The `LocalDateTime` is used to represent a combination of date and time. The classes that we saw in our previous lessons were intended to return only date or time. This class is used when we need a combination of date and time. This class offers a variety of utilities and we will look at some of the most commonly used ones.

a) Getting the current date and time

We can get the current date and time by using the static `now()` method in the `LocalDateTime` class.

```
import java.time.LocalDateTime;

class DateTimeDemo {
    public static void main( String args[] ) {
        LocalDateTime date = LocalDateTime.now();
        System.out.println(date);
    }
}
```

Run Save Reset

b) Getting a specific date and time using of() method

We can get a specific date by using the static `of()` method in the `LocalDateTime` class. This method has two overloaded versions.

Each of them is shown in the example below.

```
import java.time.LocalDateTime;
import java.time.Month;

class DateTimeDemo {
    public static void main(String args[]) {

        // of(int year, int month, int dayOfMonth, int hour, int minute)
        LocalDateTime date = LocalDateTime.of(2019, 05, 03, 12, 34);
        System.out.println(date);

        // of(int year, int month, int dayOfMonth, int hour, int minute, int second)
        date = LocalDateTime.of(2019, Month.AUGUST, 03, 23, 34);
        System.out.println(date);

    }
}
```

Run Save Reset

c) Getting a specific date and time using parse() method

We can get a specific date and time by using the static `parse()` method in the `LocalDateTime` class.

```
import java.time.LocalDateTime;

class DateTimeDemo {
    public static void main( String args[] ) {

        // parse(CharSequence text)
        LocalDateTime date = LocalDateTime.parse("2020-06-20T07:54:00");
        System.out.println(date);

    }
}
```

Run Save Reset

d) Modifying a given date and time.

We can use a whole range of addition and subtraction operation methods to modify the given DateTime.

```
import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;

class DateTimeDemo {
    public static void main( String args[] ) {

        // Adding 4 days to given date and time.
        LocalDateTime date = LocalDateTime.parse("2020-05-12T08:30:00").plusDays(4);
        System.out.println(date);

        // Adding 4 months to given date and time.
        date = LocalDateTime.parse("2020-05-12T08:30:00").plus(4, ChronoUnit.MONTHS);
        System.out.println(date);

        // Subtracting 4 months from given date and time.
        date = LocalDateTime.parse("2020-05-12T08:30:00").minusMonths(4);
        System.out.println(date);

    }
}
```

Run Save Reset