**MyDateTime Class**

**Package name**: com.mydatetime

This is a class which are used to do date and time operations to convert, compare and calculate the desired time, period and date by adding and subtracting time with date.

All the methods in this class are well handled in terms of Exceptions.

This class is totally dependent and uses the following packages for smooth execution;

**java.util;**

**java.text.ParseException;**

**java.text.SimpleDateFormat;**

**java.time;**

**java.time.format.DateTimeFormatter;**

**java.time.temporal.ChronoUnit;**

All the methods inside this class are **public** and **static** and do **casting internally** between Date, LocalDate, String and Calendar objects; also use predefined formatters.

**Predefined Formatters**

|  |  |  |
| --- | --- | --- |
| **symbol** | **Description** | **Example** |
| h | 12 hour in am-pm format | 12 |
| m | Minute in hour | 30 |
| s | Second in minute | 55 |
| d | Day of month | 10 |
| M/L | Month of year | 7; 07; Jul; July; J |
| y | Year of era | 2004; 04 |
| E | Day of week | Tue; Tuesday; |
| z | Time zone name | Pacific Standard time; PST |
| H | Hour of day 0-23 | 19 |

It consists of following methods:

1. **LocalDateToDate():** This method is used to convert the **LocalDate** object to **Date** with the instance of system's default time zone. It returns an object of type **Date**.

It uses **SimpleDateFormat** to format and return the date and **DateTimeFormatter**to validate the input pattern.

Input format: LocalDate object

signature:

public static Date LocalDateToDate(LocalDate ld)

method:

// method to convert LocalDate to Date.

publicstaticDateLocalDateToDate(LocalDatelDate) {

returnDate.from(lDate.atStartOfDay(zoneId).toInstant());

}

**2. dateToString():** This method takes 3 parameters and is used to convert the passed date to String type and returns a string in (dd/mm/yyyy) format.

It uses **SimpleDateFormat** to format and return the date and **DateTimeFormatter**to validate the input pattern.

input format: dd/MM/yyyy

signature:

public static String dateToString(int year, int month, int day)

Method:

// method to convert passed date to string.

public static StringdateToString() {

System.out.print("Enter Date in (dd/MM/yyyy):");

String date = sc.next();

try {

// casting string to localDate

LocalDate myDate = LocalDate.parse(date, Dformatter);

// formatting casted LocalDate to return a string

String dateStr = SDFormatter.format(LocalDateToDate(myDate));

System.out.println("Entered date is: " + dateStr);

Return dateStr;

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}

**3. timeToString():** This method takes 3 parameters and is used to convert the passed time to String type and returns a string in (hh:mm:ss) format.

It uses **SimpleDateFormat** to format the time.

input format: hh:mm:ss

signature:

public static String timeToString(int hour, int minute, int second)

Method:

// method to convert passed time to string.

public static String timeToString() {

System.out.print("Enter Time in (hh:mm:ss): ");

String time = sc.next();

try {

// casting string to localTime

LocalTime myTime = LocalTime.parse(time, Tformatter);

// formatting casted LocalTime to return a string

String timeStr = myTime.format(Tformatter);

System.out.println("Entered time is: " + timeStr);

return timeStr;

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}

**4. dateAndTimeToString():** This method takes 6 parameters and is used to convert the passeddate&time to String type and returns a string in (dd/MM/yyyy; hh:mm:ss) format.

It uses **SimpleDateFormat** to format the time.

date input format: dd/MM/yyyy

time input format: hh:mm:ss

signature:

public static String dateAndTimeToString(int year, int month, int day, int hour, int minute, int second)

Method:

// method to convert passed date-time to string

public static String dateAndTimeToString() {

System.out.print("Enter Date in (dd/MM/yyyy): ");

String date = sc.next();

System.out.print("Enter Time in (hh:mm:ss): ");

String time = sc.next();

String dateTime = date + "; " + time;

try {

// casting string to localDateTime

LocalDateTime myDateTime = LocalDateTime.parse(dateTime, DTformatter);

// formatting casted Localdate-time to return a string

String dateTimeStr = myDateTime.format(DTformatter);

System.out.println("Entered date-time is: "+dateTimeStr);

return dateTimeStr;

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}

**5. daysGap():** This method takes 2 parameters and is used to find the no of days between the period of 2 dates passed. It converts the passed date to **LocalDate** and returns the period in **String** format.

It uses **DateTimeFormatter**to validate the input date passed.

date input format: dd/MM/yyyy

signature:

public static String daysGap(String date1, String date2)

Method:

// method to calculate days between 2 dates

public static StringdaysGap() {

System.out.print("Enter Date1 in (dd/MM/yyyy): ");

String date1 = sc.next();

System.out.print("Enter Date2 in (dd/MM/yyyy): ");

String date2 = sc.next();

try {

// casting string to localDate

LocalDate d1 = LocalDate.parse(date1, Dformatter);

LocalDate d2 = LocalDate.parse(date2, Dformatter);

// calculating days difference

String daysGap = String.valueOf((int)ChronoUnit.DAYS.between(d1, d2));

System.out.println("Days gap is: " + daysGap);

Return daysGap + " days";

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}

**6. daysAndTimeGap():** This method takes 2 parameters and is used to find the time period between the dates passed. It converts the passed dates to **LocalDate** and returns the time period between them in **String** format.

It uses **DateTimeFormatter**to validate the input date passed.

Input format: dd/MM/yyyy; HH:mm:ss

signature:

public static String daysAndTimeGap(String date1, String date2)

Method:

// method to calculate days and time between 2 dates

public static StringdaysAndTimeGap() {

System.out.print("Enter Date-time1 in (dd/MM/yyyy; HH:mm:ss): ");

String dt1 = sc.nextLine();

System.out.print("Enter Date-time2 in (dd/MM/yyyy; HH:mm:ss): ");

String dt2 = sc.nextLine();

try {

// casting string to localDateTime

LocalDateTime d1 = LocalDateTime.parse(dt1, DTformatter);

LocalDateTime d2 = LocalDateTime.parse(dt2, DTformatter);

int YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS;

// calculating duration between two dates in 6 units

YEARS = (int) ChronoUnit.YEARS.between(d1, d2);

MONTHS = (int) ChronoUnit.MONTHS.between(d1, d2);

DAYS = (int) ChronoUnit.DAYS.between(d1, d2);

HOURS = (int) ChronoUnit.HOURS.between(d1, d2);

MINUTES = (int) ChronoUnit.MINUTES.between(d1, d2);

SECONDS = (int) ChronoUnit.SECONDS.between(d1, d2);

// calculating years difference

Stringyears = String.valueOf(YEARS) + " years";

// calculating months difference

Stringmonths = String.valueOf(MONTHS - (YEARS \* 12)) + " months";

// calculating days difference

String days = String.valueOf(DAYS - (int) (MONTHS \* 30.436875)) + " days";

// calculating hour difference

String hours = String.valueOf(HOURS - (DAYS \* 24)) + " hrs";

// calculating minute difference

String minutes = String.valueOf(MINUTES - (HOURS \* 60)) + " mins";

// calculating seconds difference

String seconds = String.valueOf(SECONDS - (MINUTES \* 60)) + " secs";

String timegap = years + "; " + months + "; " + days + "; " + hours + "; " + minutes + "; " + seconds;

System.out.println("time gap is: " + timegap);

return timegap;

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}

**7. addtime():** This method takes 5 parameters and is used to calculate the resultant date and time after adding specific time to a given date. It returns a **String** object after calculating the final date after the given time.

It uses **DateTimeFormatter**to validate the input date passed.

Input format: dd/MM/yyyy, years, months, days, hours

signature:

public static StringaddTime(String date1, int hours, int days, int months, int years)

Method:

// method for calculating resultant date after adding time

public static StringaddTime() {

System.out.print("Enter Date in (dd/MM/yyyy): ");

String date1 = sc.next();

System.out.print("Enter years to add: ");

Int years = sc.nextInt();

System.out.print("Enter months to add: ");

int months = sc.nextInt();

System.out.print("Enter days to add: ");

int days = sc.nextInt();

System.out.print("Enter hours to add: ");

int hours = sc.nextInt();

try {

// casting string to localDate

LocalDate localDate = LocalDate.parse(date1, Dformatter);

// casting localDate to Date

Date date = LocalDateToDate(localDate);

// getting an instance of calendar

Calendar cal = Calendar.getInstance();

// casting date to calendar

cal.setTime(date);

// adding hours to date

cal.add(Calendar.HOUR, hours);

cal.getTime();

// adding hourdays to date

cal.add(Calendar.DAY\_OF\_MONTH, days);

cal.getTime();

// adding months to date

cal.add(Calendar.MONTH, months);

cal.getTime();

// adding years to date

cal.add(Calendar.YEAR, years);

// casting calendar to LocalDate and returning in string format

String finalDT = LocalDateTime.ofInstant(cal.toInstant(), zoneId)

.format(DTformatter).toString();

System.out.println("Date and time after adding time: " finalDT);

return finalDT;

} catch (Exceptione) {

e.printStackTrace();

return e.toString();

}

}