**Introdução**

A classe LocalDateTime está localizada no pacote java.time, desde a versão 1.8.

LocalDateTime é um objeto de data e hora imutável que representa uma data e hora com formato padrão como aaaa-MM-dd-HH-mm-ss.zzz. Ele fornece um método de fábrica que usa os argumentos de entrada LocalDate e LocalTime para criar a instância LocalDateTime.

Abaixo segue alguns métodos que serão aplicados em cenários como:

**Métodos**

**public static LocalDateTime now()** -> Obtém a data e hora atual do relógio do sistema no fuso horário padrão. Esse método pode ser sobrecarregado. Esse método não precisa de uma instância.

**Retorno:** a data e hora atual usando o relógio do sistema e o fuso horário padrão, não nulo.

**public static LocalDateTime parse(CharSequence** text**)** -> Obtém uma instância de LocalDateTime de uma string de texto como 2007-12-03T10:15:30. Esse método pode ser sobrecarregado. Esse método não precisa de uma instância.

**Retorno:** a data e hora local, não nula.

**public DayOfWeek getDayOfWeek()** ->Obtém o campo de dia da semana, que é uma enumeração DayOfWeek.

**Retorno:** Uma enumeração do dia da semana. Não será nulo. Os dias são: FRIDAY, MONDAY, SATURDAY, SUNDAY, THURSDAY, TUESDAY, WEDNESDAY

**public boolean isAfter(ChronoLocalDateTime<?> other)** -> Verifica se esta data-hora é posterior à data-hora especificada.

**Retorno:** true se esta data e hora for posterior à data e hora especificada, caso contrário, false.

**Exemplos**

Obter a data e hora atual:

**LocalDateTime** agora = **LocalDateTime.now();**

Obter uma data e hora a partir de uma String:

**LocalDateTime** dataHora = **LocalDateTime.parse**("2007-12-03T10:15:30");

Obter o dia da semana de uma determinada data:

**DayOfWeek** diaSemana = dataHora.**getDayOfWeek();**

Verificar se uma data é posterior à outra:

**Boolean** resultado = agora**.isAfter**(amanha);