Uso das Classes Calendar, DateFormat e NumberFormat, e consequente uso das classes Date e Locale na linguagem Java.

Frederico de Oliveira Gonçalves

Classe Calendar

Essa classe é popularmente utilizada para acessar os campos de um calendário. Por se tratar de uma classe abstrata, não pode ser instanciada, então é necessário utilizar o método getInstance(), e realizar o seguinte **import**: java.util.Calendar.

```
package br.com.javacurso;

import java.util.Calendar;

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
		System.out.println("Data atual: "+calendario.getTime());
	}
}
```

Console:

```
Data atual: Fri Oct 05 10:27:07 BRT 2018
```

A partir disso é possível solicitar informações específicas, como ano, mês ou dia:

```
package br.com.javacurso;

import java.util.Calendar;

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();

		System.out.println("Data/Hora atual: "+
calendario.getTime());
		System.out.println("Ano: "+ calendario.get(Calendar.YEAR));
		System.out.println("Mês: "+ calendario.get(Calendar.MONTH));
```

```
            System.out.println("Dia do Mês: "+
calendario.get(Calendar.DAY_OF_MONTH));
      }
}
```

Console:

```
Data/Hora atual: Fri Oct 05 10:48:25 BRT 2018
Ano: 2018
Mês: 9
Dia do Mês: 5
```

Ainda é possível determinar esses valores através do set():

```
import java.util.Calendar;

public class CalendarExamples {

      public static void main(String[] args) {
            Calendar calendario = Calendar.getInstance();
            calendario.set(Calendar.YEAR, 1988);
            calendario.set(Calendar.MONTH, Calendar.FEBRUARY);
            calendario.set(Calendar.DAY_OF_MONTH, 24);

             System.out.println("Data/Hora atual:
"+calendario.getTime());
             System.out.println("Ano: "+calendario.get(Calendar.YEAR));
             System.out.println("Mês: "+calendario.get(Calendar.MONTH));
             System.out.println("Dia do Mês:
"+calendario.get(Calendar.DAY_OF_MONTH));
      }
      }
```

Console:

```
Data/Hora atual: Wed Feb 24 10:43:30 BRT 1988
Ano: 1988
Mês: 1
Dia do Mês: 24
```

Os valores retornados do método get() são do tipo `int`, portanto é possível realizar operações utilizando o método add():

```
package br.com.javacurso;

import java.util.Calendar;

public class CalendarExamples {
```

```
	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
		calendario.set(Calendar.YEAR, 1990);
		calendario.add(Calendar.YEAR, +5);
		System.out.println("Ano: "+calendario.get(Calendar.YEAR));
		calendario.add(Calendar.YEAR, -10);
		System.out.println("Ano: "+calendario.get(Calendar.YEAR));

	}
	}
```

Console:

```
Ano: 1995
Ano: 1985
```

Para que os valores do campo que se deseja operar não interfiram nos demais campos, é necessário utilizar o roll(), ao invés do add():

```
package br.com.javacurso;

import java.util.Calendar;

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
		//Uso do add()
		calendario.set(Calendar.YEAR, 1990);
		calendario.set(Calendar.MONTH, 10);
		System.out.println("Ano: "+calendario.get(Calendar.YEAR));
		System.out.println("Mês: "+calendario.get(Calendar.MONTH));
		calendario.add(Calendar.MONTH, +5);
		System.out.println("Ano: "+calendario.get(Calendar.YEAR));
		System.out.println("Mês: "+calendario.get(Calendar.MONTH));
		//Uso do roll()
		calendario.roll(Calendar.MONTH, +11);
		System.out.println("Ano: "+calendario.get(Calendar.YEAR));
		System.out.println("Mês: "+calendario.get(Calendar.MONTH));
	}
	}
```

Console:

```
Ano: 1990
Mês: 10
Ano: 1991
Mês: 3
Ano: 1991
Mês: 2
```

Sabendo qual o tipo retornado pelo método get(), é possível fazer comparações booleanas, sendo assim podemos realizar o seguinte:

```
package br.com.javacurso;

import java.util.Calendar;

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
		 int hora = calendario.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");
		 }
	}
	}
```

Console:

```
Bom Dia
```

Maiores detalhes podem ser encontrados na documentação da classe: https://docs.oracle.com/javase/8/docs/api/java/util/Calendar.html

DateFormat

Para começarmos a entender a classe **DateFormat** é importante conhecer a Classe **Date** antes, que é responsável por representar um instante específico em tempo, com precisão de milissegundos.

A classe **DateFormat** é responsável por formatar datas baseado no formato padrão, mas também sendo possível fornecer um local ou estilo de formatação. Os estilos de formatação incluem FULL, LONG, MEDIUM e SHORT.

A Classe **DateFormat** é bastante conhecida pelo uso de sua SubClasse direta: **SimpleDateFormat**.

SimpleDateFormat é responsável por formatar informações do tipo Date para String (texto), assim como também o contrário através do método Parse().

Assim como a classe **Calendar**, a classe **DateFormat** é abstrata e necessita do método getDateInstance(). Para utilizar é necessário realizar o **import** do seu pacote: java.text.DateFormat e também do pacote java.util.Date:

```
package br.com.javacurso;

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

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
		calendario.set(1988, Calendar.FEBRUARY, 24);
		 Date data = calendario.getTime();
		 System.out.println("Data sem formatação: "+data);

		 DateFormat formataData = DateFormat.getDateInstance();
		 System.out.println("Data com formatação: "+
formataData.format(data));

		 DateFormat hora = DateFormat.getTimeInstance();
		 System.out.println("Hora formatada: "+hora.format(data));

		 DateFormat dtHora = DateFormat.getDateTimeInstance();
		 System.out.println("Data e hora com formatação:
"+dtHora.format(data));
	}
	}
```

Console:

```
Data sem formatação: Wed Feb 24 11:54:40 BRT 1988
Data com formatação: 24/02/1988
Hora formatada: 11:54:40
Data e hora com formatação: 24/02/1988 11:54:40
```

Exemplos do uso dos formatos FULL, LONG, MEDIUM e SHORT:

```
package br.com.javacurso;

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

public class CalendarExamples {

	public static void main(String[] args) {
		Calendar calendario = Calendar.getInstance();
```

```
		Date data = calendario.getTime();
		DateFormat formato =
DateFormat.getDateInstance(DateFormat.FULL);
		System.out.println("Formato Full: "+formato.format(data));

		formato = DateFormat.getDateInstance(DateFormat.LONG);
		System.out.println("Formato Long: "+formato.format(data));

		formato = DateFormat.getDateInstance(DateFormat.MEDIUM);
		System.out.println("Formato Medium:
"+formato.format(data));

		formato = DateFormat.getDateInstance(DateFormat.SHORT);
		System.out.println("Formato Short: "+formato.format(data));
	}
	}
```

Console:

```
Formato Full: Sexta-feira, 5 de Outubro de 2018
Formato Long: 5 de Outubro de 2018
Formato Medium: 05/10/2018
Formato Short: 05/10/18
```

Para utilizarmos o **SimpleDateFormat** é necessário fazer o **import** através de java.text.SimpleDateFormate:

```
package br.com.javacurso;

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

public class CalendarExamples {

	public static void main(String[] args){
		Calendar calendario = Calendar.getInstance();
		Date data = calendario.getTime();

		SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
		System.out.println("Data formatada: "+sdf.format(data));

		SimpleDateFormat sdf2 = new SimpleDateFormat("h:mm a");
		System.out.println("Data formatada: "+sdf2.format(data));

		SimpleDateFormat sdf3 = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSSXXX");
		System.out.println("Data formatada: "+sdf3.format(data));
	}
	}
```

Console:

```
Data formatada: 05/10/2018
Data formatada: 12:10 PM
Data formatada: 2018-10-05T12:10:44.126-03:00
```

Outra Classe importante para a formatação de datas, que utilizaremos também para a formatação de números é o Locale, classe responsável por representar regiões político-geográficas. Ela deve ser importada utilizando o **import** java.util.Locale:

```
package br.com.javacurso;

import java.text.DateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class CalendarExamples {

	public static void main(String[] args){
		Calendar calendario = Calendar.getInstance();
		Date data = calendario.getTime();

		DateFormat f3 = DateFormat.getDateInstance(DateFormat.FULL,
Locale.US);
		System.out.println("Data e hora norte-americana:
"+f3.format(data));

		DateFormat f4 = DateFormat.getDateInstance(DateFormat.FULL,
Locale.FRANCE);
		System.out.println("Data e hora Francesa:
"+f4.format(data));
	}
	}
```

Console:

```
Data e hora norte-americana: Friday, October 5, 2018
Data e hora Francesa: vendredi 5 octobre 2018
```

Maiores detalhes podem ser encontrados na documentação da classe: https://docs.oracle.com/javase/8/docs/api/java/text/DateFormat.html

NumberFormat

Essa também é uma classe abstrata e é responsável pela formatação de números. Na maioria de seus métodos ela irá receber um número e retornar um String (texto). Através do método parse, é possível inverter esse processo.

Para ser utilizada é necessário realizar o **import** de java.text.NumberFormat. Segue exemplo de uso de um de seus métodos mais comuns, o getInstance():

```
package br.com.javacurso;

import java.text.NumberFormat;

public class NumberFormatExamples {

	public static void main(String[] args) {

		double numero = 123456.90;

		System.out.println(numero);
		System.out.println(NumberFormat.getInstance().format(numero));

	}
	}
```

Console:

```
123456.9
123.456,9
```

Assim a formatação ocorrerá através do formato padrão. É possível determinar outros formatos, inserindo o Locale nos parâmetros do método getInstance(Locale inLocale):

```
package br.com.javacurso;

import java.text.NumberFormat;
import java.util.Locale;

public class NumberFormatExamples {

	public static void main(String[] args) {

		double numero = 123456.90;

		System.out.println(numero);
		System.out.println(NumberFormat.getInstance().format(numero));
		System.out.println(NumberFormat.getInstance(Locale.UK).format(numero));

	}
	}
```

Console:

```
123456.9
```

```
123.456,9
123,456.9
```

Outro método muito utilizado desta classe é o getCurrencyInstance(), que formata o número desejado para uma formatação monetária padrão. Também é possível fazer uso do parâmetro Locale para determinar o uso de formatação conforme a moeda da região especificada:

```
package br.com.javacurso;

import java.text.NumberFormat;
import java.util.Locale;

public class NumberFormatExamples {

	public static void main(String[] args) {

		double numero = 123456.90;

		System.out.println(numero);
		System.out.println(NumberFormat.getCurrencyInstance().format(numero));

	System.out.println(NumberFormat.getCurrencyInstance(Locale.UK).format(numero));

	}
	}
```

Console:

```
123456.9
R$ 123.456,90
£123,456.90
```

Mais um método bastante utilizado desta classe é o getPercentInstance(), que irá retornar um String (texto) no formato de porcentagem:

```
package br.com.javacurso;

import java.text.NumberFormat;

public class NumberFormatExamples {

	public static void main(String[] args) {

		double numero2 = 2.0/3.0;

		System.out.println(numero2);
		System.out.println(NumberFormat.getPercentInstance().format(numero2));

	}
	}
```

Console:

```
0.6666666666666666
67%
```

Outro método que pode ser utilizado para lidar com frações é getIntegerInstance(), que irá arredondar pontos flutuantes para o inteiro mais próximo usando o arredondamento half-even (`RoundingMode.HALF_EVEN`):

```
package br.com.javacurso;

import java.text.NumberFormat;

public class NumberFormatExamples {

	public static void main(String[] args) {

		double numero2 = 2.0/3.0;
		double numero3 = 7.0/3.0;

		System.out.println(numero2);
		System.out.println(numero3);
		System.out.println(NumberFormat.getIntegerInstance().format(numero2));
		System.out.println(NumberFormat.getIntegerInstance().format(numero3));

	}
	}
```

Console:

```
0.6666666666666666
2.3333333333333335
1
2
```

Maiores detalhes podem ser encontrados na documentação da classe: https://docs.oracle.com/javase/8/docs/api/java/text/NumberFormat.html

Fontes bibliográficas:

https://www.devmedia.com.br/trabalhando-com-as-classes-date-calendar-e-simpledateformat-em-java/27401

https://docs.oracle.com/javase/8/docs/api/index.html