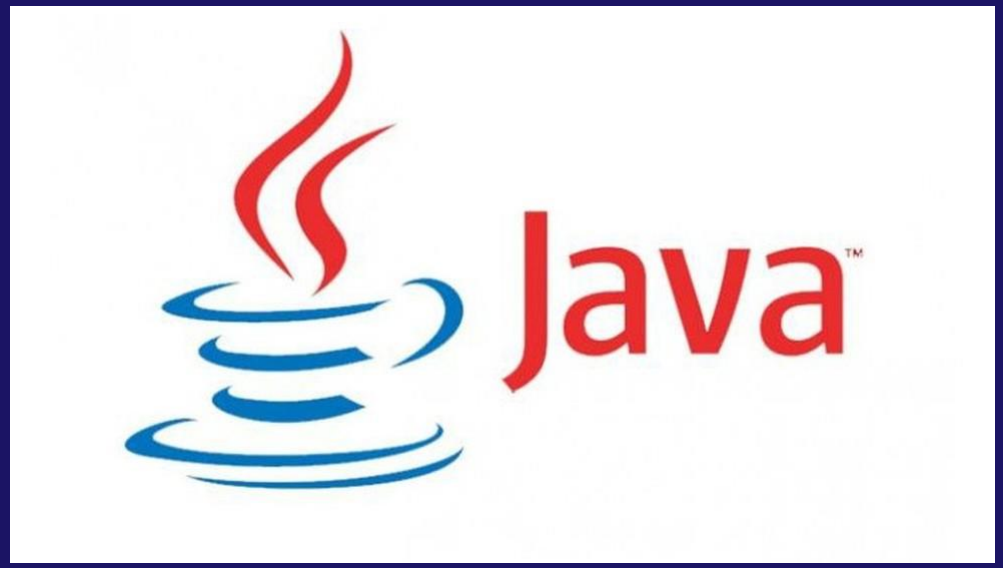
Curso Técnico em Desenvolvimento de Sistemas

UCR8 – Desenvolvimento de Interface Gráfica - Desktop

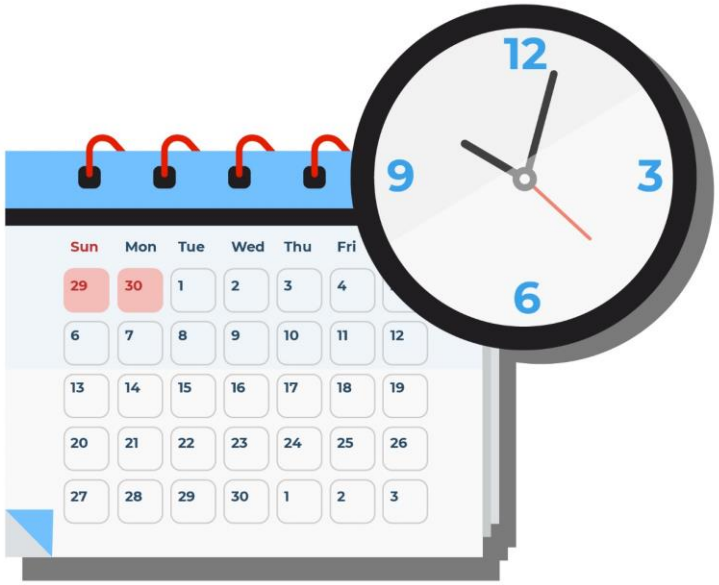
Aula 19 – Operações com Base de Tempo



Operações com Datas e Horas

- Todo programador deve saber operar datas e horas.

- Java possui diversas classes: *LocalDate, LocalTime, LocalDateTime; Duration e Period;*



Operações com Datas

Data Atual do sistema:

```
LocalDate hoje = LocalDate.now();
System.out.println(x: hoje);
```

```
run:
2023-10-23
```

Criar um objeto data:

```
LocalDate finados = LocalDate.of(year: 2023, month: 11, dayOfMonth:2);
System.out.println(x: finados);
```

```
run:
2023-11-02
```

Principais Métodos:

Adição de tempo:

```
LocalDate ferProcRep = LocalDate.of(year:2023, month:11, dayOfMonth:5);
System.out.println(x:ferProcRep.plusDays(daysToAdd:5));      // 2023-11-10
System.out.println(x:ferProcRep.plusWeeks(weeksToAdd:5));    // 2023-12-10
System.out.println(x:ferProcRep.plusMonths(monthsToAdd:5)); // 2024-04-05
System.out.println(x:ferProcRep.plusYears(yearsToAdd:5));    // 2028-11-05
```

Subtração de tempo:

```
LocalDate ferProcRep = LocalDate.of(year:2023, month:11, dayOfMonth:5);
System.out.println(x:ferProcRep.minusDays(daysToSubtract:5));      // 2023-10-31
System.out.println(x:ferProcRep.minusWeeks(weeksToSubtract:5));    // 2023-10-01
System.out.println(x:ferProcRep.minusMonths(monthsToSubtract:5)); // 2023-06-05
System.out.println(x:ferProcRep.minusYears(yearsToSubtract:5));    // 2018-11-05
```

Principais Métodos:

ChronoUnit

```
System.out.println(x: ferProcRep.plus(amountToAdd: 1, unit:ChronoUnit.CENTURIES));
```

Unidades de tempo referenciadas em ChronoUnit:

```
ChronoUnit.NANOS; // (1/1.000.000.000) seg
ChronoUnit.MICROS; // (1/1.000.000) seg
ChronoUnit.MILLIS; // (1/1.000) seg
ChronoUnit.SECONDS; // 1 segundo
ChronoUnit.MINUTES; // 1 minuto
ChronoUnit.HOURS; // 1h
ChronoUnit.HALF_DAYS; // 12h
ChronoUnit.DAYS; // 1 dia
```

```
ChronoUnit.WEEKS; // 7 dias
ChronoUnit.MONTHS; // 1 mês
ChronoUnit.YEARS; // 1 ano
ChronoUnit.DECADES; // 10 anos
ChronoUnit.CENTURIES; // 100 anos
ChronoUnit.MILLENNIA; // 1.000 anos
ChronoUnit.ERAS; // 1.000.000.000 anos
ChronoUnit.FOREVER; // valor artificial
```

Principais Métodos:

Verificação de um objeto:

```
LocalDate ferProcRep = LocalDate.of(year:2023, month:11, dayOfMonth:15);
System.out.println(x:ferProcRep.getDayOfWeek());  //WEDNESDAY
System.out.println(x:ferProcRep.getDayOfMonth());  // 15
System.out.println(x:ferProcRep.getMonth()); // NOVEMBER
System.out.println(x:ferProcRep.getMonthValue()); // 11
System.out.println(x:ferProcRep.getYear()); // 2023
System.out.println(x:ferProcRep.getLong(field:ChronoField.DAY_OF_YEAR)); // 319
```

Operações com Horas

Hora Atual do sistema:

```
LocalTime agora = LocalTime.now();
System.out.println(x: agora);
```

```
run:
18:34:58.856354400
```

Criar um objeto Hora:

```
LocalTime inicio = LocalTime.of(hour:19, minute: 02);
System.out.println(x: inicio);
```

```
run:
19:02
```

Operações com Horas

Principais Métodos:

Adição de horas:

```
LocalTime janta = LocalTime.of(hour:20, minute: 40);
System.out.println(x: janta.plusMinutes(minutesToAdd: 20));  // 21:00
System.out.println(x: janta.plusHours(hoursToAdd: 2)); // 22:40
System.out.println(x: janta.plusSeconds(secondstoAdd: 50)); // 20:40:50
System.out.println(x: janta.plus(amountToAdd: 500, unit:ChronoUnit.SECONDS)); // 20:48:20
```

Subtração de horas:

```
LocalTime janta = LocalTime.of(hour:20, minute: 40);
System.out.println(x: janta.minusMinutes(minutesToSubtract:20));  // 20:20
System.out.println(x: janta.minusHours(hoursToSubtract: 2)); // 18:40
System.out.println(x: janta.minusSeconds(secondsToSubtract:50)); // 20:39:10
System.out.println(x: janta.minus(amountToSubtract: 500, unit:ChronoUnit.SECONDS)); // 20:31:40
```

Principais Métodos:

Verificação de um objeto:

```
LocalTime janta = LocalTime.of(hour:20, minute: 40);
System.out.println(x: janta.getSecond());    // 0
System.out.println(x: janta.getMinute());    // 40
System.out.println(x: janta.getLong(field: ChronoField.SECOND_OF_DAY)); // 74400
```

Comparações de um objeto:

```
LocalTime almoco = LocalTime.of(hour:12, minute: 30);
System.out.println(x: janta.isAfter(other: almoco));    // true
System.out.println(x: janta.isBefore(other: almoco));   // false
```

Operações com Datas e Horas

Data e Hora Atual do sistema:

```
LocalDateTime hojeAgora = LocalDateTime.now();
System.out.println(x: hojeAgora);
```

```
run:
2023-10-26T17:41:53.819886200
BUILD SUCCESSFUL (total time: 2 seconds)
```

Criar um objeto Data e Hora:

```
LocalDateTime nascerSol =
        LocalDateTime.of(year: 2022, month: 10, dayOfMonth:26, hour: 05, minute: 44, second: 00);
```

```
run:
2022-10-26T05:44
BUILD SUCCESSFUL (total time: 2 seconds)
```

Operações com Datas e Horas

Principais Métodos:

```
LocalDateTime nascerSol =
        LocalDateTime.of(year: 2022, month: 10, dayOfMonth:26, hour: 05, minute: 44, second: 00);

System.out.println(x: nascerSol);  // 2022-10-26T05:44
System.out.println(x: nascerSol.getDayOfMonth()); // 26
System.out.println(x: nascerSol.plusDays(days: 2)); // 2022-10-28T05:44
System.out.println(x: nascerSol.minusMinutes(minutes: 30)); // 2022-10-26T05:14
System.out.println(x: nascerSol.toLocalDate().isLeapYear()); // false
```

- *Método: isLeapYear -> retorna se o ano da data criada é bissexto (true) ou não (false)*

Formatadores de Datas e Horas

- Por padrão, o retorno de datas e horas do Java, considera o padrão americano, iniciando pelo ano, mês e finalizando pelo número do dia.

```
run:
2023-10-23
```

- Precisamos converter esses dados em um formato comum aos Brasil, o qual utiliza o formato dd/mm/aaaa e suas variações, sempre iniciando pelo número do dia.

- Java possui uma série de recursos otimizados de formatação de datas e horários que podem ser utilizados para estas conversores. É necessário importação da biblioteca de funções.

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

Formatadores de Datas e Horas

Observe o mecanismo de formatação da apresentação da data:

```
public static void main(String[] args) {

    LocalDateTime hojeAgora = LocalDateTime.now();
    System.out.println(x: hojeAgora);

    DateTimeFormatter formDataHora1
            = DateTimeFormatter.ofPattern(pattern:"E, dd/MM/yyyy, HH:mm:ss");

    System.out.println(x: hojeAgora.format(formatter: formDataHora1));
}
```

- *Dados formatados na segunda linha do retorno do sistema.*

```
run:
2023-10-26T18:01:35.781070300
qui., 26/10/2023, 18:01:35
BUILD SUCCESSFUL (total time: 1 second)
```

Formatadores de Datas e Horas

Segunda opção de formatação da apresentação da data:

```
public static void main(String[] args) {

    LocalDateTime hojeAgora = LocalDateTime.now();
    System.out.println(x: hojeAgora);

    DateTimeFormatter formDataHora2
            = DateTimeFormatter.ofPattern
    (pattern:"EEEE, dd 'de' MMMM 'de' yyyy, 'às' HH:mm:ss");

    System.out.println(x: hojeAgora.format(formatter: formDataHora2));
}
```

- *Dados formatados na segunda linha do retorno do sistema.*

```
run:
2023-10-26T18:06:33.828510
quinta-feira, 26 de outubro de 2023, às 18:06:33
BUILD SUCCESSFUL (total time: 0 seconds)
```

Formatadores de Datas e Horas

- *Padrão adotado para formatação (pattern) utilizado dentro do System.out...*

```
(pattern:"EEEE, dd 'de' MMMM 'de' yyyy, 'às' HH:mm:ss");
```

Letra	Componente de Data ou Hora	Apresentação	Exemplos
y	Ano (yy, yyyy)	Ano	1996; 96
M	Mês (MM, MMMM)	Mês	Julho; Jul; 07
w	Semana no ano	Número	27
W	Semana no mês	Número	2
D	Dia no ano	Número	189
d	Dia no mês	Número	10
F	Dia da semana no mês	Número	2
E	Nome do dia da semana (E, EEEE)	Texto	terça-feira; ter.
u	Número do dia da semana (1 a 7)	Número	1
a	Marcador AM/PM	Texto	PM
H	Hora no dia (0 a 23)	Número	0
k	Hora no dia (1 a 24)	Número	24
K	Hora em AM/PM (0 a 11)	Número	0
h	Hora em AM/PM (1 a 12)	Número	12
m	Minuto em uma hora	Número	30
s	Segundos em um minuto	Número	55
S	Milissegundos em um segundo	Número	978
z	Fuso horário	Geral	Pacific Standard Time; PST; GMT-08:00
Z	Fuso horário	RFC 822	-800
X	Fuso horário	ISO 8601	-08; -0800; -08:00

Zone DateTime.

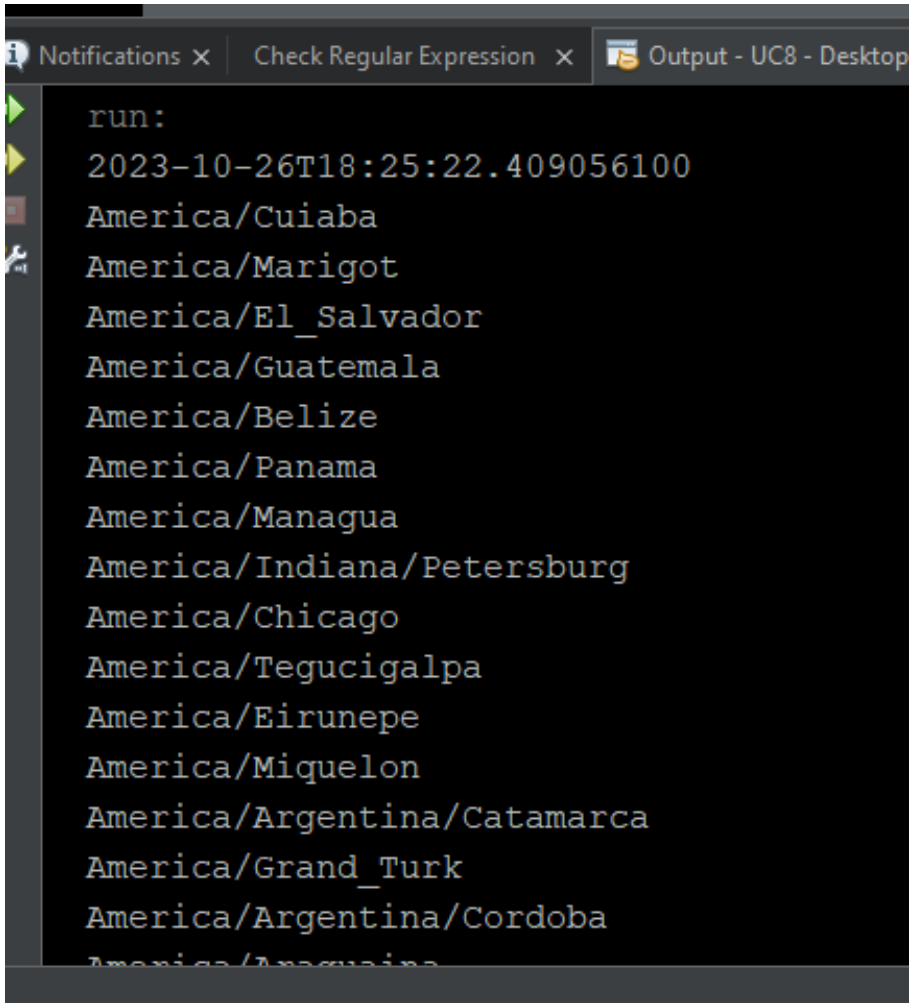
- É possível estabelecer o horário, pela região de localização do globo.

```java
public static void main(String[] args) {

    Set<String> regioes = ZoneId.getAvailableZoneIds();

    for (String regiao : regioes) {
        if (regiao.contains(s: "America")) {
            System.out.println(x: regiao);
        }
    }
}
```

```
run:
America/Sao_Paulo
BUILD SUCCESSFUL (total time: 0 seconds)
```



Zone DateTime.

- Obtendo o fuso a partir da região.

```java
Set<String> regioes = ZoneId.getAvailableZoneIds();

for (String regiao : regioes) {
    if (regiao.contains(s: "Berlin")) {
        System.out.println(x: regiao);
    }
}
```

```java
LocalDateTime hojeAgora = LocalDateTime.now();
ZoneId horarioBrasilia = ZoneId.of(zoneId: "America/Sao_Paulo");
ZonedDateTime fusoBrasilia = ZonedDateTime.of(localDateTime: hojeAgora, zone: horarioBrasilia);
System.out.println(x: fusoBrasilia);

ZoneId horarioBerlin = ZoneId.of(zoneId: "Europe/Berlin");
ZonedDateTime fusoBerlin = ZonedDateTime.of(localDateTime: hojeAgora, zone: horarioBerlin);
System.out.println(x: fusoBerlin);
```

```
run:
Europe/Berlin
2023-10-26T18:38:31.598854500-03:00[America/Sao_Paulo]
2023-10-26T18:38:31.598854500+02:00[Europe/Berlin]
BUILD SUCCESSFUL (total time: 0 seconds)
```