DIGITAL INNOVATION ONE

Desenvolvimento básico em Java

Trabalhando com datas

Marco Paulo Ollivier
Software Engineer

DIGITAL INNOVATION ONE

Objetivos da Aula

1. Aprender a manipular datas

2. Aprender a formatar datas

3. Entender a evolução do tratamento de datas no Java

Requisitos Básicos

- ✓ Lógica de programação
- ✓ Sintaxe da linguagem

Informação

Encontre todos os códigos dessa aula no Github

https://github.com/marcopollivier/DigitalInnovationOne-AulaJava

DIGITAL
INNOVATION
ONE

Parte 1: o java.util.Date

Trabalhando com datas

O java.util.Date

Antes de qualquer coisa, vamos definir aqui o ponto que estamos.

A implementação do java.util.Date está na JDK desde sua versão 1.0

Ou seja... É de se esperar que algumas coisas não se mostrem tão interessantes nos dias atuais, dado a sua idade.

O java.util.Date

Nesse primeiro momento, vamos ver como podemos trabalhar com a manipulação de datas a Classe **java.util.Date** do Java.

DIGITAL INNOVATION ONE

O java.util.Date

E o nosso primeiro passo é dar uma olhada na documentação oficial.

Vamos usar como referência o Java 8

https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

DIGITAL INNOVATION ONE

O java.util.Date

E para começar a falar sobre o Java Date, vamos falar sobre seus construtores. São eles:

```
Date()

Date(int year, int month, int date)

Date(int year, int month, int date, int hrs, int min)

Date(int year, int month, int date, int hrs, int min, int sec)

Date(long date)


Date(String s)
```

DIGITAL INNOVATION ONE

O java.util.Date

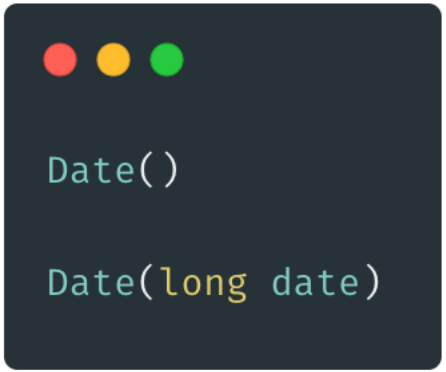
Entretanto alguns estão marcados como **deprecated**

```
//Todos estão marcados como @Deprecated desde a versão 1.1 da JDK

Date(int year, int month, int date)

Date(int year, int month, int date, int hrs, int min)

Date(int year, int month, int date, int hrs, int min, int sec)

Date(String s)
```

DIGITAL INNOVATION ONE

O java.util.Date

Portanto só vamos estudar os seguintes construtores



DIGITAL INNOVATION ONE

O java.util.Date

Date()

Este construtor vai alocar um objeto da classe Date e o **inicializará com o milissegundo mais próximo** do período da sua execução.

DIGITAL INNOVATION ONE

O java.util.Date

Date()

```
import java.util.Date;

public class Exemplo001 {
    public static void main(String[] args) {

        Date novaData = new Date();
        System.out.println(novaData);

        //retorna: Thu Jul 08 09:55:08 BRT 2019

    }
}
```

DIGITAL INNOVATION ONE

O java.util.Date

Date(long date)

Diferente do construtor anterior, esse construtor espera que você passe os milissegundos com base padrão de tempo (epoch) que usa como referência **1 de janeiro de 1970 00:00:00.**

DIGITAL INNOVATION ONE

O java.util.Date

Uma pequena pausa... O que é o Epoch?

"O epoch timestamp é um padrão largamente aceito para representar uma data como um inteiro 32-bits a partir do início do **Unix Epoch...**"

Tradução feita por Marco Paulo Ollivier do Livro "*Software Development: Na Open Source Approach*" escrito por Allen Tucker, Ralph Morelli e Chamindra de Silva. Página 163. Editora CRC Press

DIGITAL INNOVATION ONE

O java.util.Date

Date(long date)

Vamos testar com base no **System.currentTimeMillis()**

Esse método estático vai nos retornar o milissegundo mais próximo de sua execução com base no Sistema Operacional.

DIGITAL INNOVATION ONE

O java.util.Date

Date(long date)

```java
import java.util.Date;

public class Exemplo002 {
    public static void main(String[] args) {

        Long currentTimeMillis = System.currentTimeMillis();

        System.out.println(currentTimeMillis);
        // 1563127311564

        Date novaData = new Date(currentTimeMillis);

        System.out.println(novaData);
        // Sun Jul 14 15:01:51 BRT 2019

    }
}
```

O java.util.Date

Métodos úteis

Alguns métodos da classe Date são muito úteis e serão usados com frequência durante a manipulação de datas.

São eles...

DIGITAL INNOVATION ONE

O java.util.Date

Métodos úteis

Método	Retorno	Descrição
after(Date)	boolean	Checa se o objeto Data de referência é posterior ao comparado
before(Date)	boolean	Checa se o objeto Data de referência é anterior ao comparado
compareTo(Date)	int	Compara dois objetos Data
equals(Date)	boolean	Checa se os objetos são iguais
getTime()	long	Retorna a data em milissegundos
setTime(long)	void	Define uma data com base em milissegundos
from(Instante)	static Date	Define uma data com base em um Instant
toInstant()	Instant	Retorna um Instant com base em um Date

DIGITAL INNOVATION ONE

O java.util.Date

after e before

```java
import java.util.Date;

public class Exemplo003 {

    public static void main(String[] args) {

        Date dataNoPassado = new Date(15131124807691L);
        //Tue Dec 12 22:26:47 BRST 2017

        Date dataNoFuturo = new Date(16131124807691L);
        //Fri Feb 12 08:13:27 BRST 2021

        /** Comparando se a dataNoPassado é posterior a dataNoFuturo */
        boolean isAfter = dataNoPassado.after(dataNoFuturo);

        System.out.println(isAfter);
        //false

        /** Comparando se a dataNoPassado é anterior a dataNoFuturo */
        boolean isBefore = dataNoPassado.before(dataNoFuturo);

        System.out.println(isBefore);
        //true
    }
}
```

DIGITAL INNOVATION ONE

O java.util.Date

compareTo e equals

```java
import java.util.Date;

public class Exemplo004 {
    public static void main(String[] args) {

        Date dataNoPassado = new Date(15131124807691L); //Tue Dec 12 22:26:47 BRST 2017

        Date dataNoFuturo = new Date(16131124807691L); //Fri Feb 12 08:13:27 BRST 2021

        Date mesmaDataNoFuturo = new Date(16131124807691L); //Fri Feb 12 08:13:27 BRST 2021

        /** Comparando se as datas são iguais */
        boolean isEquals = dataNoFuturo.equals(mesmaDataNoFuturo);

        System.out.println(isEquals); //true

        /** Comparando uma data com a outra */
        int compareCase1 = dataNoPassado.compareTo(dataNoFuturo); //passado → futuro

        int compareCase2 = dataNoFuturo.compareTo(dataNoPassado); //futuro → passado

        int compareCase3 = dataNoFuturo.compareTo(mesmaDataNoFuturo); //datas equivalentes

        System.out.println(compareCase1); // -1

        System.out.println(compareCase2); // 1

        System.out.println(compareCase3); // 0

    }
}
```

O java.util.Date

from e toInstant

Antes de falar sobre esses dois métodos...

DIGITAL INNOVATION ONE

O java.util.Date

Classe Instant

- Surgiu na JDK 1.8;
- Imutável e Thread safe;
- Modela um ponto instantâneo de uma linha do tempo;
- Indicado para gravar marcações temporais em eventos da sua aplicação.

DIGITAL INNOVATION ONE

O java.util.Date

Classe Instant

```java
import java.time.Instant;
import java.util.Date;

/**
 *  Exemplo de conversão entre Date e Instant
 */
public class Exemplo005 {
    public static void main(String[] args) {

        Date dataInicio = new Date(1513124807691L);
        System.out.println(dataInicio);
        // Tue Dec 12 22:26:47 BRST 2017

        Instant instant = dataInicio.toInstant();
        System.out.println(instant);
        // 2017-12-13T00:26:47.691Z
    }
}
```

DIGITAL INNOVATION ONE

Exercício final

Aplique o que aprendemos:

- Descubra o **timeInMillis** do dia que você nasceu;
- Converta em um objeto Date;
- Verifique se ele é **anterior** ou **posterior** a **15 de maio de 2010.**

DIGITAL INNOVATION ONE

Leia mais sobre...

https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

https://docs.oracle.com/javase/8/docs/api/java/lang/System.html#currentTimeMillis--

https://docs.oracle.com/javase/8/docs/api/java/time/Instant.html

https://www.javatpoint.com/java-util-date

DIGITAL
INNOVATION
ONE

Parte 2: java.util.Calendar

Trabalhando com datas

O java.util.Calendar

Já na JDK 1.1 foi observada a necessidade de facilitar alguns recursos que a class Date oferecia.

Sendo assim, a classe **Calendar** foi criada.

Com isso uma série de métodos e construtores da classe Date foi depreciada. Por exemplo o construtor **Date(int year, int month, int date).**

O java.util.Calendar

Calendar é uma classe abstrata que provê métodos para converter data entre um instante específico.

O Calendar possui alguns campos específicos para manipulação como MONTH, YEAR, HOUR etc.

DIGITAL INNOVATION ONE

O java.util.Calendar

Capturando o instante atual com Calendar

```java
import java.util.Calendar;

public class Exemplo005 {
    public static void main(String[] args) {

        Calendar agora = Calendar.getInstance();

        System.out.println(agora);
    }
}
```


```
java.util.GregorianCalendar[
      time=1563147161361, areFieldsSet=true, areAllFieldsSet=true, lenient=true,
      zone=sun.util.calendar.ZoneInfo[
          id="America/Sao_Paulo",
          offset=-10800000, dstSavings=3600000, useDaylight=true, transitions=129,
          lastRule=java.util.SimpleTimeZone[
              id=America/Sao_Paulo,
              offset=-10800000,
              dstSavings=3600000,
              useDaylight=true,
              startYear=0,
              startMode=3,
              startMonth=10,
              startDay=1,
              startDayOfWeek=1,
              startTime=0,
              startTimeMode=0,
              endMode=3,
              endMonth=1,
              endDay=15, endDayOfWeek=1, endTime=0, endTimeMode=0
          ]
      ],
      firstDayOfWeek=1,
      minimalDaysInFirstWeek=1,
      ERA=1,
      YEAR=2019,
      MONTH=6,
      WEEK_OF_YEAR=29,
      WEEK_OF_MONTH=3,
      DAY_OF_MONTH=14,
      DAY_OF_YEAR=195,
      DAY_OF_WEEK=1,
      DAY_OF_WEEK_IN_MONTH=2,
      AM_PM=1,
      HOUR=8,
      HOUR_OF_DAY=20,
      MINUTE=32,
      SECOND=41,
      MILLISECOND=361,
      ZONE_OFFSET=-10800000,
      DST_OFFSET=0
   ]
```

DIGITAL INNOVATION ONE

O java.util.Calendar

Manipulando datas

```java
import java.util.Calendar;

public class Exemplo006 {
    public static void main(String[] args) {

        Calendar agora = Calendar.getInstance();

        System.out.println("A data corrente é : " + agora.getTime());
        // A data corrente é : Sun Jul 14 20:50:31 BRT 2019

        agora.add(Calendar.DATE, -15);
        System.out.println("15 dias atrás: " + agora.getTime());
        // 15 dias atrás: Sat Jun 29 20:50:31 BRT 2019

        agora.add(Calendar.MONTH, 4);
        System.out.println("4 meses depois: " + agora.getTime());
        // 4 meses depois: Tue Oct 29 20:50:31 BRT 2019

        agora.add(Calendar.YEAR, 2);
        System.out.println("2 anos depois: " + agora.getTime());
        // 2 anos depois: Fri Oct 29 20:50:31 BRT 2021

    }
}
```

O java.util.Calendar

Imprimindo datas e horas

Aqui vão algumas maneiras de se converter o resultado de um objeto **Calendar**

DIGITAL INNOVATION ONE

O java.util.Calendar

Imprimindo datas e horas

```java
import java.util.Calendar;

public class Exemplo007 {
    public static void main(String[] args) {

        Calendar agora = Calendar.getInstance();

        System.out.printf("%tc\n", agora);
        //Dom jul 14 20:58:11 BRT 2019

        System.out.printf("%tF\n", agora);
        //2019-07-14

        System.out.printf("%tD\n", agora);
        //07/14/19

        System.out.printf("%tr\n", agora);
        //08:58:11 PM

        System.out.printf("%tT\n", agora);
        //20:58:11
    }
}
```

DIGITAL INNOVATION ONE

Exercício final

Um cliente tem 10 dias para pagar uma fatura após sua data de vencimento sem que os juros sejam cobrados.

Caso essa data caia em um sábado ou domingo, o cliente pode pagar na segunda-feira seguinte.

Crie uma estrutura que receba uma data de vencimento e calcule quantos dias ele tem para pagar.

DIGITAL INNOVATION ONE

Leia mais sobre...

https://docs.oracle.com/javase/8/docs/api/java/util/Calendar.html

https://www.javatpoint.com/java-util-calendar

https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html

DIGITAL
INNOVATION
ONE

Parte 3: java.text.DateFormat

Trabalhando com datas

DIGITAL INNOVATION ONE

O java.text.DateFormat

Nesse ponto em que estamos existem, basicamente, duas classes para formatação de datas. O **DateFormat** e o **SimpleDateFormat.**

Ambos oferecem maneiras de formatar e parsear a saída das datas.

DIGITAL INNOVATION ONE

O java.text.DateFormat

DateFormat

```java
import java.text.DateFormat;
import java.util.Date;

/**
 * Exemplo de formatação de data com DateFormat
 */
public class Exemplo008 {
    public static void main(String[] args) {

        Date agora = new Date();

        String dateToStr = DateFormat.getInstance().format(agora);

        System.out.println(dateToStr);
        // 14/07/19 22:40

        dateToStr = DateFormat.getDateTimeInstance(DateFormat.LONG,DateFormat.SHORT).format(agora);

        System.out.println(dateToStr);
        // 14 de Julho de 2019 22:40

    }
}
```

DIGITAL INNOVATION ONE

O java.text.DateFormat

Já o **SimpleDateFormat** traz uma grande facilidade que é definir um padrão de formatação para a saída de data que você deseja.

O java.text.DateFormat

SimpleDateFormat

```java
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Exemplo de formatação de data com SimpleDateFormat
 */
public class Exemplo009 {
    public static void main(String[] args) {

        Date agora = new Date();

        SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

        String dataFormatada = formatter.format(agora);

        System.out.println(dataFormatada);
        // 14/07/2019

    }
}
```

DIGITAL INNOVATION ONE

Exercício final

Converta a Data atual no formato **DD/MM/YYYY HH:MM:SS:MMM**

DIGITAL INNOVATION ONE

Leia mais sobre...

https://docs.oracle.com/javase/8/docs/api/java/text/DateFormat.html

https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

DIGITAL
INNOVATION
ONE

Parte 4: Datas no Java 8+

Trabalhando com datas

DIGITAL INNOVATION ONE

Datas no Java 8+

O Java 8 veio com uma série de novidades para facilitar o trabalho com Datas.

E a grande melhoria está no pacote java.time que foi herdado do projeto Joda Time.

https://www.joda.org/joda-time/

DIGITAL INNOVATION ONE

Datas no Java 8+

Trabalhar com datas nunca foi tão fácil com esse novo pacote.

Nele destacam-se três classes:

- LocalDate;
- LocalTime;
- LocalDateTime.

Datas no Java 8+

Basicamente, o que tínhamos até então eram as classes que vimos até agora: Date e Calendar.

Com o uso constante, elas se mostram confusas e trabalhosas.

Além de serem mutáveis.

Datas no Java 8+

LocalDate é uma classe imutável para representar uma data.

Seu formato padrão é **yyyy-MM-dd**

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalDate

```java
import java.time.LocalDate;

/**
 * Exemplo de como utilizar LocalDate
 */
public class Exemplo010 {
    public static void main(String[] args) {

        LocalDate hoje = LocalDate.now();

        System.out.println(hoje);
        // 2019-07-14

    }
}
```

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalDate

```java
import java.time.LocalDate;

/**
 * Exemplo de como manipular LocalDate
 */
public class Exemplo011 {
    public static void main(String[] args) {

        LocalDate hoje = LocalDate.now();

        LocalDate ontem = hoje.minusDays(1);

        System.out.println(hoje);
        // 2019-07-14

        System.out.println(ontem);
        // 2019-07-13

    }
}
```

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalTime é uma classe imutável que representa um padrão de hora-minuto-segundo.

LocalTime pode ser representado até o nível de nanosegundos. Exemplo: **12:22:10:123212345**

Sua utilização é similar ao **LocalDate**

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalTime

```java
import java.time.LocalTime;

/**
 * Exemplo de como utilizar LocalTime
 */
public class Exemplo012 {
    public static void main(String[] args) {

        LocalTime agora = LocalTime.now();

        System.out.println(agora);
        // 23:53:58.421

    }
}
```

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalTime

```java
import java.time.LocalTime;

/**
 * Exemplo de como manioular LocalTime
 */
public class Exemplo013 {
    public static void main(String[] args) {

        LocalTime agora = LocalTime.now();

        System.out.println(agora);
        // 23:53:58.421

        LocalTime maisUmaHora = agora.plusHours(1);

        System.out.println(maisUmaHora);
        // 00:55:37.421

    }
}
```

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalDateTime funciona como uma espécie de junção entre o LocalTime e o LocalDate.

Também é uma classe imutável e você consegue trabalhar com dia e hora de uma só vez.

Você pode manipular a data e hora com precisão de nanosegundos. Exemplo: **2nd October 2007 at 13:45.30.123456789**

DIGITAL INNOVATION ONE

Datas no Java 8+

LocalDateTime

```java
import java.time.LocalDateTime;

/**
 * Exemplo de como manipular LocalDateTime
 */
public class Exemplo014 {
    public static void main(String[] args) {

        LocalDateTime agora = LocalDateTime.now();

        System.out.println(agora);
        // 2019-07-15T00:02:16.076

        LocalDateTime futuro = agora.plusHours(1).plusDays(2).plusSeconds(12);

        System.out.println(futuro);
        // 2019-07-17T01:02:28.076

    }
}
```

DIGITAL INNOVATION ONE

Exercício final

Adicione 4 anos, 6 meses e 13 horas horas ao momento 15/05/2010 10:00:00

DIGITAL INNOVATION ONE

Leia mais sobre...

https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html

https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html

https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html

DIGITAL
INNOVATION
ONE

Dúvidas?

Trabalhando com datas

DIGITAL INNOVATION ONE

Bibliografia

[1] DEITEL, Paul; DEITEL, Harvey - **Java: Como Programar**

[2] SILVEIRA, Paulo; TURINI, Rodrigo – **Java 8 Prático: Lamdas, Streams e os novos recursos da Linguagem**

[3] https://docs.oracle.com/javase/8/docs/