



Datas em Java

Manipulando Data em Java.

Data: Java x SQL

Objetivos



- Compreender o modo com Java trabalha com Datas e suas classes:
 - Date, Calendar e SimpleDateFormat
 - Converter Strings para tipo data e vice versa.
 - Cálculos com data.

Data e Horas: introdução

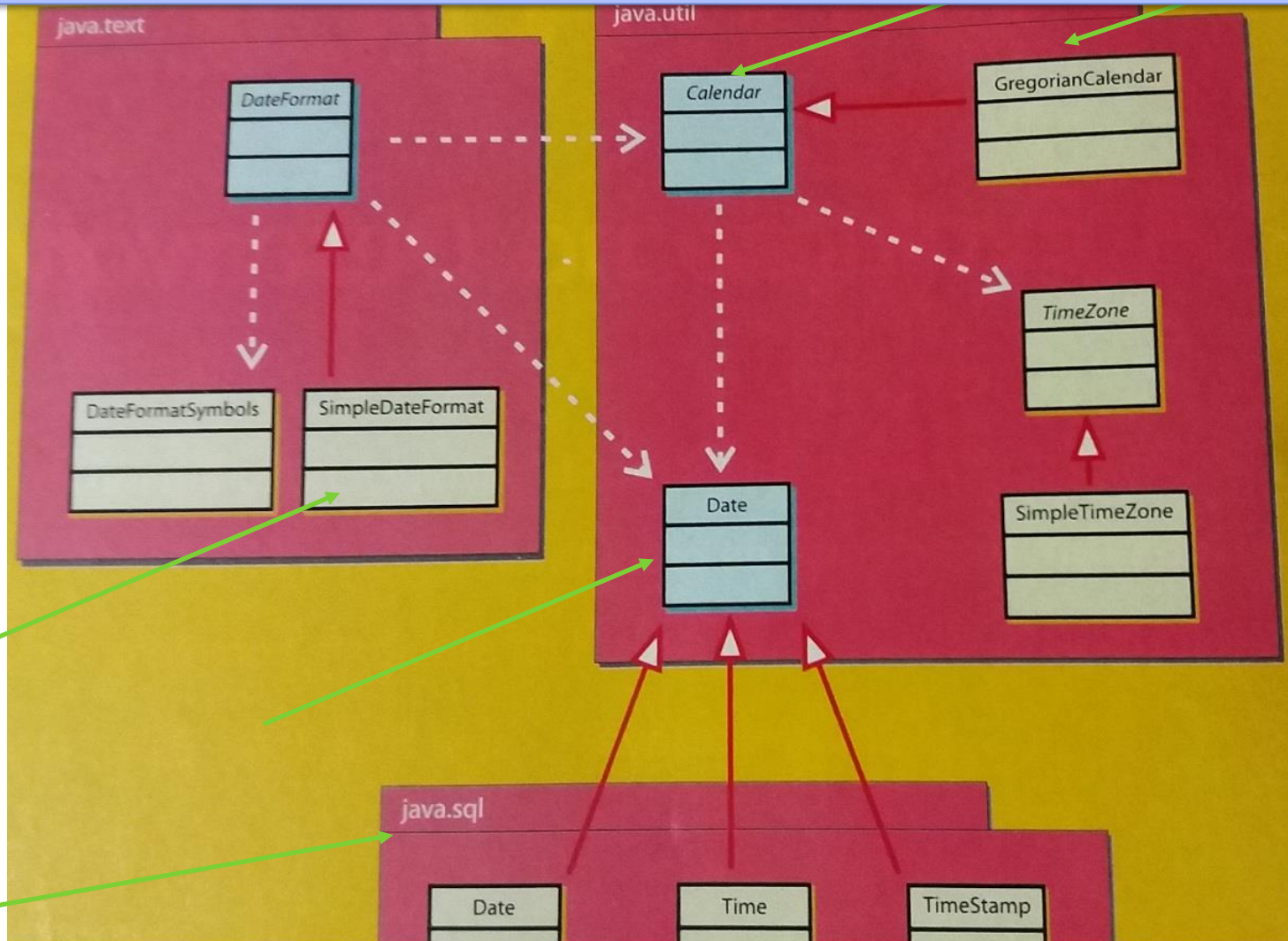


- Trabalhar com **datas (Calendário)** é trabalhoso em qualquer linguagem de programação:
 - Cada linguagem defini seu formato e modo de trabalhar com o tipo Date e as operações sobre ele: somar, subtrair, comparar.
 - A globalização trouxeram necessidades novas (internacionalização das aplicações): calendário arábico, chines..
 - Formatar a saída destes dados exige que formatação da data depende do local do usuário – ou seja, na internet estamos falando do mundo inteiro. Exemplo simples: formato Americano: Mês/Dia/Ano.
 - Linguagem criadas para usar bancos de dados, como Java, exigem do desenvolvedor um esforço **adicional**. Conversa entre duas linguagens, Java e SQL:
 - Alguém tem que fazer tradução de uma linguem para a outra, senão não tem conversa.
 - Quando armazena (insert) ou alterar (Updta): Java converte seu tipo Date para SQL Date
 - Quando recupera (select) : Java converte SQL Date para seu tipo Java Date
- A manipulação de datas em Java “pode” parecer um assunto complicado dada a quantidade de classes envolvidas (11).

Evolução de DATAS em Java

- **1996 - java.util.Date (JDK 1.0)**
 - Objeto que armazena uma data em Java
- **1998 - java.util.Calendar (JDK 1.1)**
 - Objeto que auxiliar a operações com tipo Date.
- **2005 - Joda-Time (third library)**
 - Empresa que criou API para trabalhar com Datas em Java e que foi sendo adota pelos programadores por ser atualizada (aspectos de internacionalização).
- **Java Time API : JSR-310 (versão 8.0) 25 de março de 2014***
 - As novas aplicações que exige internacionalização estão utilizando.

- As soluções simples utilizadas por outras linguagens não são adequadas, e essa profusão de classes corresponde a uma divisão bem clara de responsabilidades, visando atender a várias culturas em ambiente interconectado.
- No final das contas, a grande maioria dos casos é atendida por apenas 3 classes: **Date**, **Calendar** e **DateFormat (SimpleDateFormat)**



Classe Date

- Date representa o tempo, composto por: ano, mês, dia atual, minuto atual e segundos e milissegundos.
- É um valor do tipo **long** de valores crescente, iniciado em 01-01-1970 (unix).
- No momento da criação do objeto Date, o sistema armazena a Data/Hora.. do relógio do computador , numa representação em milissegundos.
System.currentTimeMillis.

```
1. public class Exemplo_Date {  
  
2.     public static void main(String[] args) {  
  
3.         Date data = new Date(); // data do momento da criação do objeto  
  
4.         System.out.println("Data Agora: "+ data);  
  
5.         System.out.print("\nData Agora (Long): " + data.getTime());  
6.     }  
7. }
```

(linha 4) Data Agora: Tue Jan 26 10:23:25 BRT 2021

(linha 5) Data Agora (long): 1611667405971

Classe Date

- CURIOSIDADE!!!
- Por armazenar em long significa que em **2038** teremos um “estouro”.
- O Windows utiliza um long, mas como a data de referência é 01 de janeiro de 1980, seu overflow será 10 anos depois.

Classe Date e classes auxiliares

- Hoje a maioria dos métodos da classe Date estão classificados como **deprecated** (depreciados), ou seja, são métodos que não **devem ser** mais utilizados e que não serão atualizados.
- A classe Date foi substituída pela classe **Calendar**, para suportar corretamente internacionalização do sistema de datas e operações (**GregorianCalendar**).
 - Calendário gregoriano foi promulgado pelo Papa Gregório XIII em 1582 para facilitar o relacionamento entre as nações*.
- Precisamos de classe que converta a Data para String e String para Data: **SimpleDateFormat**
 - Tipo data em Java é Date:
 - Na interface gráfica a data é obtida como String (JTextField)

Convertendo Datas

1. Converter String para Date

- Interface gráfica para armazenar num objeto Date

2. Converter Date para String

- De um objeto Date para mostrar numa interface gráfica (JTextField).

- Classes: Date e SimpleDateFormat

Convertendo: String para Date

Date e SimpleDateFormat

1. Obtém a String que representa a data na interface gráfica do usuário (JTextField):

```
//pegando dados de um formulário  
String dataStr = dataCompraTextField.getText(); // "14/09/2004"
```

2. Cria um objeto SimpleDateFormat utilizando uma máscara com o formato que deseja armazenar o objeto Date (dia, mês, ano, hora, min, seg):

```
//cria o objeto SimpleDateFormat com a máscara para converter  
SimpleDateFormat sdf= new SimpleDateFormat("dd/MM/yyyy");
```

2. Utilize o método **parse** do objeto SimpleDateFormat para converter a String para Date:

```
//usar máscara para converter  
Date dataCompra = sdf.parse(dataStr);
```

Convertendo String2Date

```
public class ConvertendoString2Data {  
  
    public static void main(String[] args) {  
  
1.        JTextField dataCompraTextField = new JTextField();  
2.        dataCompraTextField.setText("06/09/2020");//valor digitado pelo usuário  
3.  
4.        //pegando dados de um formulário  
5.        String dataStr = dataCompraTextField.getText();  
6.        //usar máscara para converter  
7.        SimpleDateFormat sdf= new SimpleDateFormat("dd/MM/yyyy");  
  
8.        try {  
9.            Date data = sdf.parse(dataStr);  
10.           System.out.println("Data de compra: "+ data);  
11.        } catch (ParseException e) {  
            System.out.println("Entre com uma nova data: formato ou  
                                data inválida");  
  
1.        }  
    }  
}
```

Data de compra: **Thu Sep 06 00:00:00 BRT 2020**

Convertendo: Date para String

Date e SimpleDateFormat

1. Identifica o objeto Date que será convertido para String (normalmente vindo do BD por meio de um VO)

```
//objeto data para ser convertido, a data de hoje
```

```
Date dataNascimento = vo.getDataNascimento();
```

2. Cria um objeto SimpleDateFormat utilizando uma máscara com o formato que deseja converter o objeto Date:

```
//cria o objeto SimpleDateFormat com a máscara para converter  
SimpleDateFormat sdf= new SimpleDateFormat("dd/MM/yyyy");
```

3. Utilize o método **format** do objeto SimpleDateFormat para converter a Date para String

```
//converte Date para String com format
```

```
String dataHojeStr = sdf1.format(dataNascimento);
```

```
//Mostra na interface gráfica para o usuário data de nascimento.
```

Convertendo Data2String: SimpleDateFormat

Mostrar a data (date) em forma de texto?

```
public class ConvertendoDate2String {  
  
    public static void main(String[] args) {  
  
1.        Date dataHoje = new Date();  
  
2.        //você pode usar outras máscaras  
3.        SimpleDateFormat sdf1= new SimpleDateFormat("dd/MM/yyyy");  
  
4.        // convert aqui com format  
5.        String dataHojeStr = sdf1.format(dataHoje);  
  
6.        // valor mostrado para usuário no JTextField  
7.        // dataHojeTextField.setText(dataHojeStr);  
  
8.        System.out.println("Data de hoje: "+ dataHojeStr);  
    }  
}
```

Data de hoje: **26/01/2021**

Datas: o que foi visto

- Como definir datas com o tipo Date :
 - `Date` dataCompra;
 - `Date` dataNascimento;
- Como mostrar (converter para String) ou obter uma data fornecida pela usuário (converter para Date) com os métodos de SimpleDateFormat:
 - `format(dataHoje);` // converte String
 - `parse(dataHojeString);` //converte para Date
- Utilizando objeto DateFormat: para formatação específicas.

Classe Calendar: operações com data


- 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()**.
- Calendário ocidental: dia do mês, mês, ano, dia da semana, hora e minuto, além de variações como numero do dia e da semana em relação ano.

Classe Calendar

```
import java.util.Calendar;
```

```
public class Data_Calendar{
```

```
1.  public static void main(String[] args)
2.  {
3.      Calendar c = Calendar.getInstance(); // new Date()
4.      System.out.println("Data e Hora atual: "+c.getTime());
    }
}
```



Data e Hora atual: Mon Apr 27 11:21:17 BRT 2015


Classe Calendar:

Mostrando o dia da semana, mês e ano

```
import java.util.Calendar;
```

```
public class TestaCalendario{
```

```
    public static void main(String[] args) {  
1.        Calendar c = Calendar.getInstance();  
2.  
3.        System.out.println("Data/Hora atual: "+ c.getTime());  
4.        System.out.println("Ano: "+ c.get(Calendar.YEAR));  
5.        System.out.println("Mês: "+ c.get(Calendar.MONTH));  
6.        System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));  
    }  
}
```



```
Data/Hora atual: Mon Apr 27 11:29:10 BRT 2015  
Ano: 2015  
Mês: 3  
Dia do Mês: 27
```

Classe Calendar:

Definindo/Alterando a data com método set

```
import java.util.Calendar;
```

```
public class TestaAlterarCalendario{
```


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

```
1.         Calendar c = Calendar.getInstance();
2.         c.set(Calendar.YEAR, 2019);
3.         c.set(Calendar.MONTH, Calendar.MARCH);
4.         c.set(Calendar.DAY_OF_MONTH, 20);
5.
6.         System.out.println("Data/Hora atual: "+ c.getTime());
7.         System.out.println("Ano: "+ c.get(Calendar.YEAR));
8.         System.out.println("Mês: "+ c.get(Calendar.MONTH));
9.         System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
        }
    }
```

```
Data/Hora atual: Mon Mar 20 11:38:35 BRT 2019
Ano: 2019
Mês: 2
Dia do Mês: 20
```

Classe Calendar: Recuperando a hora do dia

```
public class BoasVindasCalendario{  
  
    public static void main(String[] args) {  
  
        Calendar c1 = Calendar.getInstance();  
        int hora = c1.get(Calendar.HOUR_OF_DAY);  
  
        if ( hora > 6 && hora < 12 ) {  
            System.out.println("Bom Dia");  
        }else if ( hora > 12 && hora < 18 ){  
            System.out.println("Boa Tarde");  
        }else{  
            System.out.println("Boa Noite");  
        }  
    }  
}
```



Bom Tarde

Classe Calendar: Verificando dia da semana (dia útil)

```
public class DiaUtilCalendario{
```

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

```
        Calendar c1 = Calendar.getInstance();
```

```
        c1.setTime(aluno.getDataAniversario()); // Calendar c/ a data de aniver
```

```
        int diaSemana = c1.get(Calendar.DAY_OF_WEEK);
```

```
        if ( diaSemana >= Calendar.MONDAY && diaSemana <= Calendar.FRIDAY)
```

```
            System.out.println("Dia util");
```

```
        else
```

```
            System.out.println("Final de semana");
```

```
    }
```

```
}
```



Classe Calendar:

Definindo/Alterando a data/hora com método set

```
import java.util.Calendar;
```

```
public class AlteraHoraCalendario{
```

```
    Calendar c = Calendar.getInstance(); //Date do momento  
    System.out.println("Data/Hora atual: "+ c.getTime());
```

```
    //Alterando data
```

```
    c.set(Calendar.YEAR, 2020);  
    c.set(Calendar.MONTH, Calendar.JANUARY);  
    c.set(Calendar.DAY_OF_MONTH, 25);
```

```
    //Alterando hora
```

```
    c.set(Calendar.HOUR, 10);  
    c.set(Calendar.MINUTE, 20);  
    c.set(Calendar.SECOND, 00);
```

```
    System.out.println("\nData/Hora atualizada: "+ c.getTime());
```

```
}
```

Data/Hora atual: Tue Jan 26 11:43:14 BRT 2021

Data/Hora atualizada: Sat Jan 25 10:20:00 BRT 2020

Aritmética de data: Classe Calendar

add(): modifica e roll(): não modifica 31/12/2019?

```
public class SomandoDatas {  
    public static void main(String[] args) {  
  
        Date hoje = new Date();  
  
        //representa calendario  
        Calendar cal = Calendar.getInstance();  
  
        cal.setTime(hoje);  
  
        // qualquer uma das opcoes serviria para alterar para amanha  
        // cal.add(Calendar.DAY_OF_WEEK, 1);  
        // cal.add(Calendar.DAY_OF_YEAR, 1);  
        cal.add(Calendar.DAY_OF_MONTH, 1);  
  
        Date amanha = cal.getTime();  
  
        System.out.println("Hoje eh: " + hoje );  
        System.out.println("Amanha eh: " + amanha );  
    }  
}
```

Hoje eh: Thu Sep **06** 03:54:57 BRT 2018
Amanha eh: Fri Sep **07** 03:54:57 BRT 2018

Aritmética de data: Classe Calendar

add(): modifica e roll(): não modifica

23:59:59?

```
import java.util.Calendar;  
import java.util.Date;
```

```
public class SomandoHoras {  
    public static void main(String[] args) {  
  
        Date dataEntrada = new Date();  
  
        //representa calendario  
        Calendar cal = Calendar.getInstance();  
  
        cal.setTime(dataEntrada);  
  
        cal.add(Calendar.HOUR_OF_DAY, 2);  
  
        Date dataSaida = cal.getTime();  
  
        System.out.println("Entrada eh: " + dataEntrada );  
        System.out.println("Saida eh: " + dataSaida );  
    }  
}
```

Entrada eh: Mon Sep 16 **01:37:42** BRT 2019
Saida eh: Mon Sep 16 **03:37:42** BRT 2019

Aritmética de datas: `GregorianCalendar`

- Semelhante a `Calendar`.
 - `add()`
 - `get()`
 - `set()`

Reviendo: Classe **DateFormat**

Classe pai do *SimpleDateFormat*

- Essa classe permite **converter** informações do tipo **String** para data do tipo **Date**, permitindo seguir um formato.
- Consegue-se trabalhar ao contrário, **convertendo** um dado do tipo **Date** para uma **String**.
- Por ser uma **classe abstrata**, não é possível instanciá-la, por isso deve ser usado para método estático **getDateInstance()**.
- Sempre quando declarado é preciso importar o pacote **java.text**.

Mostrando datas: DateFormat

- Para mostrar datas com nomes: 16 de Setembro de 2019.
- Uso de constante: LONG, MEDIUM ou **SHORT**).

```
Date hoje = new Date();
```

```
DateFormat dataCurta = DateFormat.getDateInstance();
```

```
DateFormat horaCurta = DateFormat.getTimeInstance();
```

```
DateFormat dataHoraCurta = DateFormat.getDateTimeInstance();
```

```
System.out.println("Hoje: " + dataCurta.format(hoje ); //10/09/2019
```

```
System.out.println("Hoje: " + horaCurta.format(hoje ); //10:23:11
```

```
System.out.println("Hoje: " + dataHoraCurta.format(hoje ); // 10/09/2019 10:23:11
```

```
DateFormat horaLonga = DateFormat.getTimeInstance(DateFormat.LONG);
```

```
System.out.println("Hoje: " + horaLona.format(hoje ); // 10h23min11s
```

Classe **DateFormat** formatando datas.

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

```
    Date data = c.getTime();
```

```
    DateFormat f = DateFormat.getDateInstance(DateFormat.FULL);
```

```
    System.out.println("Data brasileira: "+ f.format(data));
```

```
    f = DateFormat.getDateInstance(DateFormat.LONG);
```

```
    System.out.println("Data sem o dia descrito: "+ f.format(data ));
```

```
    f = DateFormat.getDateInstance(DateFormat.MEDIUM);
```

```
    System.out.println("Data resumida 1: "+ f.format(data ));
```

```
    f = DateFormat.getDateInstance(DateFormat.SHORT);
```

```
    System.out.println("Data resumida 2: "+ f.format(data ));
```

```
}
```

```
}
```

Data brasileira: Segunda-feira, 27 de Abril de 2015

Data sem o dia descrito: 27 de Abril de 2015

Data resumida 1: 27/04/2015

Data resumida 2: 27/04/15

Conversões: Date/String e String/Date

Comparando *SimpleDateFormat* e *DateFormat*

- 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 (simplicidade).
- Utilizamos `DateFormat` quando precisamos formatar com específicas informações.

DateFormat e SimpleDateFormat:

Máscaras

dd/mm/aaaa, aaaa-mm-dd,...

Letra	Componente de Data ou Tempo	Tipo	Exemplos
G	Era	Texto	AD
y	Ano	Ano	1996; 96
M	Mês do ano	Mês	July; Jul; 07
w	Semana do ano	Número	27
W	Semana do mês	Número	2
D	Dia do ano	Número	189
d	Dia do mês	Número	10
F	Dia da semana do mês	Número	2
E	Dia da semana	Texto	Tuesday; Tue
a	Marcador AM/PM	Texto	PM
H	Hora do dia(0-23)	Número	0
k	Hora do dia(1-24)	Número	24
K	Hora em am/pm(0-11)	Número	0
h	Hora em am/pm(1-12)	Número	12
m	Minutos de hora	Número	30
s	Segundos de minutos	Número	55
S	Millisegundo	Número	978
z	Time zone	General Time Zone	Pacific Standard Time;
Z	Time zone	RFC 822 time zone	-0800

SimpleDateFormat: Máscaras

String	Data formatada	Comentário
dd/MM/yyyy	25/12/2010	Padrão brasileiro
MM/dd/yyyy	12/25/2010	Padrão americano
yyyy-MM-dd	2010-12-25	Padrão de alguns bancos de dados
dd MMMMMMMM yyyy	25 Dezembro 2010	Quando tem mais de 2 caracteres 'M', o resultado é o nome do mês por extenso
HH:mm:ss:SSSS	15:22:54:1264	Hora...

Datas em Banco de Dados

- Tipos de datas em banco
- Comparação de datas

Tipos SQL para data

SQL-ANSI 2011 e SQL-ANSI 2016

<i>Data Type</i>	<i>Description</i>
DATE	Represents a date. Format : yyyy-mm-dd
TIME WITHOUT TIME ZONE	Represents a time of day without time zone. Format : hh:mm:ss
TIME WITH TIME ZONE	Represents a time of day with time zone. Format : yyyy-mm-dd AT TIME ZONE -06:00.
TIMESTAMP WITHOUT TIME ZONE	Represents a combination of DATE and TIME values separated by a space. Format : yyyy-mm-dd hh:mm:ss
TIMESTAMP WITH TIME ZONE	Represents a combination of DATE and TIME values separated by a space with time zone. Format : yyyy-mm-dd hh:mm:ss AT TIME ZONE -06:00.

Data em SQL: comparação de data

- Existem 3 tipos:
 1. Apenas data sem hora: Date
 2. Apenas hora sem data: Time
 3. Instante no tempo: **Timestamp** : = java.util.Date
- Formato padrão de armazenamento de data em banco: YYYY-MM-DD
 - Insert INTO tabela (... , '2020-01-26', ...);

Data em SQL: comparando

- Comparando datas: 2 estratégias

- Utilizar DateFormat e comparar as Strings: equals().

```
DateFormat df = DateFormat.getInstance(); //short
```

```
if ( df.format(dataVencimento).equals(df.format(hoje) )....
```

- Utilizar Calendar (normalizar com set as horas) e comparar com equals().

```
java.sql.Date dataVencimento = new java.sql.Date.valueOf("2019-16-01");
```

```
Calendar cal = Calendar.getInstance();
```

```
cal.set(Calendar.HOUR_OF_DAY,0);
```

```
cal.set(Calendar.MINUTE,0);
```

```
cal.set(Calendar.SECOND,0);
```

```
cal.set(Calendar.MILLISECOND,0);
```

```
if ( dataVencimento.equals(cal.getTime()));
```

Classe Calendar: problema !!!!

```
import java.util.Calendar;
```

```
public class TestaAlterarCalendario{
```

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

```
1.         Calendar c = Calendar.getInstance();
2.         c.set(Calendar.YEAR, 1995);
3.         c.set(Calendar.MONTH, Calendar.MARCH);
4.         c.set(Calendar.DAY_OF_MONTH, 20);
5.
6.         System.out.println("Data/Hora atual: "+ c.getTime());
7.         System.out.println("Ano: "+ c.get(Calendar.YEAR));
8.         System.out.println("Mês: "+ c.get(Calendar.MONTH));
9.         System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
    }
}
```

```
Data/Hora atual: Mon Mar 20 11:38:35 BRT 1995
Ano: 1995
Mês: 2   ???
Dia do Mês: 20
```

Evolução Data em Java

Por que uma nova API?

```
public boolean isMesNatal(Calendar calendar)
{
    return calendar.get(Calendar.MONTH) == 11; //11 representa Dezembro
}
```

Por que uma nova API? Dezembro == 11 ???

- JAVA TIME API (java.time)!!!!
 - Atualmente esta é nova API para trabalhar com datas em Java.
 - Caso tenham interesse ver link nas referências.
 - “Se der” apresentarei em alguma aula com esta nova API, mas vamos ainda trabalhar com data em Java de modo tradicional, pois ainda é a mais utilizada, mas é bom saberem que existe esta nova API, pois podem ser que em algum código na internet veja o seu uso.

Referência para leituras

- Leia mais em: Trabalhando com as classes Date, Calendar e SimpleDateFormat em Java
<http://www.devmedia.com.br/trabalhando-com-as-classes-date-calendar-e-simplydateformat-em-java/27401#ixzz3YW83VufO>
- Este link
<https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/Date.html> mostra todos os detalhes de métodos e construtores que a classe **Date** do pacote **java.util** possui.
- Para mais informações sobre a classe Calendar, acesse o link
<http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Calendar.html>.
Conheça a nova API de datas do Java 8
<https://www.devmedia.com.br/java-8-e-sua-nova-api-para-datas/31462>
<http://blog.caelum.com.br/conheca-a-nova-api-de-datas-do-java-8/>
- The Date-Time API
<https://docs.oracle.com/javase/tutorial/datetime/overview/index.html>