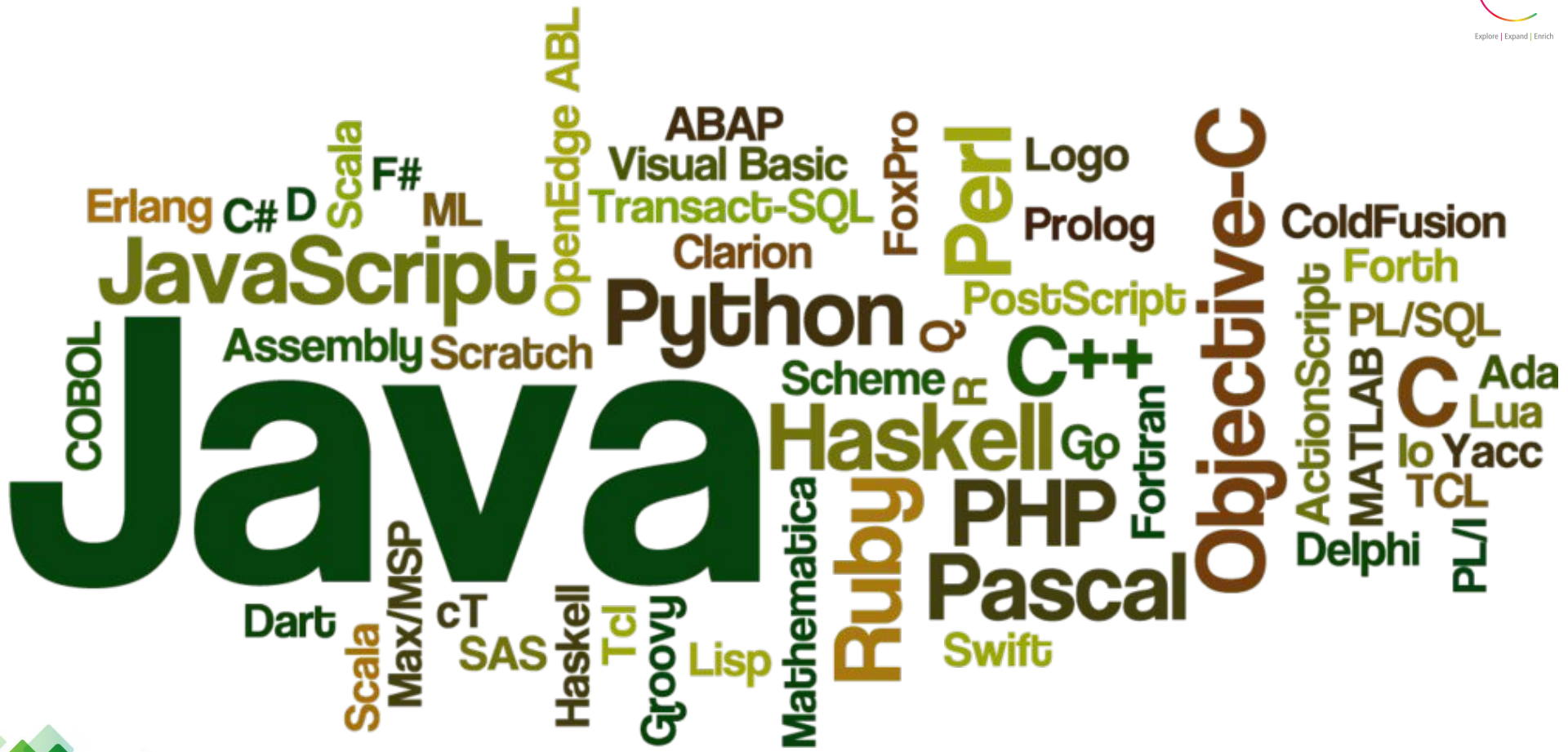
Explore | Expand | Enrich



Date and Time
IN
Java

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

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;
public class MyClass {
     public static void main(String[] args) {
          LocalDate myObj = LocalDate.now();
System.out.println(myObj);
     }
}
```

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;
public class MyClass {
     public static void main(String[] args) {
          LocalTime myObj = LocalTime.now();
          System.out.println(myObj);
     }
}
```

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);
  }
}
```

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

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
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);
     }
}
```

DISPLAY THE ofPattern()

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

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
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);
     }
}
```

DISPLAY THE ofPattern()

- The ofPattern() method accepts all sorts of values

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
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);
     }
}
```

DISPLAY THE ofPattern()

- The ofPattern() method accepts all sorts of values

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
public class MyClass {
     public static void main(String[] args) {
          LocalDateTime myDateObj = LocalDateTime.now();
          System.out.println("Before Formatting: " + myDateObj);
          DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MMM-yyyy
HH:mm:ss");
          String formattedDate = myDateObj.format(myFormatObj);
          System.out.println("After Formatting: " + formattedDate);
     }
}
```

DISPLAY THE ofPattern()

- The ofPattern() method accepts all sorts of values

```
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("E,MMM dd yyyy HH:mm:ss");
    String formattedDate = myDateObj.format(myFormatObj);
    System.out.println("After Formatting: " + formattedDate);
  }
}
```

THE DATE CLASS SUPPORTS TWO CONSTRUCTORS

Sl.No	Constructor & Description
1	`Date( )` This constructor initializes the object with the current date and time.
2	`Date(long millisec)` This constructor accepts an argument that equals the number of milliseconds that have elapsed since midnight, January 1, 1970.
3	`Date(int year, int month, int date)`
4	`Date(int year, int month, int date, int hrs, int min)`
5	`Date(int year, int month, int date, int hrs, int min, int sec)`
6	`Date(String s)`

Date()

- The ofPattern() method accepts all sorts of values

```
import java.util.*;
public class Main
{
     public static void main(String[] args)
     {
          Date d1 = new Date();
          System.out.println("Current date is " + d1);
          Date d2 = new Date(2323223232L);
          System.out.println("Date represented is "+ d2 );
     }
}
```

FOLLOWING ARE THE METHODS OF THE DATE CLASS

Sl.No	Constructor & Description
1	`boolean after(Date date)` Returns true if the invoking Date object contains a date that is later than the one specified by date, otherwise, it returns false.
2	`boolean before(Date date)` Returns true if the invoking Date object contains a date that is earlier than the one specified by date, otherwise, it returns false.
3	`Object clone( )` Duplicates the invoking Date object.
4	`boolean equals(Object date)` Returns true if the invoking Date object contains the same time and date as the one specified by date, otherwise, it returns false.


THANK YOU