Date & Time

java.util.TimeZone

Timezone trong Java được sử dụng để biểu diễn múi giờ.

Để lấy TimeZone mặc định nơi chương trình bạn đang chạy, có thể sử dụng phương thức **getDefault()**

Các phương thức trong lớp TimeZone

Method	Description
`static String[] getAvailableIDs()`	Sử dụng để lấy toàn bộ ID có sẵn được hỗ trợ
`static TimeZone getDefault()`	Lấy TimeZone ở máy chủ hiện tại
`String getDisplayName()`	Trả về tên của múi giờ
`String getID()`	Trả về ID của múi giờ
`int getOffset(long date)`	Trả về offset của múi giờ
`void setID(String ID)`	Thiết lập ID cho múi giờ

java.util.Calendar

Calendar trong Java là một lớp trừu tượng nằm trong package java.util, nó cung cấp các phương thức để chuyển đổi giữa ngày với một thời điểm cụ thể trong thời gian và một tập hợp các trường của calendar như MONTH, YEAR, HOUR,DAY_OF_MONTH,...

java.util.Calendar

Để khởi tạo Calendar có thể sử dụng các phương thức static:

- Calendar.getInstance()
- Calendar.getInstance(TimeZone zone)
- Calendar.getInstance(Locale aLocale)
- Calendar.getInstance(TimeZone zone, Locale aLocale)

java.util.Date

Lớp java.util.Date đại diện cho một thời điểm cụ thể. Nó cung cấp các constructor và phương thức để xử lý ngày và giờ trong Java

Tuy nhiên hầu hết các phương thức của nó đã lỗi thời và được khuyến cáo là không nên sử dụng. Mặc dù vậy nhưng lớp Date này vẫn được sử dụng rộng rãi

java.util.Date

Để khởi tạo Date ta có thể sử dụng 1 trong 2 constructor:

Constructor	Description
Date()	Tạo ra đối tượng Date đại diện cho ngày và giờ hiện tại
Date(long millisecond)	Tạo ra đối tượng Date theo thời gian milli giây tính từ 01/01/1997

java.util.Date

```
public static void main(String[] args) {
    Date date1 = new Date();
    System.out.println(date1);

    //Số milli giây tính từ 01/01/1970 đến hiện tại
    long milliseconds = System.currentTimeMillis();
    Date date2 = new Date(milliseconds);
    System.out.println(date2);
}
```

java.sql.Date

Lớp Date trong package java.sql.Date chỉ biểu diễn ngày. Được sử dụng chủ yếu trong JDBC vì nó thể hiện ngày tháng được lưu trong database.

Hầu hết các constructor và method của lớp này đều bị khuyến cáo là không nên sử dụng

java.sql.Date

Để khởi tạo đối tượng Date ta có thể sử dụng constructor sau:

Date(long milliseconds) : Tạo đối tượng sql Date với mili giây đã cho tính từ ngày 1 tháng 1 năm 1970, 00:00:00 GMT.

```
public class Main {
    public static void main(String[] args) {
        long milliseconds = System.currentTimeMillis();
        Date date = new Date(milliseconds);
        System.out.println(date);
    }
}
```

Một số phương thức trong lớp Date

Methods	Description
`void setTime(long time)`	Cập nhật lại sql date với time chỉ định
`Instant toInstant()`	Chuyển đổi sql date thành đối tượng `Instant`
`LocalDate toLocalDate()`	Chuyển sql date thành đối tượng `LocalDate`
`String toString()`	Chuyển sql date thành chuỗi `String`
`static Date valueOf(LocalDate date)`	Trả về đối tượng `Date` từ `LocalDate` chỉ định
`static Date valueOf(String date)`	Trả về đối tượng `Date` từ chuỗi `String` chỉ định

Package java.time

LocalDate

LocalDate là một lớp bất biến đại diện cho ngày với định dạng mặc định là yyyy-MM-dd

Để khởi tạo LocalDate object ta có thể sử dụng các method static là **now()** và **of()**

Với **now()** sẽ lấy ra ngày hiện tại, **of()** giúp bạn chỉ định một ngày cụ thể

LocalDate

```
LocalDate localDate1 = LocalDate.now();

LocalDate localDate2 = LocalDate.of(2022, 7, 10);

LocalDate localDate3 = LocalDate.of(2022, Month.JULY, 20);
```

LocalTime

Lớp Localtime là một lớp bất biến đại diện cho thời gian (giờ, phút,giây,...) trong một ngày.

Để khởi tạo LocalTime object ta có thể sử dụng hai phương thức static là **now()** và **of()**

Với **now()** giúp lấy ra thời gian hiện tại, **of()** chỉ định thời gian cụ thể dựa theo tham số truyền vào.

LocalTime

```
LocalTime localTime1 = LocalTime.now();

LocalTime localTime2 = LocalTime.of(12, 45);

LocalTime localTime3 = LocalTime.of(12, 45, 30);

LocalTime localTime4 = LocalTime.of(12, 45, 30, 1000);
```

LocalDateTime

LocalDateTime đại diện cho cả ngày và giờ trong hệ thống lịch ISO

Tương tự LocalDate và LocalTime, LocalDateTime cũng cung cấp 2 phương thức static là **now()** và **of()** để tạo đối tượng.

LocalDateTime

```
LocalDateTime localDateTime1 = LocalDateTime.now();

LocalDate localDate = LocalDate.now();
LocalTime localTime = LocalTime.now();
LocalDateTime localDateTime2 = LocalDateTime.of(localDate, localTime);

LocalDateTime localDateTime3 = LocalDateTime.of(2022, 8, 12, 6, 30);
```

DateTimeFormatter

Bạn có thể sử dụng lớp DateTimeFormatter và phương thức ofPattern() trong cùng một package để định dạng hoặc phân tích cú pháp các đối tượng date-time.

DateTimeFormatter

```
public class Main {
    public static void main(String[] args) {
        LocalDateTime myDateObj = LocalDateTime.now();
        String pattern = "dd-MM-yyyy HH:mm:ss";
        System.out.println("Before formatting: " + myDateObj);
        DateTimeFormatter myFormatObj =
DateTimeFormatter.ofPattern(pattern);

        String formattedDate = myDateObj.format(myFormatObj);
        System.out.println("After formatting: " + formattedDate);
    }
}
```