**FATEC CARAPCUIBA**

**Philip Viana Soares**

**1430482423014**

**TRABALHO DE LINGUAGEM DE PROGRAMAÇÃO**

**TRATAMENTO DE EXCEÇÕES**

**PHILIP VIANA SOARES**

**1430482423014**

**TRABALHO DE LINGUAGEM DE PROGRAMAÇÃO**

**TRATAMENTO DE EXCEÇÕES**

**Atividade exercida a matéria Linguagem de Programação da Faculdade Fatec Carapicuiba, como parte dos requisitos para obtenção nota ao segundo método de avalição da matéria (P2).**

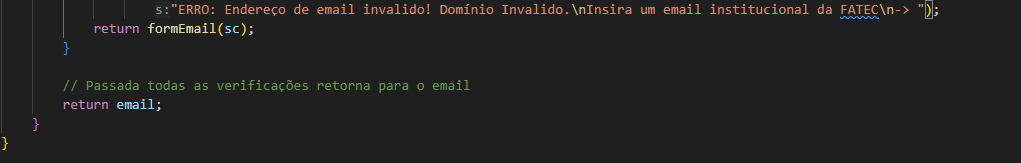
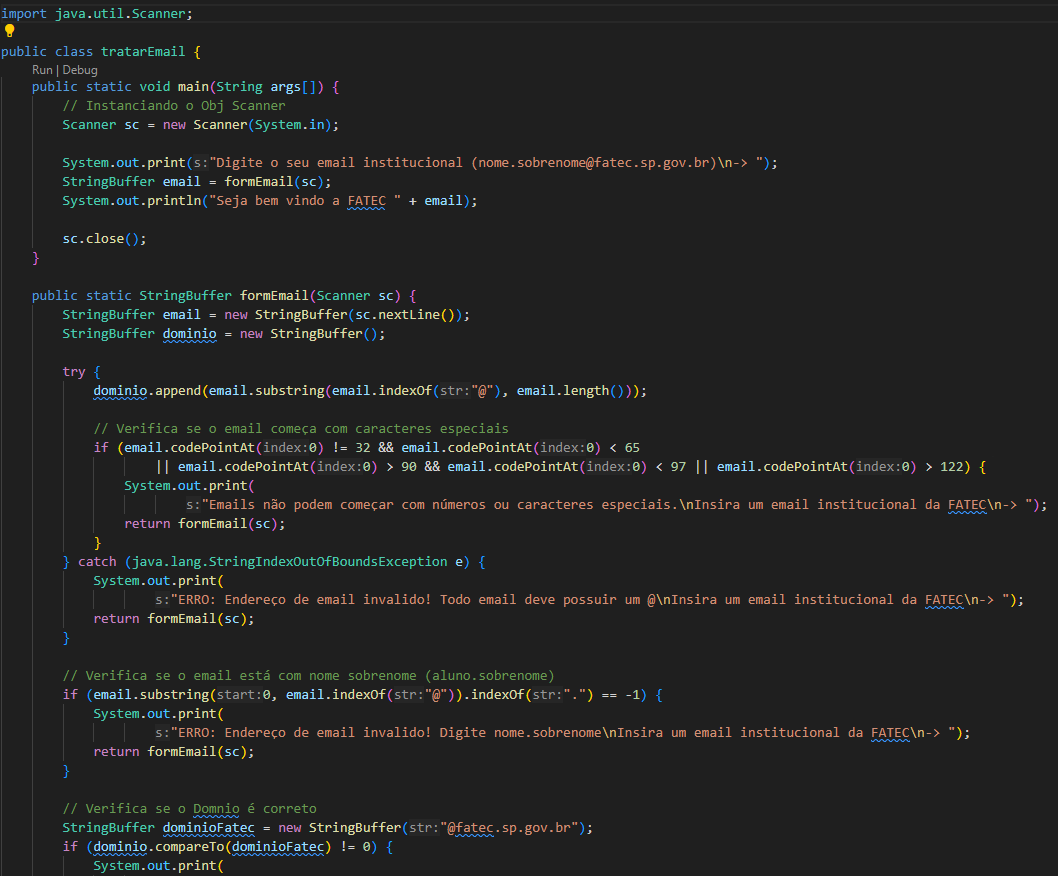
**Area de Concentração: Linguagem de Programação**

**Orientador: Prof. Sandra Bianca Henriques Geroldo.**

**Tratamento de exceções: Email FATEC**

Este problema foi proposto pela professora Sandra, na qual nos solicitava a validação de um EMAIL INSTITUCIONAL da FATEC.

Quando o usuário nos informar o seu email o método ***formEmail*** será executado:

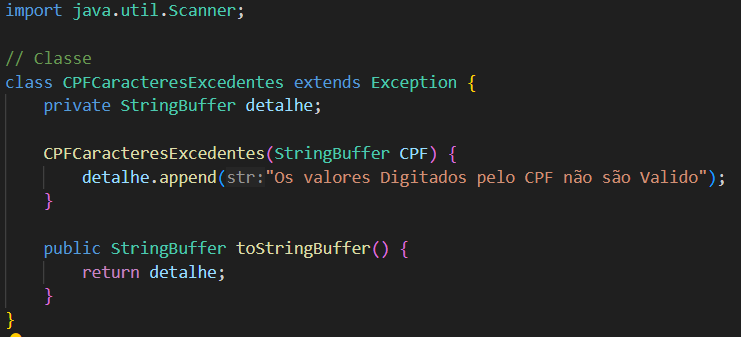


* Este código realiza a criação de 3 Objetos ***StringBuffer***:
  + ***Email***: realiza a captura do email informado pelo usuário;
  + ***Domino***: uma ***sub-string do objeto email***, na qual recorta todo o domínio do email (tudo após o @);
  + ***DominoFatec:*** domínio utilizado pela FATEC.
* Realiza a verificação, da possibilidade de atribuir a ***sub-string domínio*** na variável:
  + Caso ele não encontre o index atrelado ao caractere ‘@’, nos retornará uma exceção ***StringIndexOutOfBoundsException*** (não é iniciar a ***sub-string*** a partir index -1), solicitando novamente que o usuário digite o email;
  + Junto a isto, realiza a verificação do primeiro caractere que não pode iniciar com caracteres especiais. Caso sim, solicita novamente que o usuário digite o email;
* Após isto realiza a verificação, se o usuário digitou na formatação correta de um email da FATEC (nome.usuario), caso não, solicita novamente que o usuário digite o email.
* Por último compara o domínio digitado pelo usuário com a variável ***dominioFatec***, se estiver tudo certo o email é retornado para o método principal a qual emite uma mensagem de “bem-vindo!”.

**Tratamento de exceções: CPF e Estado emissor**

Este problema foi proposto pela professora Sandra, na qual nos solicitava a validação de um CPF e seu estado emissor. O Problema nos pede para fazer a utilização de uma exceção criada por nós mesmos.

Neste caso, realizei a criação da classe ***CPFCaracteresExcedentes***:

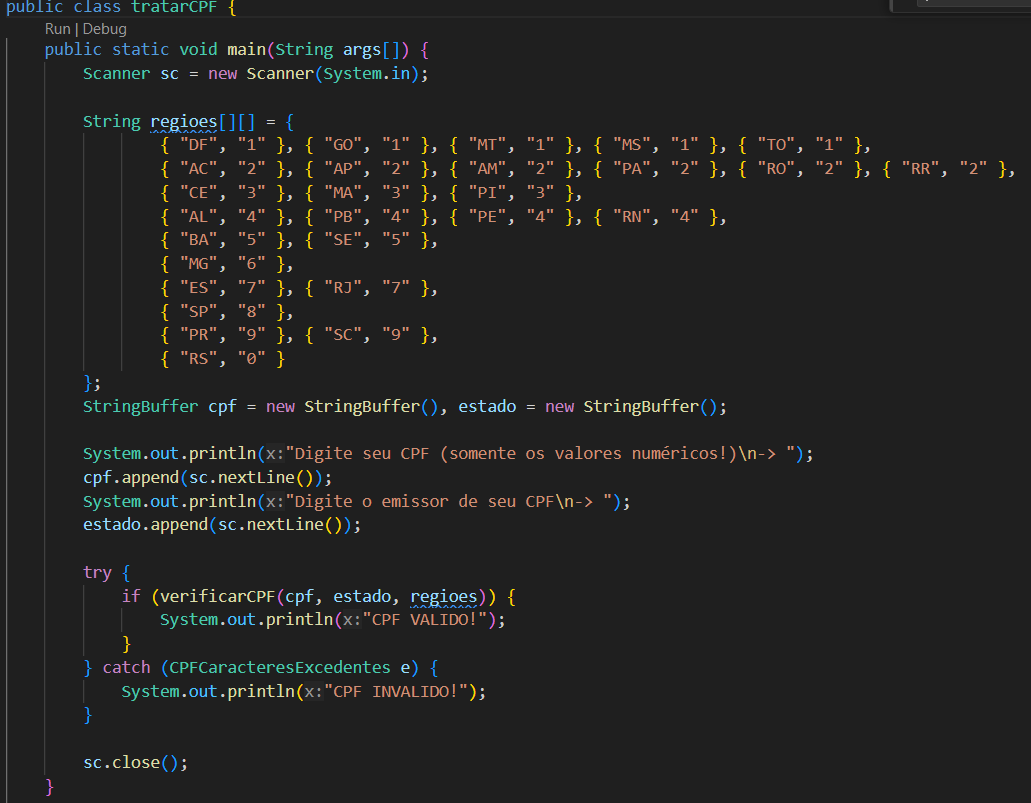


Esta classe só é chamada na verificação dos valores imputados pelo usuário. Pela regra de negócio o usuário deve digitar somente os números do CPF, excluindo todo e qualquer caractere especial que o acompanha. Ex.:

**Valor Invalido**: 123.456.789.10

**Valor valido**: 12345678910

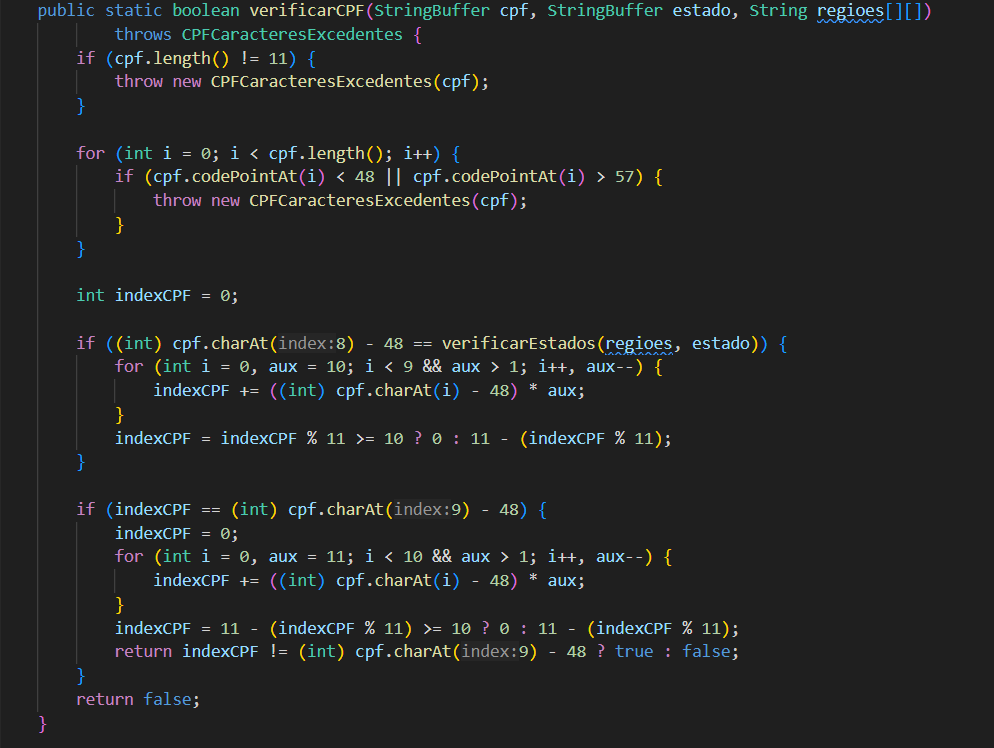
**MÉTODO PRINCIPAL**



Dentro do método principal da classe TratarCPF, temos a criação de 3 objetos:

* ***String regiões[][]***: Vetor de Constantes a qual acopla SIGLA do estado que o CPF foi emitido;
* ***CPF***: O CPF a qual o usuário quer validar. A não escolha de usar o tipo primitivo ***int***, é por causa do tamanho que o tipo pode suportar 2147483647 (10 caracteres), já que o CPF possui sempre 11 caracteres.
* ***ESTADO***: O estado a qual o CPF foi emitido.

E o nosso principal tratamento de exceções, a qual verifica se é possível validar o CPF através do método ***verificarCPF***. Caso ela nos gere uma exceção ***CPFCaracteresExcedentes***, é informado que o CPF é invalido.



O método ***verificarCPF:***

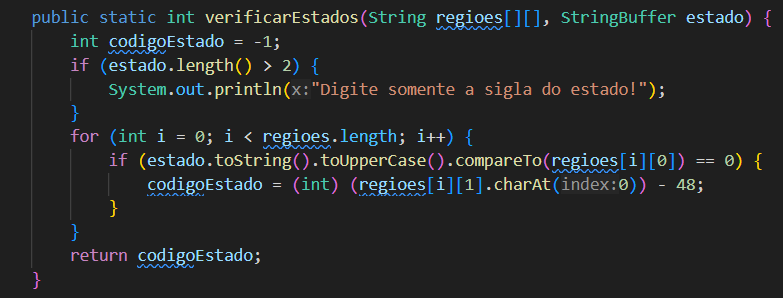
Este método é do tipo ***boolean*** a qual recebe 3 parametros:

* ***StringBuffer cpf***, é o CPF digitado pelo usuário.
* ***StringBuffer estado***, é o estado emissor do CPF digitado pelo usuário.
* ***String regioes[][]***, o vetor das regiões com as siglas dos estados e seus códigos de emissor.

O método realiza:

* Inicia com 2 verificações a qual nos podem gerar a exceção ***CPFCaracteresExcedente***:
  + A quantidade de caracteres digitadas pelo usuário ser diferente da quantidade de um CPF *(!=11)*;
  + O usuário digitar qualquer outro caractere que não seja número, como não podemos fazer a verificação via ***java.util.InputMismatchException***, já que estamos trabalhando com ***StringBuffer***.
* Cria e inicia a variável IndexCPF com valor igual a 0;
* Verificar se o 9º caractere (index 8) do objeto cpf é igual ao código fiscal de emissão de CPF do estado a qual o usuário digitou Através do método ***verificarEstados.*** Faço a conversão deste caractere, porém, devemos ter a diferença de 48, pois, quando fazemos esta conversão, ***char para int***, recebemos o valor decimal dele na tabela ***ascii***.
  + Feito isto iniciamos um looping, em que a variável ***indexCPF*** some o produto dos 9 primeiros caracteres e uma variável auxiliar que inicia em 10 e termina em 2.
  + Isso tudo nos resultará no primeiro digito do CPF (ex.: xxx.xxx.xxx-(**I)**x), através da subtração entre 11 por modulo de ***indexCPF por 11***, caso ***indexCPF*** seja maior ou igual a 10 o primeiro digito é 0;
* Após isso fazemos a verificação do 2 dígito do CPF:
  + Zeramos o ***indexCPF***, pois, o primeiro digito já foi confirmado.
  + Iniciamos um looping, em que a variável ***indexCPF*** some o produto dos 10 caracteres, os 9 primeiros caracteres e o 1 dígito, e uma variável auxiliar que inicia em 11 e termina em 2.
  + Isso tudo nos resultará no segundo digito do CPF (ex.: xxx.xxx.xxx-x(**I)**), através da subtração entre 11 por modulo de ***indexCPF por 11***, caso ***indexCPF*** seja maior ou igual a 10 o segundo digito é 0;
  + Após comparar se o segundo digito gerado é igual o valor do segundo digito digitado são iguais, é retornado ***true*** como valor, exibindo a mensagem CPF Válido. Caso contrário CPF inválido.

Como disse antes a obtenção do código fiscal do estado emissor é feito através deste método:

******

* Ele inicia verificando se o usuário digitou somente as siglas do estado que emitiu o CPF. Caso não ele exibe a mensagem “Digite somente a sigla do estado!” e retorna para o método ***verificarCPF*** o valor de -1.
* Passado esta verificação inicia um looping que a qual compara o estado digitado com os estados salvos no vetor ***regioes***. Caso ele não encontre o estado que o usuário digitou, significa que o estado não existe e retorna -1. Caso contrario é retornado o código fiscal do estado emitido.

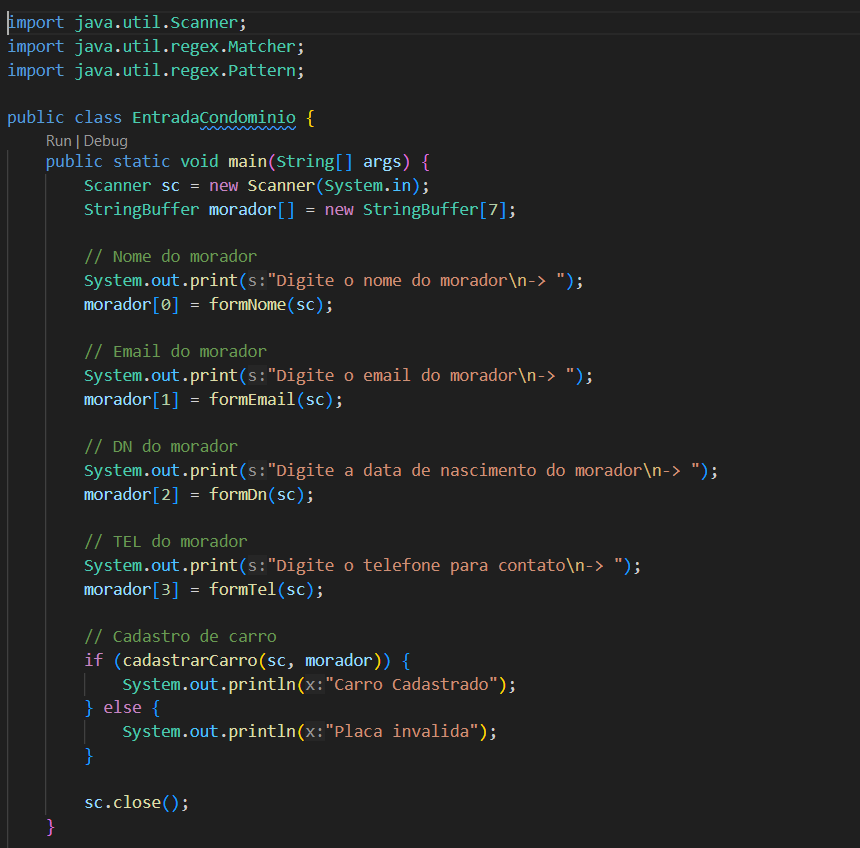
**Tratamento de exceções: Cadastro no estacionamento do condomínio.**

Este problema é o mais complexo, pois é solicitado um cadastro para pode utilizar o estacionamento de um condomínio. Para isso solicitamos 7 dados:

* **Nome completo.**
* **Email pessoal.**
* **Data de nascimento.**
* **Telefone.**
* **Modelo do carro.**
* **Marca do carro.**
* **Placa do carro**

Segue abaixo a resolução do problema:

**Método main():**

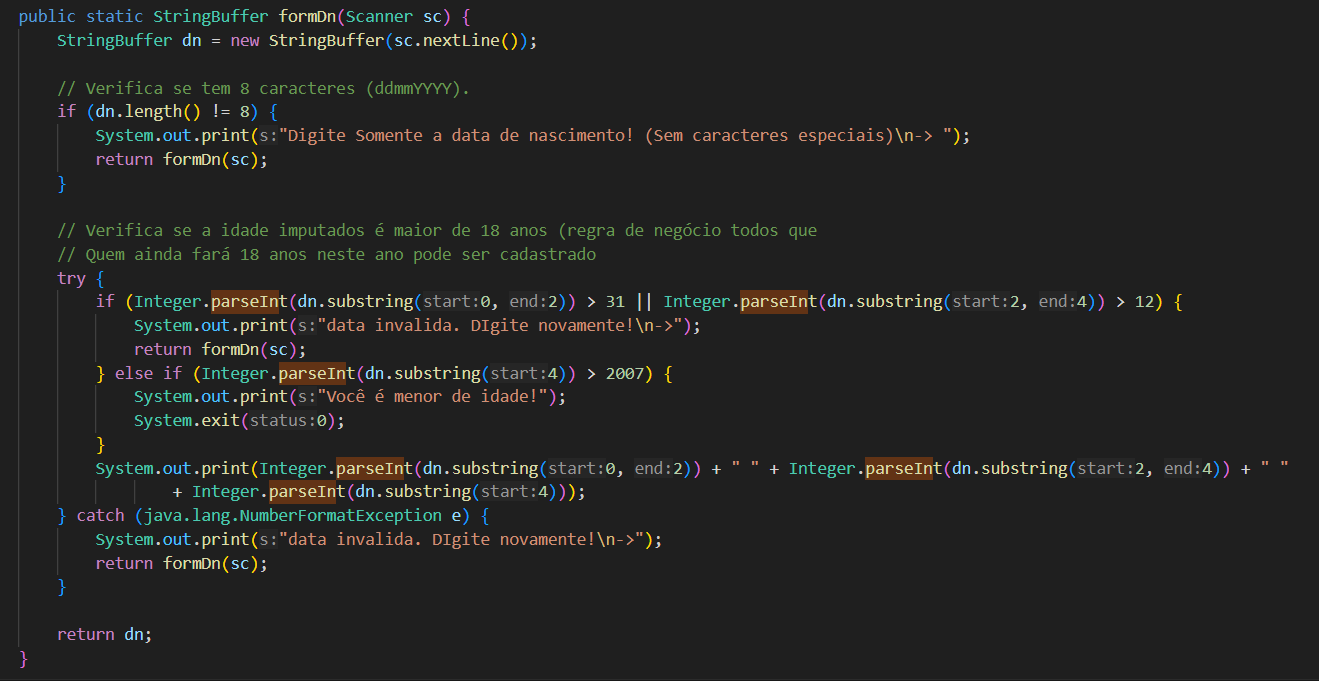


**Método formNome():**

****

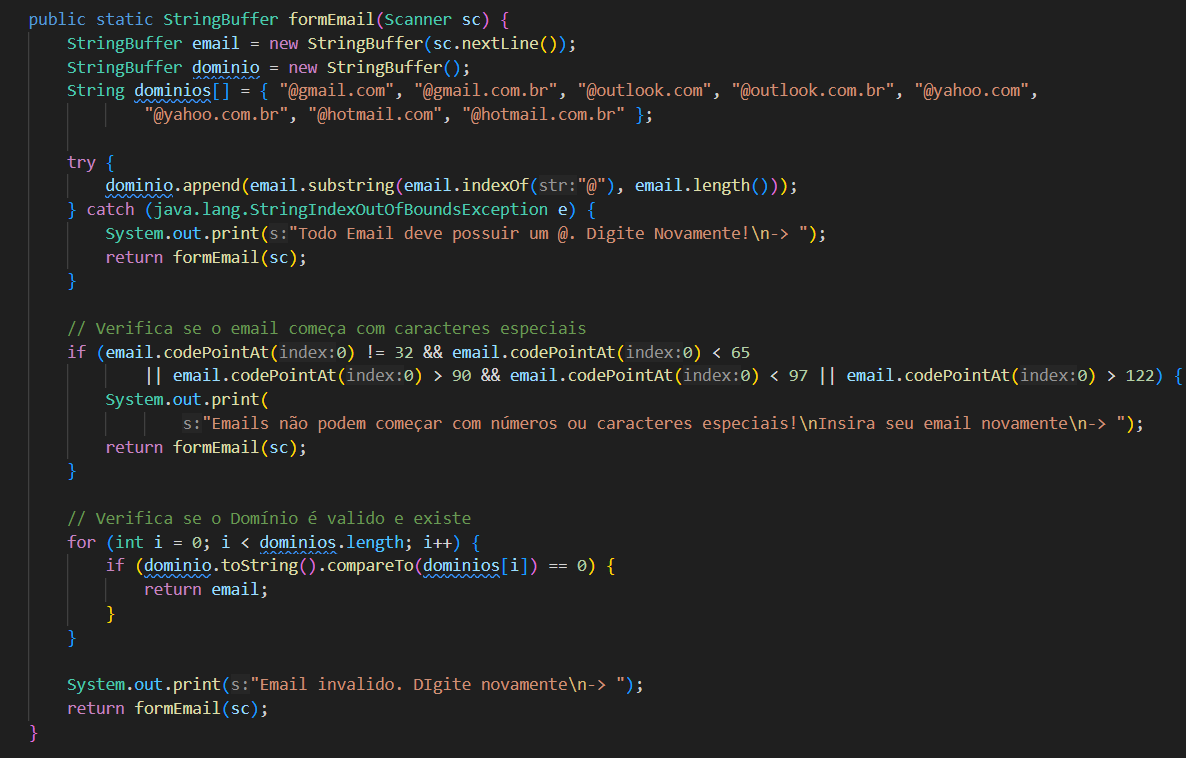
* Solicita nome e sobrenome obrigatoriamente.
* Solicita que o nome não deve caracteres especiais obrigatoriamente.
* Caso não seja atendido nenhuma dessas solicitações é solicitado a inserção do dado novamente
* Caso seja atendida nos retorna o valor verificado e aprovado para o formulário.

**Método formDn():**

****

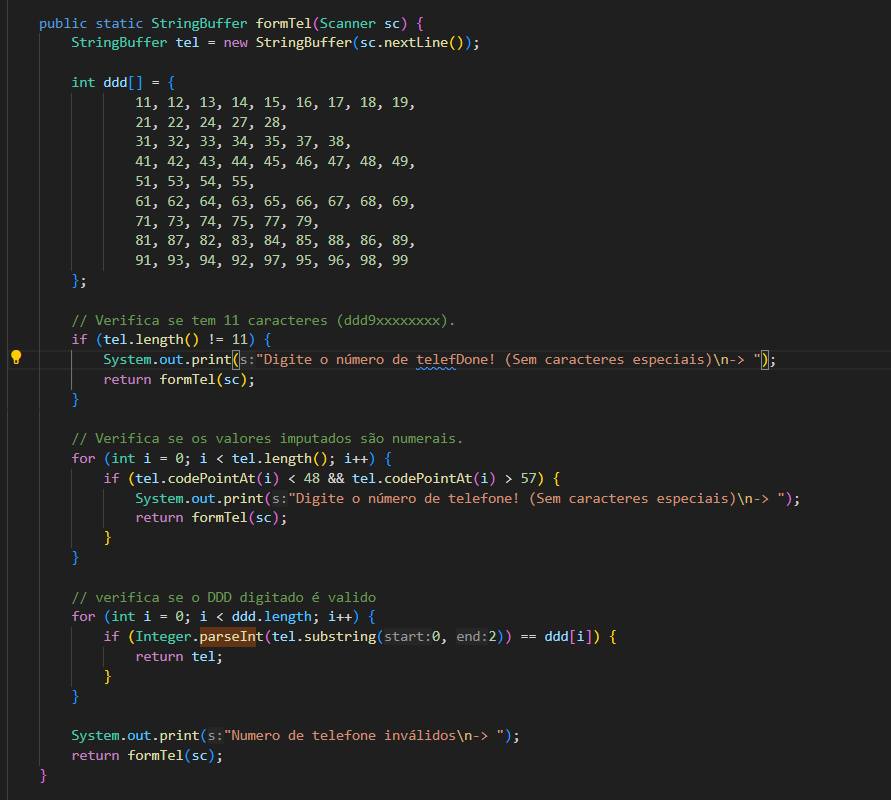
* Solicita a idade do usuário.
* Caso seja imputado algum valor errado nos solicita a inserção do dado novamente.
* Encerra o programa caso você seja menor de idade.
* Caso seja atendida nos retorna o valor verificado e aprovado para o formulário.

**Método formEmail():**

****

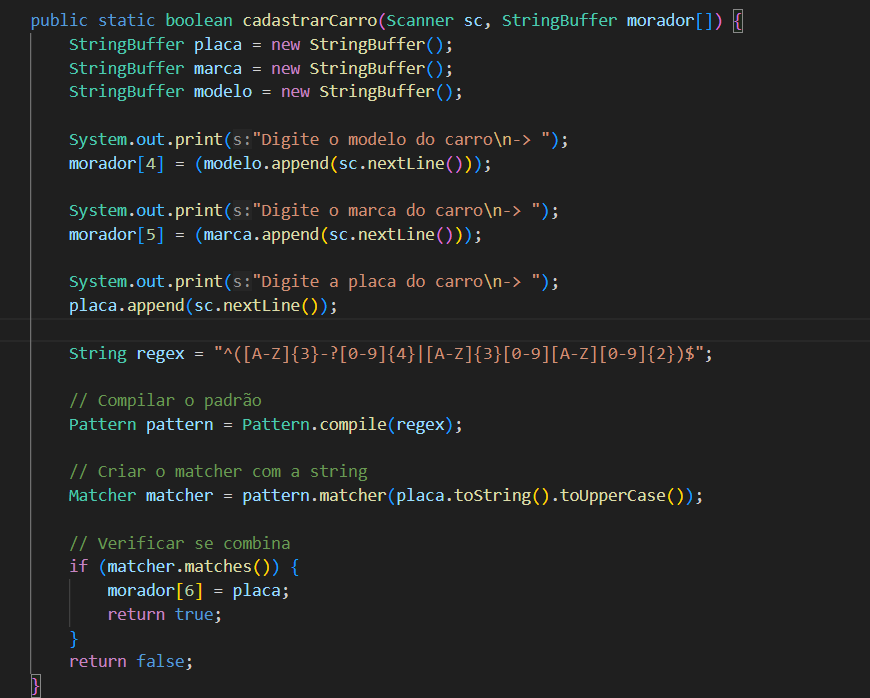
* Solicita um email pessoal, então somente os domínios dentro do vetor ***dominio[]*** são aceitos.
* Verifica se possui um domínio;
* Verifica se o email inicia com caractere especial;
* Caso seja atendida nos retorna o valor verificado e aprovado para o formulário.

**Método formTel():**



* Verifica se o telefone possui 11 digitos (DDD + número de telefone);
* Verifica se só tem valores numéricos;
* Verifica se o DDD existe no Brasil;
* Caso seja atendida nos retorna o valor verificado e aprovado para o formulário.

**Método cadastrarCarro():**

****

* Imputa os valores de modelo e carro;
* Utiliza regex para validar a placa   
  ***(regex = "^([A-Z]{3}-?[0-9]{4}|[A-Z]{3}[0-9][A-Z][0-9]{2})$";)***
* Verifica se combina a placa digitada com o padrão do Brasil ou Mercosul.