**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 Progemação da Faculdade Fatec Carapicuiba, como parte dos requisitos para obtenção nota ao segundo método de avalição da matéria (P2).**

**Area de Concentração: Linguagem de Programção**

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

Sumário

[Resumo 4](#_Toc198332034)

[Formulário de Login. 7](#_Toc198332035)

[Criando minha própria exceção. 11](#_Toc198332036)

# Resumo

Este trabalho tem como objetivo, explicar e exemplificar, a importância da utilização de tratamento de exceções em um ambiente de desenvolvimento de algum software, utilizado a linguagem de programação proposta pela professora, ***o JAVA***.

Iniciando com a resolução de um problema especifico a qual o professor nos proporcionou, o Cálculo do Fundo de Garantida de Tempo de Serviço (FGTS), de acordo com a Consolidação das Leis do Trabalho (CLT).

Além disso, foi realizado, por Philip Soares, a criação de 1 programas que utiliza de problemas reais, vindas por parte do usuário, que ajuda a não gerar qualquer tipo de problema na utilização do sistema.

E por último a criação de uma exceção exclusiva, utilizando os recursos do JAVA em um problema simples de resolução.**Cálculo do Fundo de Garantida de Tempo de Serviço (FGTS)**

Este problema foi proposto pela professora Sandra, para nos mostrar inicialmente como podemos estar fazendo os tratamentos de exceções em nossos softwares.

Um usuario deve nos informar 2 campos para:

* **NOME**: O nome que o usuario deve preencher segue algumas regras:
  + O nome completo do usuario.
  + Este nome deve ter no mínimo 5 caracteres e não pode ultrapassar dos 50.
  + Não deve nenhum caractere numeral no nome.

Para resolvermos isso, desenvolvi um método chamado de ***formNome,*** a qual possui um parâmetro do tipo Scanner para fazer a leitura dos dados do usuário.

Quando o usuário digita seu nome a primeira verificação de tamanho de nome inserido é realizada, identificando um erro é exibida uma mensagem alertando o usuário e o método é chamado novamente para ele reinserir os dados.

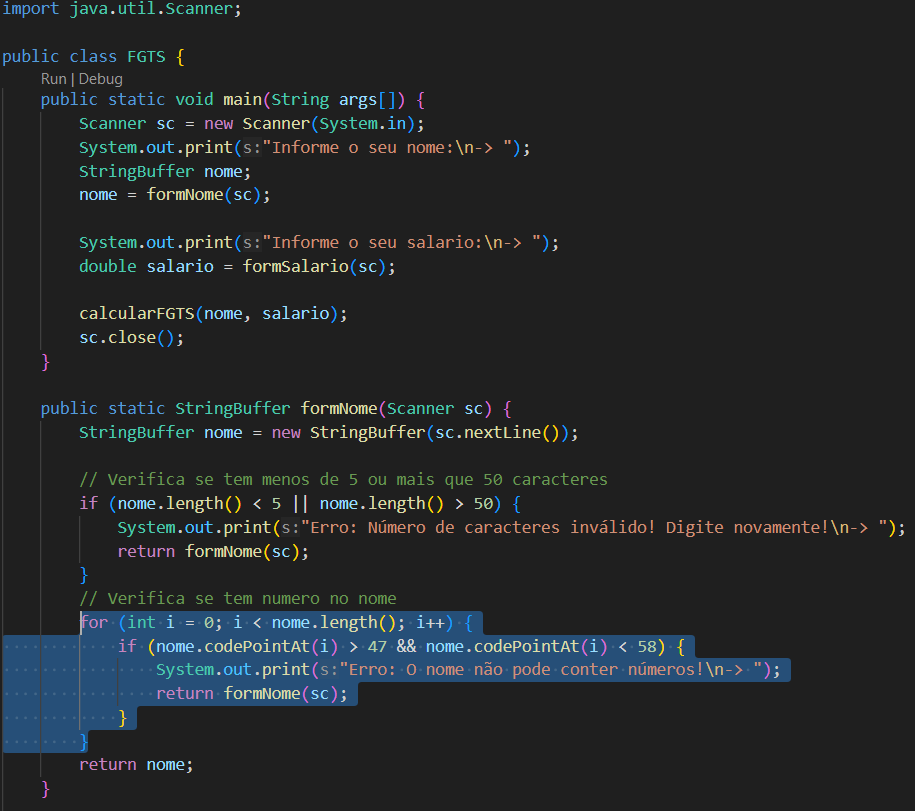
Ao atender este requisito o programa parte para a segunda verificação, a qual verifica a existência de um caractere numeral, verificando cada caractere e comparando se está dentro do *range* decimal dos números. Após isto o programa parte para capturar o salário do usuário.

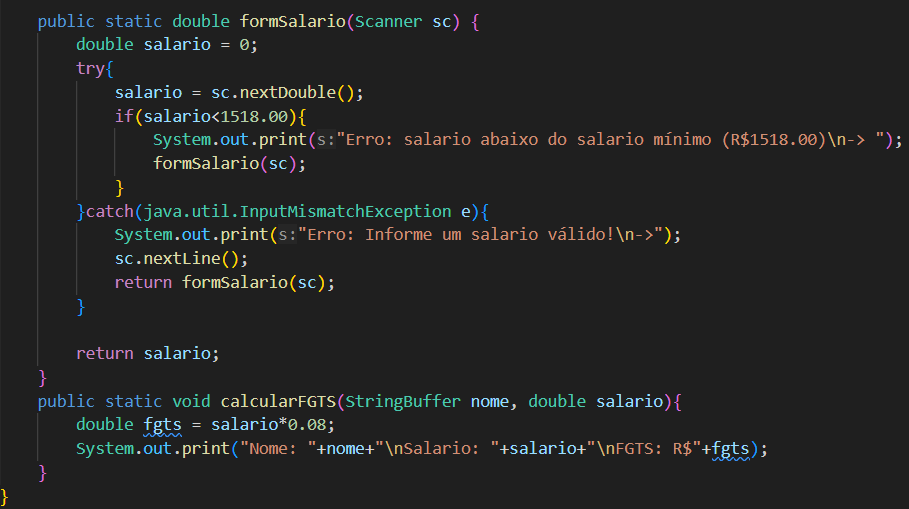
* **SALÁRIO**: O salário atual do usuário, no qual também é feito uma verificação para validar o valor digitado:
  + O salário deve ser acima do registrado na CLT atualmente (R$1518,00)
  + O usuário não deve digitar letras.

Para resolvermos isso, desenvolvi um método chamado de ***formSalario,*** a qual possui um parâmetro do tipo Scanner para fazer a leitura dos dados do usuário.

Quando o usuário digita seu nome a primeira verificação do valor do salário inserido é realizada, identificando um erro é exibida uma mensagem alertando o usuário e o método é chamado novamente para ele reinserir os dados de nome.

Ao atender este requisito o programa parte para a segunda verificação, a qual verifica a existência de um caractere alfabéticos, como os tipos informados são diferentes o JAVA a caba gerando uma exceção “**java.util.InputMismatchException**”, que ao ser captura gera uma mensagem de alerta ao usuário e o método é chamado novamente para ele reinserir os dados do salário.



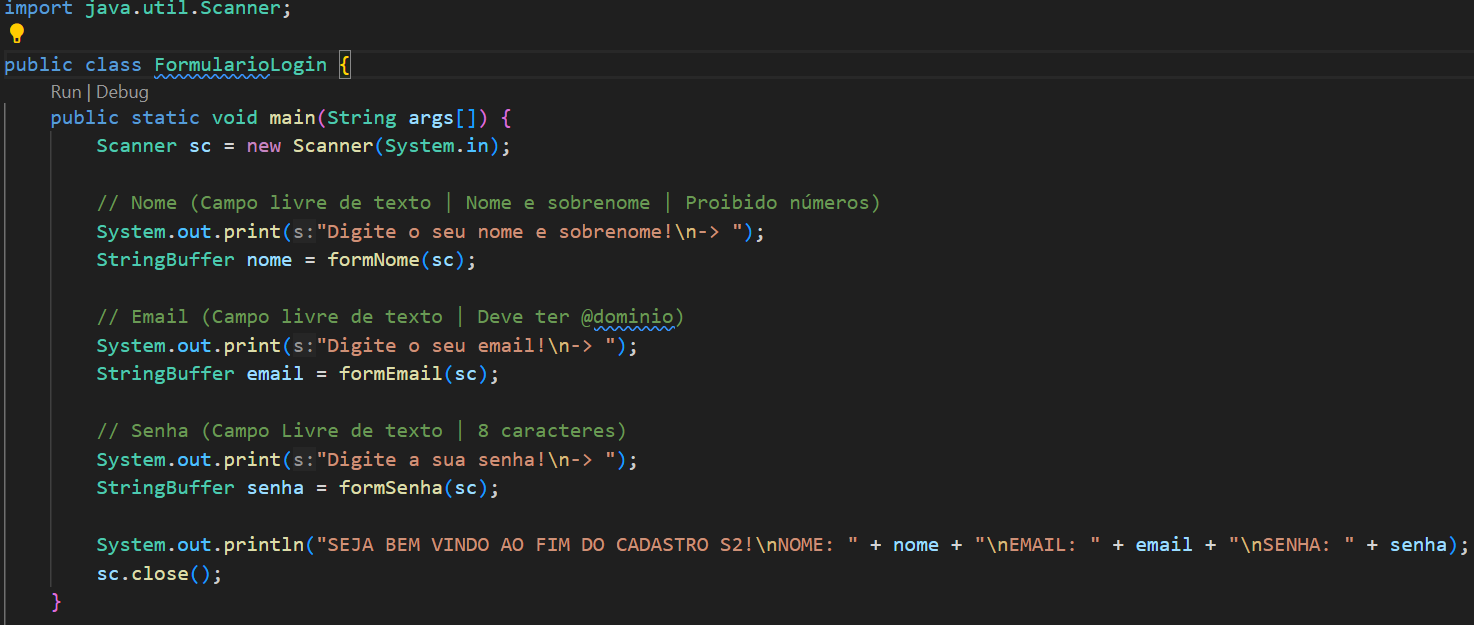


# Formulário de Login.

Meu primeiro Programa é um é formulário de login básico de um site.  
O usuário deve inserir seu nome, e-mail e senha e o programa deve fazer a verificações de todos os campos, em caso de erros, retornar uma mensagem ao usuário para reescrever a sua resposta.

**METODO MAIN**

O método principal é um bloco simples contendo a instanciação do objeto scanner (responsável por nos deixar inserir dados via console) e a chamada de 3 metodos nomeados de ***formNome, formEmail, formSenha***. Estes são responsáveis por realizar o tratamento de exceções e possíveis erros de usuário.

****

**Método formName(sc):**

**\* @param sc Scanner para leitura do nome**

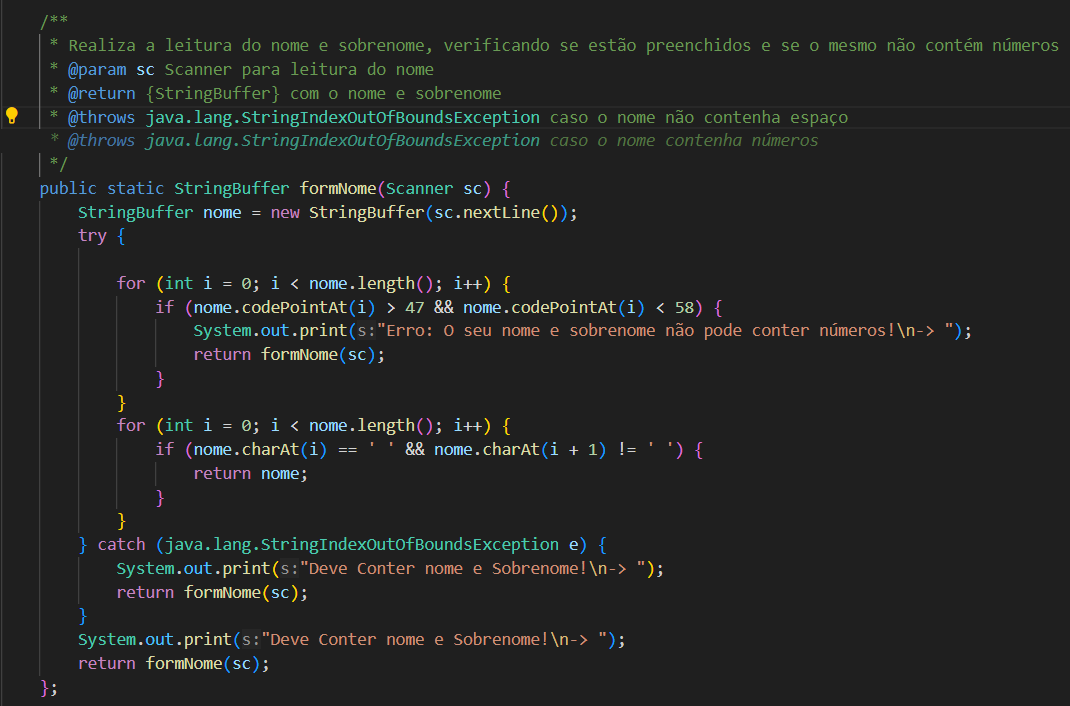
**\* @return {StringBuffer} com o nome e sobrenome**

**\* @throws java.lang.StringIndexOutOfBoundsException caso o nome não contenha espaço**

Este método é responsável por fazer a captura do nome e sobrenome que o usuário digitar e faz a verificação de 2 possibilidades de erros:

* O nome digitado possui um algum caractere numérico, ao ser identificado na linha 39, retornará uma mensagem de alerta ao usuário, “Erro: o seu nome e sobrenome não podem conter números!”, e pede para o mesmo reinseri-lo.
* Caso o usuário escreva somente o primeiro nome, o programa irá gerar um erro, ***java.lang.StringIndexOutOfBoundsException***, ao ser identificado na linha 48 retornará uma mensagem de alerta, “Deve Conter nome e Sobrenome!”, e pedirá para o usuário reinseri-lo.

Caso tudo ocorra tudo certo o programa irá solicitar o próximo campo.

****

**Método formEmail(sc):**

**\* @param sc Scanner utilizado para leitura do e-mail.**

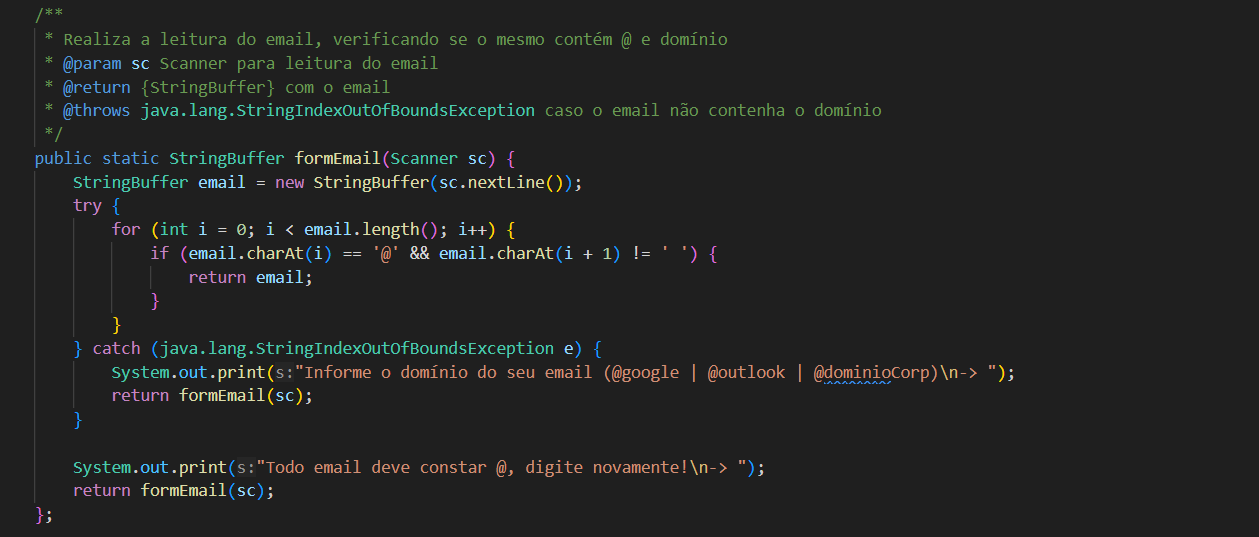
**\* @return {StringBuffer} contendo o e-mail validado.**

**\* @throws java.lang.StringIndexOutOfBoundsException caso o e-mail não contenha um domínio válido após o caractere @.**

Este método é responsável por capturar o e-mail digitado pelo usuário e verificar se ele contém obrigatoriamente o caractere @ seguido de um domínio (como @google, @outlook ou @dominioCorp).

Caso o caractere @ esteja presente e logo após ele não haja um espaço, o e-mail é considerado válido e será retornado. Se o caractere @ estiver ausente ou mal posicionado, o método exibe a mensagem de erro:

* “Todo email deve constar @, digite novamente!” e solicita que o e-mail seja digitado novamente.
* Se houver um erro de índice ao acessar um caractere após o @ (por exemplo, quando o @ estiver no final da string), será lançada a exceção StringIndexOutOfBoundsException, que é tratada com a exibição da mensagem:  
  “Informe o domínio do seu email (@google | @outlook | @dominioCorp)”, e o método solicita uma nova entrada do e-mail.



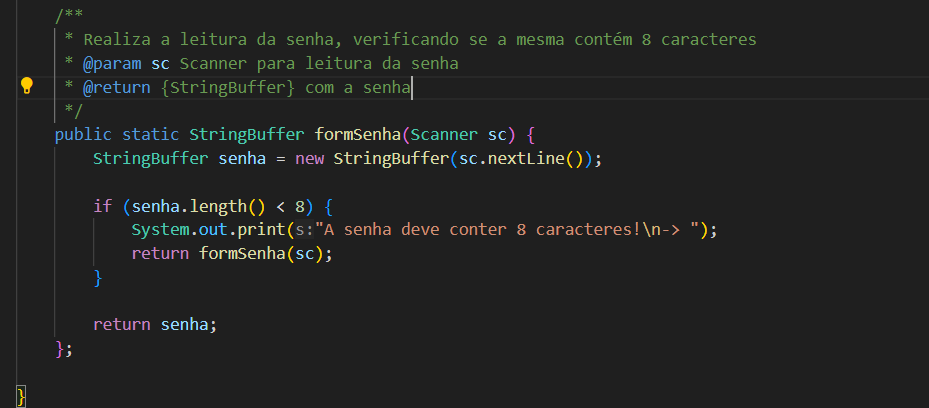
Método formSenha(sc):

\* @param sc Scanner utilizado para leitura da senha.

\* @return {StringBuffer} contendo a senha validada.

Este método realiza a leitura da senha digitada pelo usuário e verifica se ela contém no mínimo 8 caracteres. Caso a senha digitada possua menos de 8 caracteres, o método exibe a seguinte mensagem de erro: “A senha deve conter 8 caracteres!” Em seguida, solicita ao usuário que digite a senha novamente.

Se a senha atender ao requisito de comprimento mínimo, ela é considerada válida e será retornada pelo método.



# Criando minha própria exceção.

Para exemplificar como podemos criar nossas próprias exceções, *desenvolvi este simples programa:*

Ele nos pergunta se queremos acessar algum sistema fictício, e d*evemos responder com a letra ‘s’* para entrar, caso não respondemos com este único parâmetro, o programa nos retorna um alerta avisando que a resposta é invalide e encerra o programa.

Caso respondemos da maneira certa o programa nos pergunta qual idade possuímos e devemos responder que somo maiores de idade, caso contrario ele nos alerta que somos menores de idade e que não podemos entrar assim encerrando o programa.

