POO - PROGRAMAÇÃO ORIENTADA A OBJETOS

Trabalhando com Datas e Períodos de Tempos

Rodrigo R Silva

Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense
Campus Bagé

TADS
Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Nesta Aula Veremos...

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Introdução

Conceito

- Java consegue manipular as datas, horas e internacionalizá-las.
- No Java o tempo é representado em milissegundos, sendo medido a partir da data 01/01/1970. Essa data pode ser recuperada através do comando: **System.currentTimeMillis();**

Classe Date

- A data representa o tempo, um tempo é composto por ano, mês, dia atual, minuto atual, entre outras propriedades que essa classe possui.
- Hoje a maioria dos métodos da classe Date estão classificados como deprecated (depreciados), ou seja, são métodos que não são mais utilizados, por isso essa classe foi substituída pela classe Calendar, para haver suporte correto à internacionalização do sistema de datas.

```
1         Date date = new Date();
2
3         System.out.println(date);
4
5         System.out.println("Data em milesegundos "+date.getTime());
6
7         System.out.println("Dia do m s "+date.getDate());
8
9         System.out.println("Ano "+ (date.getYear()+1900));
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

After / Before

- O método **before()** na classe Java Date que realiza a comparação do objeto Date e o dado Date instantâneo. Ele retorna true quando o objeto Date vem mais cedo do que o objeto Date dado.
- Outra abordagem para conseguir esta comparação é utilizar o método **after()** na classe Java Date. Este método é oposto ao método anterior e retorna true quando o objeto Date vem mais tarde do que o objeto Date dado.

```
1         simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy");
2
3         Date dataVencimentoBoleto = simpleDateFormat.parse("15/12/2021");
4
5         Date dataAtual = simpleDateFormat.parse("16/11/2021");
6
7         //After: data 1 maior que data 2
8         //before: data 1 menor que data 2
9         if(dataVencimentoBoleto.after(dataAtual)) {//data 1 maior q a
      data 2?
10            System.out.println("Boleto n o vencido");
11        }else {
12            System.out.println("Boleto vencido");
13        }
14
15        if(dataVencimentoBoleto.before(dataAtual)) {//data 1 menor q a
      data 2?
16            System.out.println("Boleto vencido");
17        }else {
18            System.out.println("Boleto n o vencido");
19        }
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Conversões

SimpleDateFormat

- Às vezes é preciso transformar um texto em uma data ou vice versa, para isso abaixo é exibida a função parse e a classe **SimpleDateFormat**. Geralmente a classe **SimpleDateFormat** é mais usada quando trata-se de formatação de datas, pois já no seu construtor, quando instanciada, permite passar como argumento o formato da data desejada, como apresentada nos exemplos abaixo.

```
1         SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd/MM/
      yyyy");
2
3         System.out.println("Data atual: "+simpleDateFormat.format(date));
4
5         simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
6
7         System.out.println("Data para BD: "+simpleDateFormat.format(date)
      );
8
9         simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
10
11        System.out.println("Data com horas: "+simpleDateFormat.format(
      date));
12
13        simpleDateFormat.parse("1982-07-08 20:10:07");
14
15        System.out.println(simpleDateFormat.parse("1982-07-08 20:10:07"))
      ;
16
17        simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy");
18
19        System.out.println(simpleDateFormat.parse("08/07/1982"));
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

ChronoUnit

ChronoUnit

- Um conjunto padrão de unidades de períodos de data.
- Este conjunto de unidades fornece acesso baseado em unidade para manipular uma data, hora ou data-hora. O conjunto padrão de unidades pode ser estendido implementando TemporalUnit.
- Essas unidades devem ser aplicadas em vários sistemas de calendário. Por exemplo, a maioria dos sistemas de calendário não ISO define unidades de anos, meses e dias, apenas com regras ligeiramente diferentes. A documentação de cada unidade explica como ela funciona.

```
1        long dias = ChronoUnit.DAYS.between(LocalDate.parse("2021-01-01")
     , LocalDate.now());
2
3        System.out.println("Passaram "+dias+" dias...");
4
5        long meses = ChronoUnit.MONTHS.between(LocalDate.parse("
     2021-01-01"), LocalDate.now());
6
7        System.out.println("Passaram "+meses+" meses...");
```

Calendar

Classe Calendar

- Essa classe pode produzir os valores de todos os campos de calendário necessários para implementar a formatação de data e hora, para uma determinada língua e estilo de calendário. Por exemplo, japonês, americano, italiano, brasileiro entre outros.
- A classe Calendar é a mais usada quando se trata de datas, mas como é uma classe abstrata, não pode ser instanciada, portanto para obter um calendário é necessário usar o método estático **getInstance()**.

```
1         Calendar calendar = Calendar.getInstance();
2
3         System.out.println("Data em milesegundos "+calendar.
      getTimeInMillis());
4
5         System.out.println("Dia do m s "+calendar.get(Calendar.
      DAY_OF_MONTH));
6
7         System.out.println("Dia da semana "+calendar.get(Calendar.
      DAY_OF_WEEK));
8
9         System.out.println("Hora do dia "+(calendar.get(Calendar.
      HOUR_OF_DAY)));
10
11        System.out.println("Minuto "+calendar.get(Calendar.MINUTE));
12
13        System.out.println("Segundo "+calendar.get(Calendar.SECOND));
14
15        System.out.println("Ano "+calendar.get(Calendar.YEAR));
16
17        simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy");
18
19        System.out.println("Formata o "+simpleDateFormat.format(
      calendar.getTime()));
20
21        simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
22
23        System.out.println("Formata o "+simpleDateFormat.format(
      calendar.getTime()));
```

```
1         Calendar calendar1 = Calendar.getInstance();
2
3         calendar1.setTime(new SimpleDateFormat("dd-MM-yyyy").parse("
      19-11-2021"));
4
5         calendar1.add(Calendar.DAY_OF_MONTH, +5);
6
7         System.out.println(new SimpleDateFormat("dd-MM-yyyy").format(
      calendar1.getTime()));
8
9         calendar1.add(Calendar.MONTH, 2);
10
11        System.out.println(new SimpleDateFormat("dd-MM-yyyy").format(
      calendar1.getTime()));
12
13        Calendar calendar2 = Calendar.getInstance();
14        Date dataInicial = new SimpleDateFormat("yyyy-MM-dd").parse("
      2021-11-19");
15
16        calendar2.setTime(dataInicial);
17
18        for(int i=1;i<=12;i++) {
19            calendar2.add(Calendar.MONTH, 1);
20            System.out.println("Parcela n mero: "+i+" vencimento    em:
      "
21                    +new SimpleDateFormat("dd/MM/yyyy").format(calendar2.
      getTime()));
22
23        }
```

LocalDateTime

LocalDate, LocalTime, LocalDateTime

- A **java.time** API foi uma nova API de datas introduzida no Java 8 visando resolver problemas que tínhamos com as velhas classes Date e Calendar.
- Sugestão de consulta

```
1         LocalDate dataHoje = LocalDate.now();
2         System.out.println("Data: "+dataHoje);
3
4         LocalTime horaAtual = LocalTime.now();
5         System.out.println("Hora: "+horaAtual);
6
7         LocalDateTime dataHora = LocalDateTime.now();
8         System.out.println("Data e Hora: "+dataHora);
9
10        //Formatando
11        System.out.println("Data e Hora formatados: "+dataHora.format(
      DateTimeFormatter.BASIC_ISO_DATE));
12        System.out.println("Data e Hora formatados: "+dataHora.format(
      DateTimeFormatter.ofPattern("dd/MM/yyyy")));
13
14        LocalDate localDate = LocalDate.now();
15        System.out.println("Data atual formatada: "+localDate.format(
      DateTimeFormatter.ofPattern("dd/MM/yyyy")));
16
17        System.out.println("Dia da semana: "+localDate.getDayOfWeek());
18        System.out.println("Dia do m s: "+localDate.getDayOfMonth());
19        System.out.println("Dia do ano: "+localDate.getDayOfYear());
20        System.out.println("M s: "+localDate.getMonth());
21        System.out.println("Ano: "+localDate.getYear());
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Class Instant

- Na linguagem Java, a Classe Instantânea é usada para representar o instante de tempo específico na linha do tempo atual. A Classe Instantânea estende a Classe de Objeto e implementa a interface Comparable.
-
- Sugestão de consulta

```
1         Instant inicio = Instant.now();
2
3         Thread.sleep(1000);
4
5         Instant ifinal = Instant.now();
6
7         Duration duration = Duration.between(inicio, ifinal);
8
9         System.out.println("Nano: "+duration.toNanos());
10        System.out.println("Milesegundo: "+duration.toMillis());
11        System.out.println("Minutos: "+duration.toMinutes());
12        System.out.println("Hora: "+duration.toHours());
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Class Period

- É comum precisar saber a diferença entre dois objetos Data.
- A classe **Period** possui vários métodos utilitários para esta finalidade, esta classe representa o tempo em anos, meses e dias.

```
1         //Per odos entre datas
2         LocalDate localDate2 = LocalDate.of(2019, 7, 1);
3         LocalDate localDate3 = LocalDate.of(2021, 11, 19);
4
5         System.out.println("Data antiga    maior que data nova: "+
      localDate2.isAfter(localDate3));
6         System.out.println("Data antiga    anterior a data nova: "+
      localDate2.isBefore(localDate3));
7         System.out.println("Datas iguais: "+localDate2.isEqual(localDate3
      ));
8
9         Period period = Period.between(localDate2, localDate3);
10        System.out.println("Per odo    : "+period.getYears()+" anos "+
      period.getMonths()+" meses e "
11                  +period.getDays()+" dias");
12        System.out.println(period.toTotalMonths());
13
```

```
1         //Add tempos
2         LocalDate dataBase = LocalDate.parse("2021-11-19");
3         System.out.println("Data base: "+dataBase);
4         System.out.println("Mais 5 dias: "+(dataBase = dataBase.plusDays
      (5)));
5         System.out.println("Mais 4 semana: "+(dataBase = dataBase.
      plusWeeks(4)));
6         System.out.println("Mais 7 meses: "+(dataBase = dataBase.
      plusMonths(7)));
7         System.out.println("Mais 2 anos : "+(dataBase = dataBase.
      plusYears(2)));
8         System.out.println("Menos 1 ano: "+(dataBase = dataBase.
      minusYears(1)));
```

OBRIGADO!