Introduzione a LocalDate

`LocalDate` è una classe introdotta in Java 8 per rappresentare una data senza ora né fuso orario.

Metodi Principali di `LocalDate`

1. `parse()`

Converte una stringa in un oggetto `LocalDate`.

```
LocalDate data = LocalDate.parse("2025-03-19");
System.out.println(data); // 2025-03-19
```

2. `isBefore()` *(estremo escluso)*

Verifica se una data è **strettamente precedente** a un'altra data (la data stessa non è compresa).

```
LocalDate oggi = LocalDate.now();
LocalDate altraData = LocalDate.parse("2025-12-31");

oggi.isBefore(altraData);
// restituisce true se oggi è prima del 31 dicembre 2025 (31 dicembre escluso)
```

3. `isAfter()` *(estremo escluso)*

Verifica se una data è **strettamente successiva** a un'altra data (la data stessa non è compresa).

```java
LocalDate oggi = LocalDate.now();
LocalDate altraData = LocalDate.parse("2024-01-01");

oggi.isAfter(altraData);
// restituisce true se oggi è dopo il 1 gennaio 2024 (1 gennaio escluso)
```

Se vuoi includere anche gli estremi nelle verifiche delle date, puoi usare una combinazione dei metodi `isEqual()`, `isBefore()` e `isAfter()`:

Estremi compresi (range inclusivo):

```java
LocalDate dataInizio = LocalDate.parse("2023-05-01");
LocalDate dataFine = LocalDate.parse("2023-05-09");
LocalDate dataTest = LocalDate.now();

boolean compreso = (dataTest.isAfter(dataInizio) || dataTest.isEqual(dataInizio))
                && (dataTest.isBefore(dataFine) || dataTest.isEqual(dataFine));

System.out.println(compreso);
```

Questo codice verifica se la data `dataTest` è compresa tra le due date specificate **inclusi gli estremi**.

4. `now()`

Restituisce la data attuale.

```java
LocalDate oggi = LocalDate.now();
System.out.println(oggi); // data corrente es. 2025-03-19
```