America/Catamarca, America/Godthab, Europe/Sarajevo, Australia/ACT, GB-Eire, Africa/Lagos, America/Cordoba, Europe/Rome, Asia/Dacca, Indian/Mauritius, Pacific/Samoa, America/Regina, America/Fort\_Wayne, America/Dawson\_Creek, Africa/Algiers, Europe/Mariehamn, America/St\_Johns, America/St\_Thomas, Europe/Zurich, America/Anguilla, Asia/Dili, America/Denver, Africa/Bamako, Europe/Saratov, GB, Mexico/General, Pacific/Wallis, Europe/Gibraltar, Africa/Conakry, Africa/Lubumbashi, Asia/Istanbul, America/Havana, NZ-CHAT, Asia/Choibalsan, America/Porto\_Acre, Asia/Omsk, Europe/Vaduz, US/Michigan, Asia/Dhaka, America/Barbados, Europe/Tiraspol, Atlantic/Cape\_Verde, Asia/Yekaterinburg, America/Louisville, Pacific/Johnston, Pacific/Chatham, Europe/Ljubljana, America/Sao\_Paulo, Asia/Jayapura, America/Curacao, Asia/Dushanbe, America/Guyana, America/Guayaquil, America/Martinique, Portugal, Europe/Berlin, Europe/Moscow, Europe/Chisinau, America/Puerto\_Rico, America/Rankin\_Inlet, Pacific/Ponape, Europe/Stockholm, Europe/Budapest, America/Argentina/Jujuy, Australia/Eucla, Asia/Shanghai, Universal, Europe/Zagreb, America/Port\_of\_Spain, Europe/Helsinki, Asia/Beirut, Asia/Tel\_Aviv, Pacific/Bougainville, US/Central, Africa/Sao\_Tome, Indian/Chagos, America/Cayenne, Asia/Yakutsk, Pacific/Galapagos, Australia/North, Europe/Paris, Africa/Ndjamena, Pacific/Fiji, America/Rainy\_River, Indian/Maldives, Australia/Yancowinna, SystemV/AST4, Asia/Oral, America/Yellowknife, Pacific/Enderbury, America/Juneau, Australia/Victoria, America/Indiana/Vevay, Asia/Tashkent, Asia/Jakarta, Africa/Ceuta, Asia/Barnaul, America/Recife, America/Buenos\_Aires, America/Noronha, America/Swift\_Current, Australia/Adelaide, America/Metlakatla, Africa/Djibouti, America/Paramaribo, Asia/Qostanay, Europe/Simferopol, Europe/Sofia, Africa/Nouakchott, Europe/Prague, America/Indiana/Vincennes, Antarctica/Mawson, America/Kralendijk, Antarctica/Troll, Europe/Samara, Indian/Christmas, America/Antigua, Pacific/Gambier, America/Indianapolis, America/Inuvik, America/Iqaluit, Pacific/Funafuti, UTC, Antarctica/Macquarie, Canada/Pacific, America/Moncton, Africa/Gaborone, Pacific/Chuuk, Asia/Pyongyang, America/St\_Vincent, Asia/Gaza, Etc/Universal, PST8PDT, Atlantic/Faeroe, Asia/Qyzylorda, Canada/Newfoundland, America/Kentucky/Louisville, America/Yakutat, America/Ciudad\_Juarez, Asia/Ho\_Chi\_Minh, Antarctica/Casey, Europe/Copenhagen, Africa/Asmara, Atlantic/Azores, Europe/Vienna, ROK, Pacific/Pitcairn, America/Mazatlan, Australia/Queensland, Pacific/Nauru, Europe/Tirane, Asia/Kolkata, SystemV/MST7, Australia/Canberra, MET, Australia/Broken\_Hill, Europe/Riga, America/Dominica, Africa/Abidjan, America/Mendoza, America/Santarem, Kwajalein, America/Asuncion, Asia/Ulan\_Bator, NZ, America/Boise, Australia/Currie, EST5EDT, Pacific/Guam, Pacific/Wake, Atlantic/Bermuda, America/Costa\_Rica, America/Dawson, Asia/Chongqing, Eire, Europe/Amsterdam, America/Indiana/Knox, America/North\_Dakota/Beulah, Africa/Accra, Atlantic/Faroe, Mexico/BajaNorte, America/Maceio, Etc/UCT, Pacific/Apia, GMT0, America/Atka, Pacific/Niue, Australia/Lord\_Howe, Europe/Dublin, Pacific/Truk, MST7MDT, America/Monterrey, America/Nassau, America/Jamaica, Asia/Bishkek, America/Atikokan, Atlantic/Stanley, Australia/NSW, US/Hawaii, SystemV/CST6, Indian/Mahe, Asia/Aqtobe, America/Sitka, Asia/Vladivostok, Africa/Libreville, Africa/Maputo, Zulu, America/Kentucky/Monticello, Africa/El\_Aaiun, Africa/Ouagadougou, America/Coral\_Harbour, Pacific/Marquesas, Brazil/West, America/Aruba, America/North\_Dakota/Center, America/Cayman, Asia/Ulaanbaatar, Asia/Baghdad, Europe/San\_Marino, America/Indiana/Tell\_City, America/Tijuana, Pacific/Saipan, SystemV/YST9, Africa/Douala, America/Chihuahua, America/Ojinaga, Asia/Hovd, America/Anchorage, Chile/EasterIsland, America/Halifax, Antarctica/Rothera, America/Indiana/Indianapolis, US/Mountain, Asia/Damascus, America/Argentina/San\_Luis, America/Santiago, Asia/Baku, America/Argentina/Ushuaia, Atlantic/Reykjavik, Africa/Brazzaville, Africa/Porto-Novo, America/La\_Paz, Antarctica/DumontDUrville, Asia/Taipei, Antarctica/South\_Pole, Asia/Manila, Asia/Bangkok, Africa/Dar\_es\_Salaam, Poland, Atlantic/Madeira, Antarctica/Palmer, America/Thunder\_Bay, Africa/Addis\_Ababa, Asia/Yangon, Europe/Uzhgorod, Brazil/DeNoronha, Asia/Ashkhabad, Etc/Zulu, America/Indiana/Marengo, America/Creston, America/Punta\_Arenas, America/Mexico\_City, Antarctica/Vostok, Asia/Jerusalem, Europe/Andorra, US/Samoa, PRC, Asia/Vientiane, Pacific/Kiritimati, America/Matamoros, America/Blanc-Sablon, Asia/Riyadh, Iceland, Pacific/Pohnpei, Asia/Ujung\_Pandang, Atlantic/South\_Georgia, Europe/Lisbon, Asia/Harbin, Europe/Oslo, Asia/Novokuznetsk, CST6CDT, Atlantic/Canary, America/Knox\_IN, Asia/Kuwait, SystemV/HST10, Pacific/Efate, Africa/Lome, America/Bogota, America/Menominee, America/Adak, Pacific/Norfolk, Europe/Kirov, America/Resolute, Pacific/Kanton, Pacific/Tarawa, Africa/Kampala, Asia/Krasnoyarsk, Greenwich, SystemV/EST5, America/Edmonton, Europe/Podgorica, Australia/South, Canada/Central, Africa/Bujumbura, America/Santo\_Domingo, US/Eastern, Europe/Minsk, Pacific/Auckland, Africa/Casablanca, America/Glace\_Bay, Canada/Eastern, Asia/Qatar, Europe/Kiev, Singapore, Asia/Magadan, SystemV/PST8, America/Port-au-Prince, Europe/Belfast, America/St\_Barthelemy, Asia/Ashgabat, Africa/Luanda, America/Nipigon, Atlantic/Jan\_Mayen, Brazil/Acre, Asia/Muscat, Asia/Bahrain, Europe/Vilnius, America/Fortaleza, Etc/GMT0, US/East-Indiana, America/Hermosillo, America/Cancun, Africa/Maseru, Pacific/Kosrae, Africa/Kinshasa, Asia/Kathmandu, Asia/Seoul, Australia/Sydney, America/Lima, Australia/LHI, America/St\_Lucia, Europe/Madrid, America/Bahia\_Banderas, America/Montserrat, Asia/Brunei, America/Santa\_Isabel, Canada/Mountain, America/Cambridge\_Bay, Asia/Colombo, Australia/West, Indian/Antananarivo, Australia/Brisbane, Indian/Mayotte, US/Indiana-Starke, Asia/Urumqi, US/Aleutian, Europe/Volgograd, America/Lower\_Princes, America/Vancouver, Africa/Blantyre, America/Rio\_Branco, America/Danmarkshavn, America/Detroit, America/Thule, Africa/Lusaka, Asia/Hong\_Kong, Iran, America/Argentina/La\_Rioja, Africa/Dakar, SystemV/CST6CDT, America/Tortola, America/Porto\_Velho, Asia/Sakhalin, Etc/GMT+10, America/Scoresbysund, Asia/Kamchatka, Asia/Thimbu, Africa/Harare, Etc/GMT+12, Etc/GMT+11, Navajo, America/Nome, Europe/Tallinn, Turkey, Africa/Khartoum, Africa/Johannesburg, Africa/Bangui, Europe/Belgrade, Jamaica, Africa/Bissau, Asia/Tehran, WET, Europe/Astrakhan, Africa/Juba, America/Campo\_Grande, America/Belem, Etc/Greenwich, Asia/Saigon, America/Ensenada, Pacific/Midway, America/Jujuy, Africa/Timbuktu, America/Bahia, America/Goose\_Bay, America/Virgin, America/Pangnirtung, Asia/Katmandu, America/Phoenix, Africa/Niamey, America/Whitehorse, Pacific/Noumea, Asia/Tbilisi, Europe/Kyiv, America/Montreal, Asia/Makassar, America/Argentina/San\_Juan, Hongkong, UCT, Asia/Nicosia, America/Indiana/Winamac, SystemV/MST7MDT, America/Argentina/ComodRivadavia, America/Boa\_Vista, America/Grenada, Asia/Atyrau, Australia/Darwin, Asia/Khandyga, Asia/Kuala\_Lumpur, Asia/Famagusta, Asia/Thimphu, Asia/Rangoon, Europe/Bratislava, Asia/Calcutta, America/Argentina/Tucuman, Asia/Kabul, Indian/Cocos, Japan, Pacific/Tongatapu, America/New\_York, Etc/GMT-12, Etc/GMT-11, America/Nuuk, Etc/GMT-10, SystemV/YST9YDT, Europe/Ulyanovsk, Etc/GMT-14, Etc/GMT-13, W-SU, America/Merida, EET, America/Rosario, Canada/Saskatchewan, America/St\_Kitts, Arctic/Longyearbyen, America/Fort\_Nelson, America/Caracas, America/Guadeloupe, Asia/Hebron, Indian/Kerguelen, SystemV/PST8PDT, Africa/Monrovia, Asia/Ust-Nera, Egypt, Asia/Srednekolymsk, America/North\_Dakota/New\_Salem, Asia/Anadyr, Australia/Melbourne, Asia/Irkutsk, America/Shiprock, America/Winnipeg, Europe/Vatican, Asia/Amman, Etc/UTC, SystemV/AST4ADT, Asia/Tokyo, America/Toronto, Asia/Singapore, Australia/Lindeman, America/Los\_Angeles, SystemV/EST5EDT, Pacific/Majuro, America/Argentina/Buenos\_Aires, Europe/Nicosia, Pacific/Guadalcanal, Europe/Athens, US/Pacific, Europe/Monaco]

Current Zoned DateTime: 2024-11-17T00:51:27.956193800+05:30[Asia/Calcutta]

Specific Zoned DateTime: 2024-11-17T00:51:27.957191900+05:30[Asia/Kolkata]

2024-11-17T00:51:27.958593200+05:30[Asia/Calcutta]

**5) Instant**

// 5) Instant: Represents a specific point in time (e.g.,

// 2024-11-16T05:45:30.123Z).

Instant now = Instant.*now*(); // Current timestamp(machine understandable)

System.*out*.println("Instant Now: " + now);

System.*out*.println("");

**Output**

Instant Now: 2024-11-16T19:26:21.354455600Z

**6) Formatting and Parsing**

// 6) Formatting and Parsing

// The DateTimeFormatter class is used to format and parse date and time

// objects.

LocalDateTime dateTime = LocalDateTime.*now*();

DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd-MM-yyyy HH:mm:ss");

String formattedDateTime = dateTime.format(formatter);

System.*out*.println("Formatted DateTime: " + formattedDateTime);

System.*out*.println("");

// parsing

String dateTimeString = "16-11-2024 10:15:30";

DateTimeFormatter formatter1 = DateTimeFormatter.*ofPattern*("dd-MM-yyyy HH:mm:ss");

LocalDateTime dateTime1 = LocalDateTime.*parse*(dateTimeString, formatter1);

System.*out*.println("Parsed DateTime: " + dateTime1);

System.*out*.println("");

**Output**

Formatted DateTime: 17-59-2024 12:59:26

Parsed DateTime: 2024-11-16T10:15:30

**7) Working with Period and Duration**

// 7) Working with Period and Duration

// Period: Represents the amount of time in terms of years, months, and days.

LocalDate startDate = LocalDate.*of*(2000, 9, 30);

LocalDate endDate = LocalDate.*now*();

Period period = Period.*between*(startDate, endDate);

System.*out*.println("Years: " + period.getYears());

System.*out*.println("Months: " + period.getMonths());

System.*out*.println("Days: " + period.getDays());

System.*out*.println("");

// Duration: Represents the amount of time in seconds or nanoseconds.

LocalTime startTime = LocalTime.*of*(9, 30);

LocalTime endTime = LocalTime.*of*(17, 45);

Duration duration = Duration.*between*(startTime, endTime);

System.*out*.println("Duration in hours: " + duration.toHours());

System.*out*.println("Duration in minutes: " + duration.toMinutes());

System.*out*.println("");

**Output**

Years: 24

Months: 1

Days: 18

Duration in hours: 8

Duration in minutes: 495

1. **How do you format a LocalDateTime object to a specific date-time string pattern?**

LocalDateTime dateTime = LocalDateTime.*now*();

DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss");

String formattedDate = dateTime.format(formatter);

System.***out***.println("Formatted DateTime: " + formattedDate);

**Output**

Formatted DateTime: 2024-11-17 01:20:57

1. **How do you parse a date or time string using DateTimeFormatter?**

String dateStr = "2024-11-17";

DateTimeFormatter formatter1 = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");

LocalDate date = LocalDate.*parse*(dateStr, formatter1);

System.*out*.println("Parsed Date: " + date);

**Output**

Parsed Date: 2024-11-17

Ex2:

// Input date string in "yyyy-MM-dd" format

String str = "2000-09-12";

// Define the input format (yyyy-MM-dd)

DateTimeFormatter inputFormatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");

// Parse the string into LocalDate

LocalDate date1 = LocalDate.*parse*(str, inputFormatter);

// Define the output format (dd-MM-yyyy)

DateTimeFormatter outputFormatter = DateTimeFormatter.*ofPattern*("dd-MM-yyyy");

// Format the LocalDate into the new format

String formattedDate1 = date1.format(outputFormatter);

// Print the formatted date

System.***out***.println(formattedDate1); // Outputs: 12-09-2000

**Output**

12-09-2000

1. **How do you add or subtract time using the java.time API?**

LocalDate today = LocalDate.*now*();

LocalDateTime now = LocalDateTime.*now*();

// Add days

LocalDate nextWeek = today.plusDays(7);

LocalDateTime nextHour = now.plusHours(1);

// Subtract days

LocalDate lastWeek = today.minusDays(7);

LocalDateTime lastHour = now.minusHours(1);

System.*out*.println("Next Week: " + nextWeek);

System.*out*.println("Next Hour: " + nextHour);

System.*out*.println("Last Week: " + lastWeek);

System.*out*.println("Last Hour: " + lastHour);

**Output**

Next Week: 2024-11-24

Next Hour: 2024-11-17T02:24:56.238206100

Last Week: 2024-11-10

Last Hour: 2024-11-17T00:24:56.238206100