01

Integrando moneta com a Stella

Transcrição

Já sabemos fazer as operações matemáticas básicas. Seria interessante fazer porcentagem. Vamos simular o texto de um email da Alura, dizendo que os cliente ganharam 10% de desconto caso ouçam um podcast. Já sabemos que o valor do desconto será de `90 reais` , pois o total é `900` . Mas é preciso que o programa saiba dizer isso para nós.

Para calcular porcentagem nessa API nova do Java, temos uma parte chamada `MonetaryOperators` . Usaremos como base o código desenvolvido anteriormente, tendo em vista que a porcentagem é calculada sobre o `valorTotal` . O método usado é o `with(MonetaryOperators)` , note que é o que está no plural. Dentro dele, usaremos o `percent(decimal)`

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    valorTotal.with(MonetaryOperators.percent(10));
  }

}
```

Criaremos então um `MonetaryAmount` para receber essa informação, chamado `desconto` . Em seguida, faremos um `sysout` para imprimir o resultado no console.

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);
  }

}
```

Ao rodar a classe, o console nos mostra o seguinte:

```
nov 07, 2016 3:32:33 PM org.javamoney.moneta.DefaultMonetaryContextFactory createMonetaryContext
INFORMAÇÕES: Using custom MathContext: precision=256, roundingMode=HALF_EVEN
BRL 75
BRL 900
BRL 9E+1
```

O resultado é o valor `9E+1`. Esse `E+1` é a precisão do `BigDecimal`. Embora seja importante, fica um pouco estranho usá-lo na comunicação com os nossos clientes. Para resolver isso, podemos fazer a integração com a biblioteca do Stella, que estamos usando no projeto, e converter esse código em escrita por extenso.

Usaremos a classe `NumericToWordsConverter`, que receberá o `FormatoDeReal`.

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);
    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
  }

}
```

Já sabemos que para escrever por extenso, devemos usar o `conversor.toWords`, passando para ele o que queremos escrever. No caso, é o `desconto`.

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);
    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    conversor.toWords(desconto);
  }

}
```

É preciso que uma string receba o que resultará do conversor, assim temos:

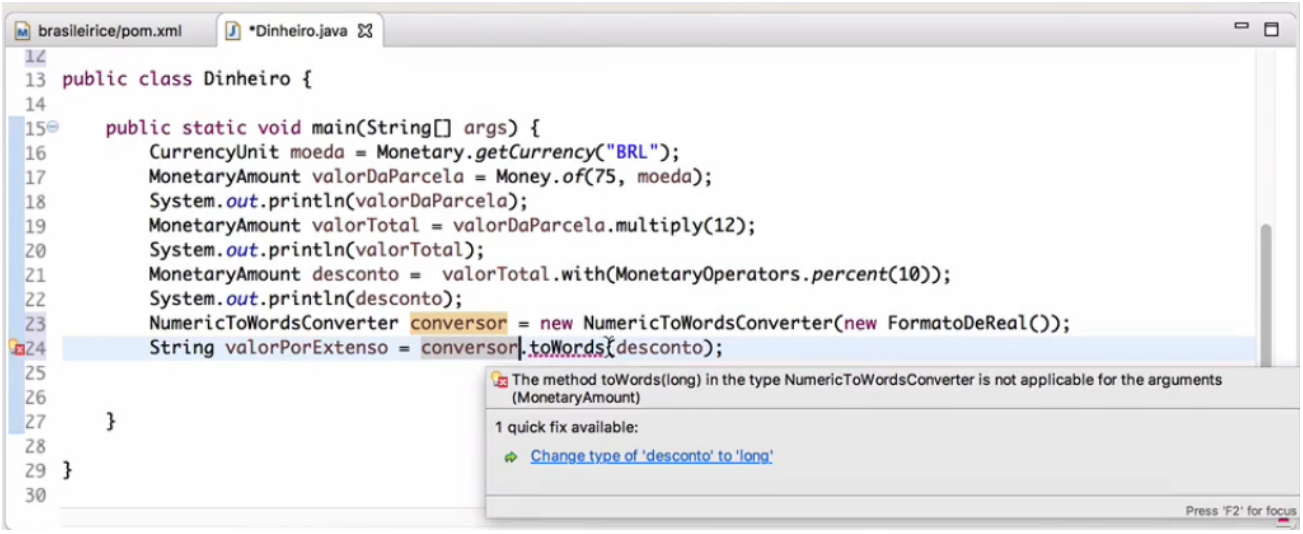
```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);
    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    String valorPorExtenso = conversor.toWords(desconto);
  }

}
```

O programa nos acusa erro.



O método `toWords` recebe em `double`, e estamos passando para ele um `MonetaryAmount`. Precisamos fazer a conversão deste para `double`. O `MonetaryAmount` define valor e moeda, e vem do `numberValue`, que precisamos transformar em um `double`. Faremos isso depois da linha em que definimos o `desconto`, usando o `getNumber` para que retorne um `NumberValue`.

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);

    desconto.getNumber();
```

```
    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    String valorPorExtenso = conversor.toWords(desconto);
  }

}
```

Como queremos um `NumberValue` , precisamos acrescentá-lo. Chamaremos de `descontoSemMoeda` .

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);

    NumberValue descontoSemMoeda = desconto.getNumber();

    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    String valorPorExtenso = conversor.toWords(desconto);
  }

}
```

Agora precisamos atualizar a string, que ainda está com o `desconto` . Em seguida, pediremos um `sysout` .

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);

    NumberValue descontoSemMoeda = desconto.getNumber();

    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    String valorPorExtenso = conversor.toWords(descontoSemMoeda.doubleValue());
    System.out.println(valorPorExtenso);
  }

}
```

Ao rodar, veremos no console:

```
nov 07, 2016 3:32:33 PM org.javamoney.moneta.DefaultMonetaryContextFactory createMonetaryContext
INFORMAÇÕES: Using custom MathContext: precision=256, roundingMode=HALF_EVEN
BRL 75
BRL 900
BRL 9E+1
noventa reais
```

Agora já podemos simular o email para o aluno, com outro `sysout` .

```
package br.com.alura;

public class Dinheiro {

  public static void main(String[] args){
    CurrencyUnit moeda = Monetary.getCurrency("BRL");
    MonetaryAmount valorDaParcela = Money.of(75, moeda);
    System.out.prinln(valorDaParcela);
    MonetaryAmount valorTotal = valorDaParcela.multiply(12);
    System.out.println(valorTotal);
    MonetaryAmount desconto = valorTotal.with(MonetaryOperators.percent(10));
    System.out.println(desconto);

    NumberValue descontoSemMoeda = desconto.getNumber();

    NumericToWordsConverter conversor = new NumericToWordsConverter(new FormatoDeReal());
    String valorPorExtenso = conversor.toWords(descontoSemMoeda.doubleValue());
    System.out.println(valorPorExtenso);
    System.out.println("Olá, Aluno. Ganhe " + valorPorExtenso + "ouvindo nosso podcast. LINK")
  }

}
```

Ao rodar, teremos no console:

```
nov 07, 2016 3:32:33 PM org.javamoney.moneta.DefaultMonetaryContextFactory createMonetaryContext
INFORMAÇÕES: Using custom MathContext: precision=256, roundingMode=HALF_EVEN
BRL 75
BRL 900
BRL 9E+1
noventa reais
Olá, Aluno. Ganhe noventa reais ouvindo nosso podcast. LINK
```

Assim, conseguimos integrar a API de dinheiro com a do Stella para fazer um texto útil de comunicação com o aluno.

Até a próxima!