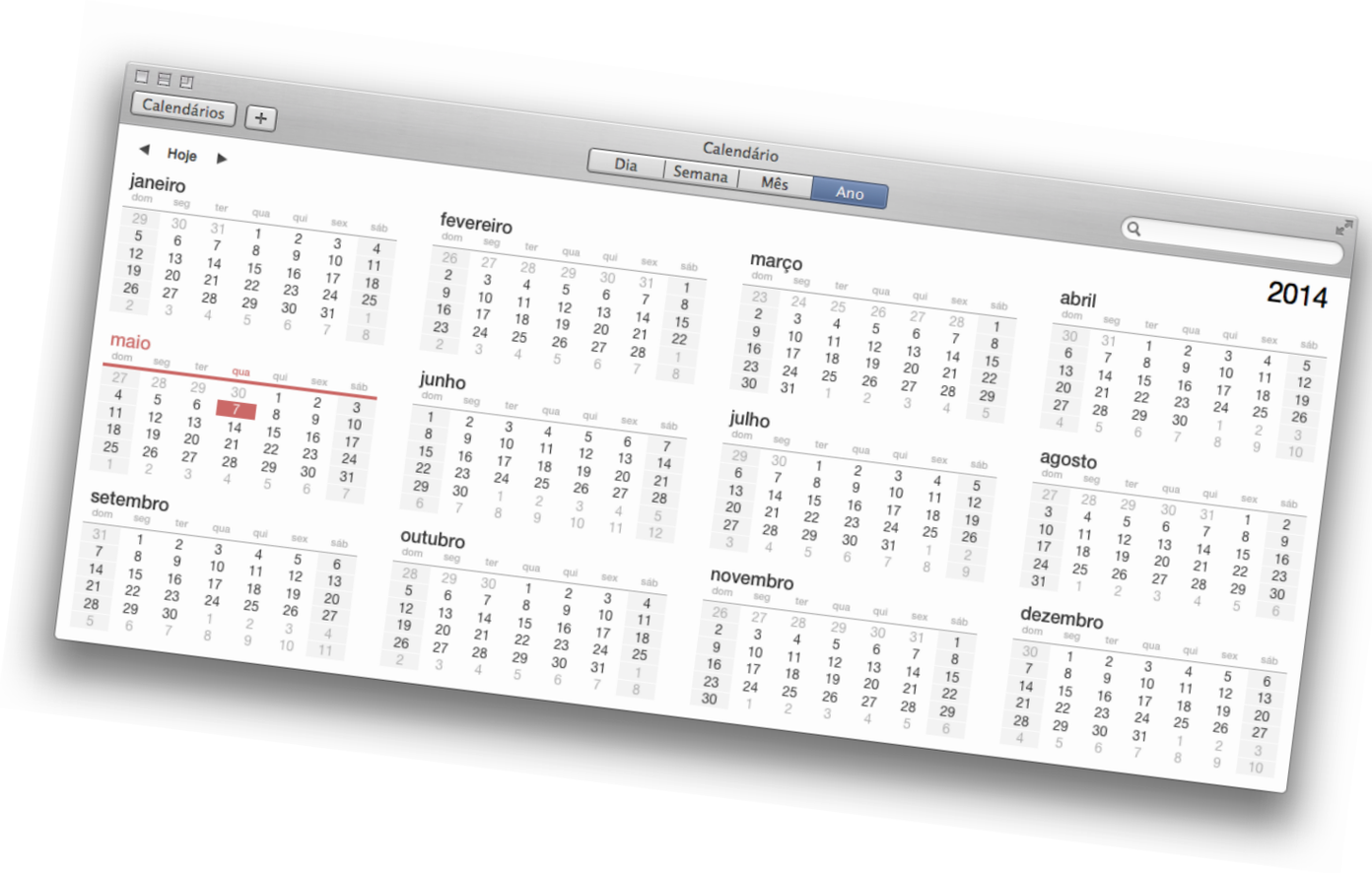
API

Agenda

- Datas e Horas
- Enumerações



Datas

As principais classes que manipulam data e hora até a versão 1.7 em Java são:

- java.util.Date
- java.util.Calendar
- java.util.GregorianCalendar

```
Date agora = new Date();

Calendar outraData =
    new GregorianCalendar(2014,
            Calendar.MAY, 5, 17, 12, 00);

outraData.add(Calendar.MONTH, -2);

long diferenca = 0;
if(outraData.before(agora)) {
    diferenca =
        outraData.getTimeInMillis() - agora.getTime();
} else {
    diferenca =
        agora.getTime() - outraData.getTimeInMillis();
}

long dias = diferenca / 1000 / 60 / 60 / 24;

System.out.format(
    "São %d dias de diferença entre as datas:\n", dias);
System.out.format(
    "Agora: %1$Td/%1$Tm/%1$TY%n", agora);
System.out.format(
    "e a outra data: %1$Td/%1$Tm/%1$TY%n", outraData);
```

Datas

Com a chegada do Java 8 temos uma grande quantidade de classes para manipular data, hora e suas subdivisões.

Abaixo estão listadas algumas classes do pacote java.time.

- LocalDate, LocalTime, LocalDateTime, ZoneId, ZoneInfo
- Instant, Period, ChronoUnit, ChronoField etc

```
Instant agora = Instant.now();

LocalDate outraData = LocalDate.of(2014, Month.MAY, 5);

outraData.minusMonths(2);

LocalDate agoraLocal = LocalDateTime.ofInstant(agora,
        ZoneId.systemDefault()).toLocalDate();
```

```
long dias = 0;
if(outraData.isBefore(agoraLocal)) {
    dias =
        ChronoUnit.DAYS.between(outraData, agoraLocal);
} else {
    dias =
        ChronoUnit.DAYS.between(agoraLocal, outraData);
}

System.out.format(
    "São %d dias de diferença entre as datas:\n", dias);
System.out.format(
    "Agora: %1$Td/%1$Tm/%1$TY%n", agoraLocal);
System.out.format(
    "e a outra data: %1$Td/%1$Tm/%1$TY%n", outraData);
```

Datas

Uma operação essencial é a conversão de texto para Data, isto é possível com as classes:

- DataFormat e SimpleDataFormat (Java 7)
- DateTimeFormatter (Java 8)

```
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
df.setLenient(false);

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
sdf.setLenient(false);

DateTimeFormatter dtFmt = new DateTimeFormatterBuilder()
        .appendValue(ChronoField.DAY_OF_MONTH, 2).appendLiteral('/')
        .appendValue(ChronoField.MONTH_OF_YEAR, 2).appendLiteral('/')
        .appendValue(ChronoField.YEAR, 4)
        .toFormatter()
        .withResolverStyle(ResolverStyle.STRICT);
```

Datas

Quando utilizamos DateFormat e SimpleDataFormat para a conversão de texto para data obtemos como resultado um objeto do tipo java.util.Date

```
try {
  String temp = JOptionPane.showInputDialog("Informe uma data (dd/mm/yyyy)");

  Date dataInformada = df.parse(temp);

  // ...
} catch(ParseException ex) {
    JOptionPane.showMessageDialog(null, "Data Inválida!");
}
```

Datas

Quando utilizamos DateTimeFormatter para a conversão de texto para data obtemos como resultado um objeto do tipo java.time.LocalDate ou LocalDateTime ou ZoneDateTime, dependendo da formatação escolhida.

```
try {
  String temp = JOptionPane.showInputDialog("Informe uma data (dd/mm/yyyy)");

  LocalDate dataInformada = LocalDate.parse(temp, dtFmt);

  //...
} catch(DateTimeParseException ex) {
    JOptionPane.showMessageDialog(null, "Data Inválida!");
    ex.printStackTrace();
}
```

Enumerações

A partir do Java 5 os tipos enumerados passou a ser suportado.

Estes tipos enumerados são conhecidos por enumerações ou **enum**.

Na sua forma mais simples as enumerações se parecem como uma lista de valores.

```
public enum Estacoes {
    PRIMAVERA,
    VERAO,
    OUTONO,
    INVERNO;
}
```

Enumerações

Algumas características das enumerações:

- Definem um tipo da mesma forma que a classe
- Limita os valores possíveis para este tipo
- São constantes
- Definem implicitamente os métodos toString(), valueOf() e values().

```
public enum Estacoes {
    PRIMAVERA,
    VERAO,
    OUTONO,
    INVERNO;
}
```

Enumerações

Também podemos adicionar métodos nas enumerações.

Se o método for abstrato, cada enumeração implementará a sua versão para o método.

Podemos definir construtores e métodos públicos e privados.

```
public enum Operacao {
  SOMA {
      public double execute(double x, double y) {
          return x + y;
      }
  },
  SUBTRAI {
      public double execute(double x, double y) {
          return x - y;
      }
  },
  MULTIPLICA {
      public double execute(double x, double y) {
          return x * y;
      }
  },
  DIVIDE {
      public double execute(double x, double y) {
          return x / y;
      }
  };

  public abstract double execute(double x, double y);
}
```

Referências

- Programando em Java2 - Teoria & Aplicações
  Rui Rossi dos Santos - Axcel Books - 2004

- Core Java2 - Volume I - Fundamentos
  Cay S. Horstmann & Gary Cornell - The Sun Microsystems Press - Série Java - 2003

- Java Programming
  Nick Clements, Patrice Daux & Gary Williams - Oracle Corporation - 2000

- http://docs.oracle.com/javase/tutorial/index.html

- http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html