

Data e hora

Data e hora

Para trabalhar com datas e horas no Java utilizamos normalmente a API **java.time**, introduzida na versão Java 8, e que possui diversas classes e métodos para simplificar o trabalho com esse tipo de dado. As principais classes são:

- **LocalDate**
- **LocalDateTime**
- **LocalTime**

LocalDate

Usamos `LocalDate` para representar uma data sem informação de fuso horário, e temos algumas formas de atribuir valores a uma variável que seja desse tipo:

.now obtém a data atual

```
LocalDate dataCompra = LocalDate.now();  
LocalDate dataPrimeiraParcela = LocalDate.of(2025, 5, 15);  
LocalDate dataLimitePagamento = LocalDate.parse("2025-06-01");
```

.parse consegue converter uma `String` para data, desde que escrita no padrão ISO

.of atribui uma data específica usando valores inteiros para ano, mês e dia

LocalDateTime

Usamos `LocalDateTime` para incluir, além da data, a hora, também sem informação de fuso horário. A forma de atribuição é muito similar:

```
LocalDateTime agora = LocalDateTime.now();  
LocalDateTime inicioEvento = LocalDateTime.of(2025, 5, 15, 14, 30);  
LocalDateTime fimEvento = LocalDateTime.parse("2025-05-15T17:30");
```

.now é igual ao anterior, sem alterações

.parse também precisará contemplar no padrão ISO as horas e minutos

.of aqui precisa de mais dois parâmetros para representar a hora e os minutos

LocalTime

Usamos LocalTime para trabalhar apenas com horas, sem informação de fuso horário. A forma de atribuição também é muito similar:

```
LocalTime agora = LocalTime.now();  
LocalTime hrInicioTrabalho = LocalTime.of(14, 30);  
LocalTime hrFimTrabalho = LocalTime.parse("17:30");
```

.now é igual aos anteriores, sem alterações

.parse também precisará contemplar no padrão ISO somente as horas e minutos

.of aqui precisa apenas de dois parâmetros para representar a hora e os minutos

Manipulação de Datas e Horas

Podemos usar alguns métodos de verificação de ordem, para conferir se uma data está antes ou depois da outra, com os métodos **isBefore** ou **isAfter**, respectivamente.

.isBefore vai comparar se a data/hora da variável *agora* é anterior à data/hora da variável *hrInicioTrabalho*. Se positivo, a variável **antes** ficará com o valor **true**.

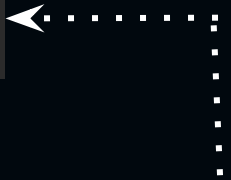
```
boolean antes = agora.isBefore(hrInicioTrabalho);  
boolean depois = hrFimTrabalho.isAfter(hrInicioTrabalho);
```

.isAfter vai comparar se a data/hora da variável *hrFimTrabalho* é posterior à data/hora da variável *hrInicioTrabalho* e alimentar a variável **depois** com **true** ou **false**

Cálculo de Datas e Horas

Podemos usar alguns métodos para acrescentar ou diminuir dias, meses, anos e/ou horas a uma data/hora através de métodos iniciados por *plus* ou *minus*.

```
LocalDate dataCompra = LocalDate.now();  
LocalDate dataPrimeiraParcela = dataCompra.plusDays(30);
```



.plusDays vai acrescentar 30 dias à variável *dataCompra* e atribuir a nova data à variável *dataPrimeiraParcela*

Cálculo de Datas e Horas

Podemos usar alguns métodos para acrescentar ou diminuir dias, meses, anos e/ou horas a uma data/hora através de métodos iniciados por *plus* ou *minus*.

```
LocalDate inicioTrimestre = LocalDate.now().minusMonths(3);
```

.minusMonths vai subtrair 3 meses de uma variável, ou no caso desse exemplo, 3 meses da data atual e atribuir a nova data à variável *inicioTrimestre*

Cálculo de Datas e Horas

Podemos utilizar também as classes **Period** e **Duration** para fazer cálculos utilizando datas e horas.

```
Period diferenca = Period.between(dataPassada, dataAtual);  
int anos = diferenca.getYears();  
int meses = diferenca.getMonths();  
int dias = diferenca.getDays();
```

Criamos uma variável do tipo *Period* chamada **diferenca**, que irá receber o período entre duas data. Com isso conseguimos verificar quantos *anos*, *meses* e *dias* de diferença há entre elas.

Cálculo de Datas e Horas

Podemos utilizar também as classes **Period** e **Duration** para fazer cálculos utilizando datas e horas.

```
Period diferenca = Period.between(dataPassada, dataAtual);  
int anos = diferenca.getYears();  
int meses = diferenca.getMonths();  
int dias = diferenca.getDays();
```

Criamos uma variável do tipo *Period* chamada **diferenca**, que irá receber o período entre duas data. Com isso conseguimos verificar quantos *anos*, *meses* e *dias* de diferença há entre elas.

Formatação de dados

- **DateTimeFormatter**
- **ZoneDateTime**

Cálculo de diferença entre datas

- Duration
- Period

Compartilhe um resumo de seus novos
conhecimentos em suas redes sociais.

[#aprendizadoalura](#)

alura



Escola Programação