Date Format

JSR 310, part of Java SE 8, introduced the new date and time API in the java.time package. It aims to provide a comprehensive and flexible date-time handling mechanism. Here's how you can format dates using the java.time package:

Importing Required Classes

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
```

Formatting a LocalDate

```
// Create a LocalDate instance
LocalDate date = LocalDate.of(2024, 7, 23);

// Define a formatter
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

// Format the date
String formattedDate = date.format(formatter);
System.out.println(formattedDate);  // Output: 23/07/2024
```

Formatting a LocalDateTime

```
// Create a LocalDateTime instance
LocalDateTime dateTime = LocalDateTime.of(2024, 7, 23, 14, 30);

// Define a formatter
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy
HH:mm");

// Format the date-time
String formattedDateTime = dateTime.format(formatter);
System.out.println(formattedDateTime);  // Output: 23/07/2024 14:30
```

Common Date-Time Patterns

Common Date-Time Patterns

1. **Basic Date Patterns**

    - `dd/MM/yyyy` - Day/Month/Year (e.g., 23/07/2024)
    - `MM/dd/yyyy` - Month/Day/Year (e.g., 07/23/2024)
    - `yyyy-MM-dd` - Year-Month-Day (e.g., 2024-07-23)

2. **Including Time**

    - `dd/MM/yyyy HH:mm` - Day/Month/Year Hour

        - (24-hour format) (e.g., 23/07/2024 14:30)
    - **`MM/dd/yyyy hh:mm a`** - Month/Day/Year Hour
        - AM/PM (12-hour format) (e.g., 07/23/2024 02:30 PM)
- `yyyy-MM-dd'T'HH:mm:ss` - ISO 8601 format (e.g., 2024-07-23T14:30:00)

3. **Extended Date-Time Patterns**

- `EEEE, MMMM d, yyyy` - Day of week, Month day, Year (e.g., Tuesday, July 23, 2024)
- `EEE, MMM dd, yyyy HH:mm:ss` - Day of week (short), Month (short) day, Year Hour:Minute

(e.g., Tue, Jul 23, 2024 14:30:00)

- `dd-MMM-yyyy` - Day-Month (short name)-Year (e.g., 23-Jul-2024)

4. **Including Time Zone**

- `yyyy-MM-dd HH:mm:ss Z` - Year-Month-Day Hour:Minute

Time zone offset (e.g., 2024-07-23 14:30:00 +0000)

- `yyyy-MM-dd HH:mm:ss z` - Year-Month-Day Hour:Minute

Time zone abbreviation (e.g., 2024-07-23 14:30:00 GMT)

Parsing Dates

```
// Define a formatter
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

// Parse a string to LocalDate
LocalDate date = LocalDate.parse("23/07/2024", formatter);
System.out.println(date);  // Output: 2024-07-23
```

Handling Different Locales

```
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

// Create a LocalDate instance
LocalDate date = LocalDate.now();

// Define a formatter with Locale
DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL)
                                                .withLocale(Locale.FRENCH);

// Format the date
String formattedDate = date.format(formatter);
System.out.println(formattedDate);  // Output: mardi 23 juillet 2024
```

By using `java.time` package and `DateTimeFormatter`, you can flexibly and effectively format and parse dates in Java.