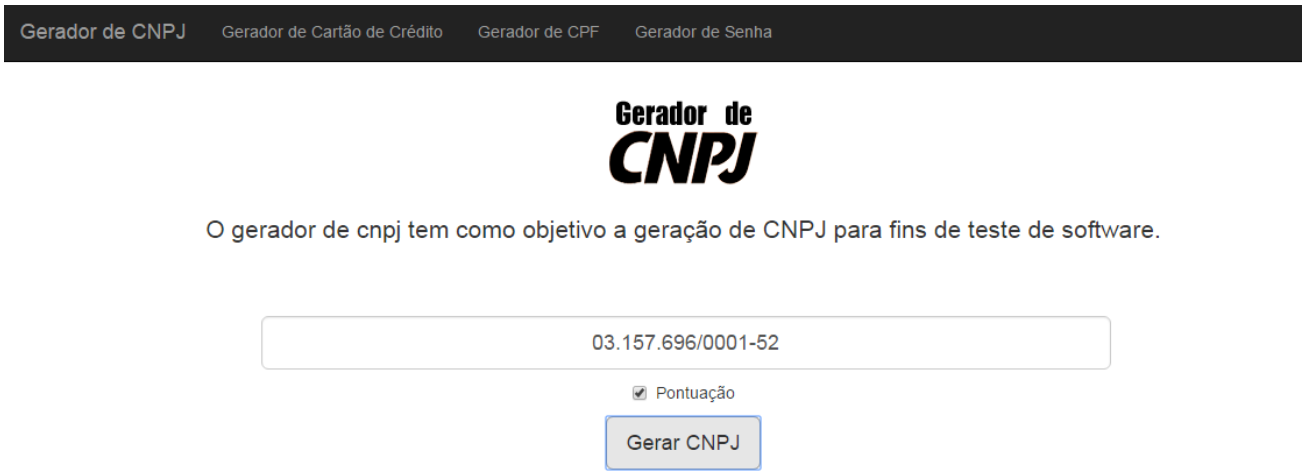
03

Validando CNPJ e Título de Eleitor

Transcrição

Já aprendemos a validar o CPF, precisamos aprender a validar outros documentos brasileiros, como o CNPJ. Para isso usaremos um gerador de CNPJ (http://www.4devs.com.br/gerador_de_cnpj). Esse site, assim como diversos outros, nos permite gerar CNPJs com ou sem pontuação.



Precisamos apenas dos números, então desmarcaremos a opção de pontuação. Para validar um número gerado no site, usaremos a biblioteca do Stella, assim como fizemos para o CPF. Começaremos criando uma `string`

```
package br.com.alura;

public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="22222222222";
    CPFValidator validador = new CPFValidator();
    try{
  validador.assertValid(cpf);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
    String cnpj = "82588641000173";

  }

}
```

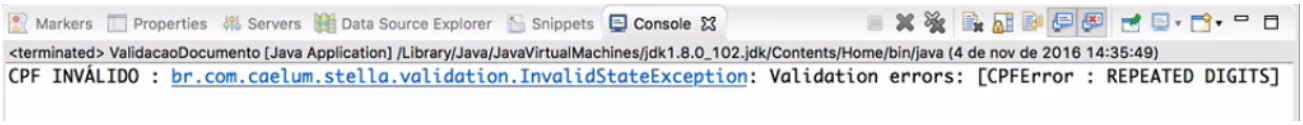
Já temos um validador para o CPF, então não podemos usar o mesmo para o CNPJ. Criaremos então uma classe, e um objeto dentro dela, ambos específicos para CNPJ.

```
    String cnpj = "82588641000173";
    CNPJValidator validadorCNPJ = new CNPJValidator();
```

Novamente, usaremos o `assertValid`. Mas em vez de passar o CPF, passaremos o CNPJ.

```
String cnpj = "82588641000173";
CNPJValidator validadorCNPJ = new CNPJValidator();
validadorCNPJ.assertValid(cnpj);
```

Vamos fazer o teste e rodar. Clicaremos com o botão direito do mouse `Run As > Java Application`. O console nos mostra o seguinte:



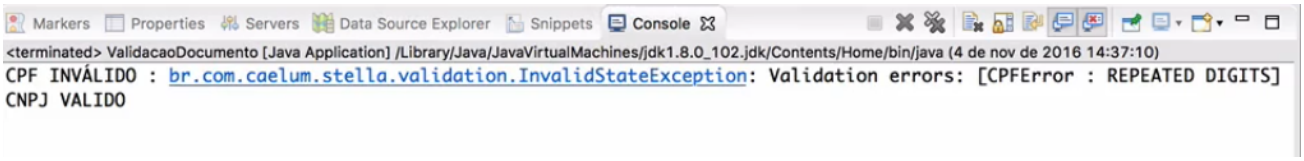
É uma mensagem referente ao CPF que deixamos li (`22222222222`), e não há nada refente ao CNPJ. Não colocamos um `try/catch` para ele, então o faremos agora.

```
String cnpj = "82588641000173";
CNPJValidator validadorCNPJ = new CNPJValidator();
try{

}catch (InvalidStateException e) {
}

validadorCNPJ.assertValid(cnpj);
```

E passaremos o `validadorCNPJ` para dentro do `try`, acrescentando as mensagens para o console.

```
String cnpj = "82588641000173";
CNPJValidator validadorCNPJ = new CNPJValidator();
try{
    validadorCNPJ.assertValid(cnpj);
    System.out.println("CNPJ VÁLIDO");

}catch (InvalidStateException e) {
    System.out.println("CNPJ INVÁLIDO :" + e);
}
```

Rodando novamente, temos no console:

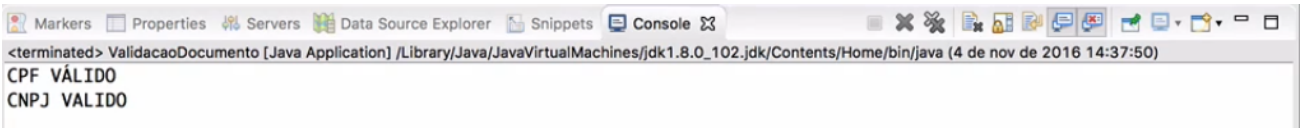


Substituiremos o CPF pelo primeiro número usado, que é `86288366757`.

```
package br.com.alura;

public class ValidacaoDocumento {
```

```
public static void main (String[] args) {
  String cpf="86288366757";
  CPFValidator validador = new CPFValidator();
  try{
validador.assertValid(cpf);
System.out.println("CPF VÁLIDO");

  }catch (InvalidStateException e) {
    System.out.println("CPF INVÁLIDO : " + e);
  }

  String cnpj = "82588641000173";
  CNPJValidator validadorCNPJ = new CNPJValidator();
  try{
      validadorCNPJ.assertValid(cnpj);
      System.out.println("CNPJ VÁLIDO");

  }catch (InvalidStateException e) {
      System.out.println("CNPJ INVÁLIDO :" + e);
  }
```

O console nos retorna a seguinte mensagem:



Agora, com a biblioteca, conseguimos validar tanto o CPF como o CNPJ. Precisamos também validar o título de eleitor, e a Stella será útil novamente. Com um gerador de título de eleitor (http://4devs.com.br/gerador_de_titulo_de_eleitor)

Criaremos uma nova `string` abaixo do bloco referente ao CNPJ:

```
String tituloEleitor="417453530116";
TituloEleitoralValidator validadorTitulo = new TituloEleitoralValidator();
validadorTitulo.assertValid(tituloEleitor);
```

Já sabemos que é preciso tratar com `try/catch` , então acrescentaremos isso agora.

```
String tituloEleitor="417453530116";
TituloEleitoralValidator validadorTitulo = new TituloEleitoralValidator();
try{
  validadorTitulo.assertValid(tituloEleitor);
    System.out.println("TÍTULO VÁLIDO");

}catch (InvalidStateException e) {
  System.out.println("TÍTULO INVÁLIDO :" + e);
}
```

Com o código completo, podemos rodar para testar as validações.

Os três documentos são válidos. Vamos trocar o último algarismo do título por zero, para ver se o programa detecta o erro.

```
String tituloEleitor="417453530110";
TituloEleitoralValidator validadorTitulo = new TituloEleitoralValidator();
try{
  validadorTitulo.assertValid(tituloEleitor);
    System.out.println("TÍTULO VÁLIDO");

}catch (InvalidStateException e) {
  System.out.println("TÍTULO INVÁLIDO :" + e);
}
```

O console dos exibe a mensagem:

```
CPF VÁLIDO
CNPJ VÁLIDO
TÍTULO INVÁLIDO : br.com.caelum.stella.validation.InvalidStateException: Validation errors: [Ti
```

Ou seja, o programa percebeu que um dos dígitos verificadores está errado. Voltaremos o título para o número original, e faremos uma troca no CNPJ, mudando o penúltimo número para zero.

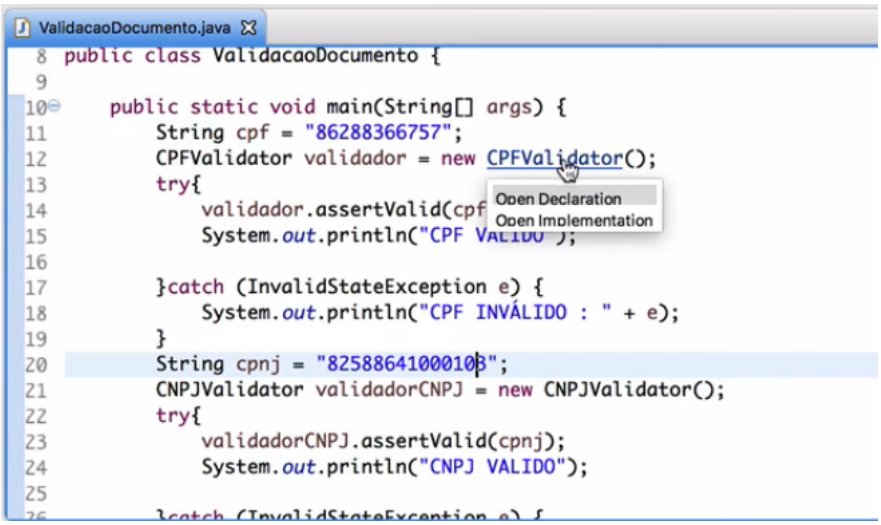
```
String cnpj = "82588641000103";
CNPJValidator validadorCNPJ = new CNPJValidator();
try{
    validadorCNPJ.assertValid(cnpj);
    System.out.println("CNPJ VÁLIDO");

}catch (InvalidStateException e) {
    System.out.println("CNPJ INVÁLIDO :" + e);
}
```

E o console nos mostrará o seguinte:

```
CPF VÁLIDO
CNPJ INVÁLIDO : br.com.caelum.stella.validation.InvalidStateException: Validation errors: [CNPJI
TÍTULO VÁLIDO
```

Agora temos certeza de que está tudo funcionando adequadamente. Observando o código novamente, notamos que o validador está se repetindo bastante. Podemos deixar o código mais elegante eliminando essa repetição. É importante reparar que são variações do validaor: `validadorCPF` , `validadorCNPJ` e `validadorTitulo` .

Quando passamos o mouse sobre o `CPFValidator` , ele nos mostra a opção de abrir a classe em outra aba.



Clicaremos sobre ela com o `Ctrl` pressionado para abri-la. Em seu código, vemos a seguinte linha:

```
public class CFPValidator implements Validador<String> {
  ...
}
```

Todos os validadores implementam a interface `Validator` . Quando verificamos no validador do CNPJ e do título, vemos linhas correspondentes. Portanto, podemos extrair essa validação e criar um método, que recebe um validador e um documento. Basta informar a instância (que é o validador específico que queremos usar), que ele verificará se o documento em questão é válido.

Inseriremos o novo trecho de código após a chave que encerra o método `main` . O método que criaremos pode ser `private` e tem que ser `static` porque estamos chamando no método main. Ele não precisa retornar nada porque estará capturando do `try` , portanto será `void` . Assim:

```
private static void validarDocumentos (Validator<String> validador, String documento)
```

Precisamos chamar o `validador` e o método, que é o `assertValid` , passando o documento que queremos chamar.

```
private static void validarDocumentos (Validator<String> validador, String documento){
  validador.assertValid(documento);
}
```

Com o método feito, podemos chamá-lo lá no main. Começaremos pelo trecho do CPF, que está assim:

```
public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="86288366757";
    CPFValidator validador = new CPFValidator();
    try{
  validador.assertValid(cpf);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
}
```

Para chamar o método que acabamos de criar, podemos apagar o `validador`.

```
public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="86288366757";
    CPFValidator validador = new CPFValidator();
    try{
        validarDocumentos(validador, documento);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
}
```

Nesse caso, o `validador` é o `new CPFValidator`, e o `documento` é o `cpf`. Com as devidas substituições:

```
public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="86288366757";
    CPFValidator validador = new CPFValidator();
    try{
        validarDocumentos(new CPFValidator(), cpf);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
}
```

Agora podemos excluir a linha que precede o `try`, que redunda com a que inserimos.

```
public class ValidacaoDocumento {
```

```
  public static void main (String[] args) {
    String cpf="86288366757";
    try{
        validarDocumentos(new CPFValidator(), cpf);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
}
```

Se rodarmos, o programa reconhece que é um CPF válido:

```
CPF VÁLIDO
CNPJ INVÁLIDO : br.com.caelum.stella.validation.InvalidStateException: Validation errors: [CNPJI
TÍTULO VÁLIDO
```

E se trocarmos o último dígito do CPF por zero?

```
public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="86288366757";
    try{
        validarDocumentos(new CPFValidator(), cpf);
  System.out.println("CPF VÁLIDO");

    }catch (InvalidStateException e) {
      System.out.println("CPF INVÁLIDO : " + e);
    }
}
```

Se colocarmos para rodar:

```
CPF INVÁLIDO : br.com.caelum.stella.validation.InvalidStateException: Validation errors: [CPFrr
CNPJ INVÁLIDO : br.com.caelum.stella.validation.InvalidStateException: Validation errors: [CNPJI
TÍTULO VÁLIDO
```

Ele percebeu que os dígitos validadores estão incorretos. Podemos fazer isso para os outros dois documentos. O código completo fica:

```
package br.com.alura;

public class ValidacaoDocumento {

  public static void main (String[] args) {
    String cpf="86288366757";
    try{
        validarDocumentos(new CPFValidator(), cpf);
  System.out.println("CPF VÁLIDO");
```

```
}catch (InvalidStateException e) {
  System.out.println("CPF INVÁLIDO : " + e);
}

String cnpj = "82588641000173";
try{
    validarDocumentos(new CNPJValidator(), cnpj);
    System.out.println("CNPJ VÁLIDO");

}catch (InvalidStateException e) {
    System.out.println("CNPJ INVÁLIDO :" + e);
}

String tituloEleitor="417453530116";
try{
  validarDocumentos (new TituloEleitoralValidator(), tituloEleitor);
    System.out.println("TÍTULO VÁLIDO");

}catch (InvalidStateException e) {
  System.out.println("TÍTULO INVÁLIDO :" + e);
}
```

Inserindo novos valores gerados pelos sites para cada documento e rodando, o console nos mostra:

```
CPF VÁLIDO
CNPJ VÁLIDO
TÍTULO VÁLIDO
```

O que fizemos é uma abordagem para melhorar o código, que pode ser usada para validar qualquer documento. Basta passar o validador e o documento em questão. Espero que tenha gostado. Até a próxima!