Java Dates

Java does not have a built-in Date class, but we can import the `java.time` package to work with the date and time API. The package includes many date and time classes. For example:

Class	Description
`LocalDate`	Represents a date (year, month, day (yyyy-MM-dd))
`LocalTime`	Represents a time (hour, minute, second and milliseconds (HH-mm-ss-zzz))
`LocalDateTime`	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss.zzz)
`DateTimeFormatter`	Formatter for displaying and parsing date-time objects

Display Current Date

To display the current date, import the `java.time.LocalDate` class, and use its `now()` method:

Example

```
import java.time.LocalDate; // import the LocalDate class


public class MyClass {
  public static void main(String[] args) {
    LocalDate myObj = LocalDate.now(); // Create a date object
    System.out.println(myObj); // Display the current date
  }
}
```

The output will be:

```
2020-04-16
```

Display Current Time

To display the current time (hour, minute, second, and milliseconds), import the `java.time.LocalTime` class, and use its `now()` method:

Example

```
import java.time.LocalTime; // import the LocalTime class


public class MyClass {

  public static void main(String[] args) {

    LocalTime myObj = LocalTime.now();

    System.out.println(myObj);

  }

}
```

The output will be:

```
23:19:11.337744
```

Display Current Date and Time

To display the current date and time, import the `java.time.LocalDateTime` class, and use its `now()` method:

Example

```
import java.time.LocalDateTime; // import the LocalDateTime class


public class MyClass {

  public static void main(String[] args) {

    LocalDateTime myObj = LocalDateTime.now();

    System.out.println(myObj);

  }

}
```

The output will be:

```
2020-04-16T23:19:11.637091
```

Formatting Date and Time

The "T" in the example above is used to separate the date from the time. You can use the `DateTimeFormatter` class with the `ofPattern()` method in the same package to format or parse date-time objects. The following example will remove both the "T" and milliseconds from the date-time:

Example

```
import java.time.LocalDateTime; // Import the LocalDateTime class

import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter class


public class MyClass {

  public static void main(String[] args) {

    LocalDateTime myDateObj = LocalDateTime.now();

    System.out.println("Before formatting: " + myDateObj);

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


    String formattedDate = myDateObj.format(myFormatObj);

    System.out.println("After formatting: " + formattedDate);

  }

}
```

The output will be:

```
Before Formatting: 2020-04-16T23:19:11.638020
After Formatting: 16-04-2020 23:19:11
```

Run example »

The `ofPattern()` method accepts all sorts of values, if you want to display the date and time in a different format. For example:

Value	Example
yyyy-MM-dd	"1988-09-29"
dd/MM/yyyy	"29/09/1988"
dd-MMM-yyyy	"29-Sep-1988"
E, MMM dd yyyy	"Thu, Sep 29 1988"