Java Redes — Atividade 01

UTFPR – Universidade Tecnológica Federal do Paraná
Prof. Titular Dr. Rogério Santos Pozza • Prof. Adjunto Esp. Djeizon Barros

Enunciado da atividade

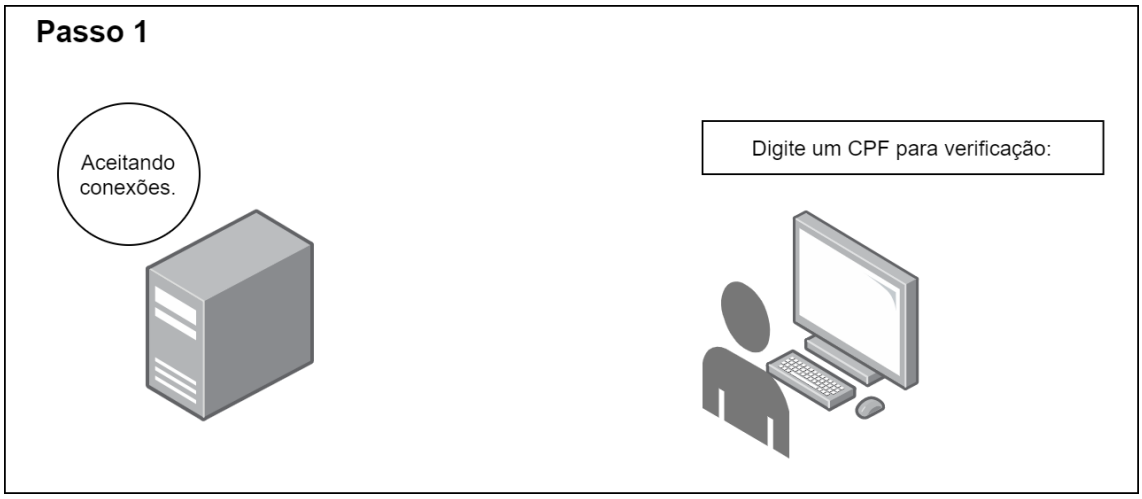
Faça um programa baseado em **Sockets**, capaz de realizar o seguinte:

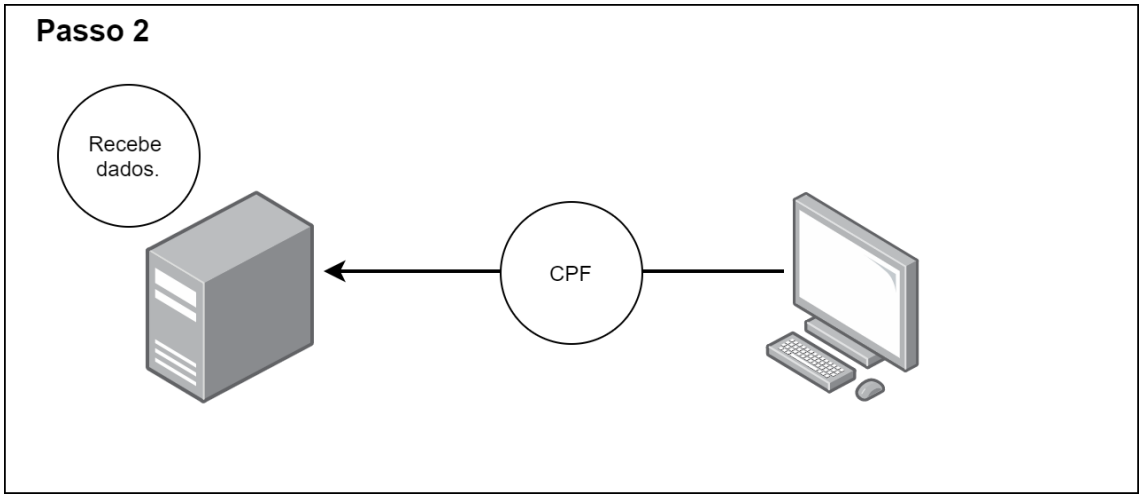
a) A máquina **cliente** deverá instar o usuário no *terminal* ou *janela*, pedindo:

> **Digite um CPF para verificação:**
> O usuário deverá entrar um número de CPF.

b) A máquina que é o servidor receberá o CPF e verificará se o número é válido ou inválido. Se o **CPF** for válido, o servidor deverá retornar a mensagem "**Este CPF é válido**" para o cliente. Se o **CPF** for inválido, o servidor deverá retornar a mensagem "**Este CPF é inválido.**" em seu console.

Ilustração do que o programa deve fazer

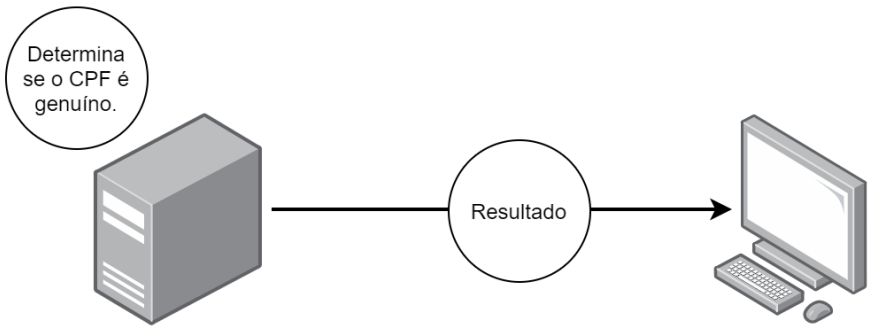




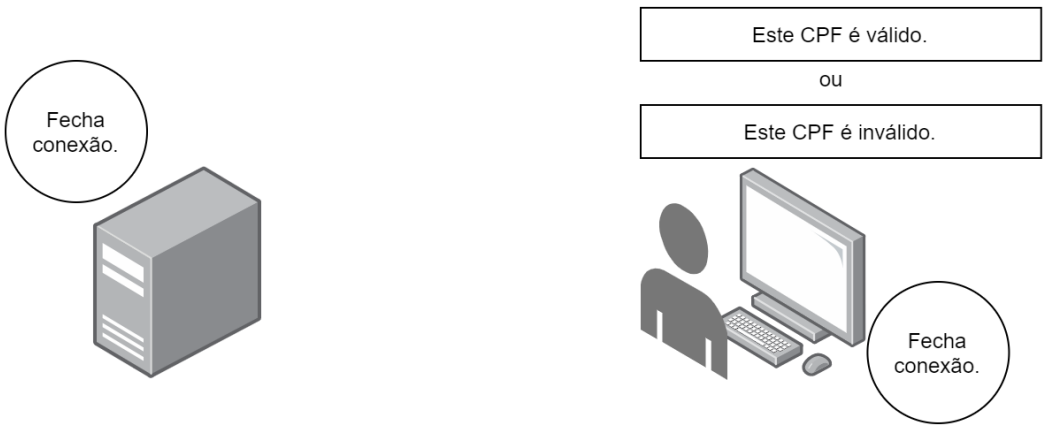
UTFPR – Universidade Tecnológica Federal do Paraná
Prof. Titular Dr. Rogério Santos Pozza • Prof. Adjunto Esp. Djeizon Barros

Passo 3

Passo 4



Passo 5



Regras para construir o programa

Faça apenas uma classe **Servidor** e uma classe **Cliente.** Não utilize nomes diferentes destas classes, para facilitar a correção. O cliente tratará de capturar a entrada do usuário, enviar o CPF; e, recuperar o CPF quando o servidor responder se o CPF é válido ou inválido. O servidor tratará de receber o CPF que vem do cliente, tratar os dados, conferindo se é um CPF genuíno ou não, e devolver o resultado – **no fim de tudo, o servidor deverá fechar a conexão.**

Todo tratamento do CPF deverá ser feito no servidor. Não utilize nenhuma classe externa auxiliar. O programa precisa interagir com o usuário via entrada do teclado. O endereço de IP do *localhost*, ou seja, do escopo **127/8**, mais precisamente o IP **127.0.0.1**, deverá ser o IP entrado neste exercício e não o endereçamento de sua **LAN** (ex. 192.168.10.10). A porta deverá ser uma porta efêmera, **50000**.

Este programa pode ser feito no **modo console** ou com **interface gráfica** (sem o uso de arquivos `.form`), desde que os passos **1 a 5** sejam observados em **Cliente** e **Servidor** – cada um com sua verbosidade de **`output`**.

Dicas para validar o CPF

O algoritmo que verifica o **CPF** segue a seguinte lógica: O **CPF** é composto por **11 dígitos**. Os últimos **2 dígitos**, são dígitos verificadores, obtidos por duas etapas de cálculo. Observe o **CPF** a seguir. O destaque em vermelho aponta os dois dígitos supracitados.

222.222.222-22

Para obter o valor do primeiro dígito verificador, é realizado um cálculo com base nos **9 primeiros** dígitos. Para obter o valor do segundo dígito verificador, é realizado um cálculo com base nos **9 primeiros** em **conjunto com o número obtido no primeiro cálculo. Obtendo o primeiro dígito verificador:**

O cálculo inicia-se da direita para a esquerda. O multiplicador começa em 2 e incrementa-se em 1. O resultado é observado na última linha (linha verde).

Digito 9	Digito 8	Digito 7	Digito 6	Digito 5	Digito 4	Digito 3	Digito 2	Digito 1
2	2	2	2	2	2	2	2	2
x 10	x 9	x 8	x 7	x 6	x 5	x 4	x 3	x 2
20	18	16	14	12	10	8	6	4

A **soma** de todos os números na última linha vale o número **108**.

Este número obtido deverá ser dividido por **11**.

108
÷ 11
9, **818181818181818181818181818**

Se o resto dessa divisão **for menor** que o número **2** (na prática, se o **primeiro número do resto** da divisão for o número *1* ou *0*), o primeiro dígito verificador no **CPF** será **0** (zero – e deverá ser usado para a próxima conta do segundo dígito verificador). Porém, se o resto dessa divisão for **igual ou maior que 2**, o primeiro dígito verificador será o resultado de **11** subtraindo o valor do módulo daquela divisão. Para obter o módulo, realize a conta **0,818181818181818181818181818 x 11.** O resultado é o módulo **9**. Não se preocupe, pois o **Java** já possui um operador só para cuidar dessa conta.

Momento em que se encontra o **primeiro dígito** verificador
11
-9
2

Em outras palavras: se o seu dígito não for *0 (zero)*, **você acabou de descobrir o penúltimo número do seu CPF**, caso o número de seu CPF seja válido.

Obtendo o segundo dígito verificador:

O novo cálculo inicia-se da direita para a esquerda. O multiplicador começa agora com o dígito verificador encontrado, adicionando uma posição na direita. Mantém-se os 9 dígitos anteriores, mas agora a casa do dígito **9** tem mais um incremento que é multiplicado por **11**. O resultado é observado na última linha.

Digito 9	Digito 8	Digito 7	Digito 6	Digito 5	Digito 4	Digito 3	Digito 2	Digito 1	Digito Verificador Achado
2	2	2	2	2	2	2	2	2	2
× 11	× 10	× 9	× 8	× 7	× 6	× 5	× 4	× 3	× 2
22	20	18	16	14	12	10	8	6	4

A **soma** de todos os números na última linha vale o número **130**. Este número obtido deverá ser dividido por **11**.

130
÷ 11
11, **8181818181818181818181818182**

Como este resto também maior que **2**. O dígito novamente não será **0** (se zero – assim finalizaria sua numeração), e sim novamente o número **11** subtraindo o resto encontrado dessa divisão, considerando como valor para subtração, o valor do módulo desta divisão.

Para obter o módulo, realize a conta **0,8181818181818181818181818182 x 11.** O resultado é o módulo **9**.

Momento em que se encontra o **segundo dígito** verificador
11
-9
2

Em outras palavras: **você acabou de descobrir o último número do seu CPF**, caso o número de seu **CPF** seja válido.

Então, para todo **CPF** informado, o mesmo pode ser validado somente com os 9 primeiros dígitos, **que renderão os 2 últimos**. O algoritmo é comparar se essa conta rende realmente os dois números informados pelo usuário. Se não render, o **CPF** é inválido. Se render, o **CPF** é válido.

Ainda é necessária mais uma validação: precisa-se validar se o CPF possui 11 dígitos. Se possuir menos ou mais dígitos, **o CPF é inválido**. Um **CPF** só é válido com 11 dígitos.

Outra validação que se precisa ater-se é que a sequência de 11 zeros é inválida para o **CPF**, ou seja, um CPF **000.000.000-00** é inválido. Outras sequências com o mesmo número não são necessariamente um CPF inválido. Ou seja: O CPF que usamos para fazer esse cálculo, o **222.222.222-22** é um **CPF válido!**

Para fixar este algoritmo na sua mente, faça os cálculos usando o seu próprio CPF. Se validar o **CPF** ainda é algo desafiador para você, nesse exercício, você pode seguir as dicas no apêndice deste documento e incorporá-las na sua classe **Servidor**.

Regras da Entrega

Atenção, seu projeto deverá observar **3 coisas especiais:**

1) O seu projeto deverá ter **seu nome como autor**, em um comentário. A classe que não tiver seu nome como autor, renderá desconto global no exercício. Exemplo:

```
/**
 *
 * @author Djeizon Barros
 *
 */
```

2) O seu projeto deverá ser entregue com a package **`local.javaredes`**

```
package local.javaredes;
```

O aplicativo que for entregue com *package* diferente, receberá desconto.

3) Anexe na atividade, os arquivos:

- **Servidor.java** (código fonte)
- **Cliente.java** (código fonte).

Somente isso. **Não compactar nenhum arquivo.** O envio compactado, como é exigido nas outras disciplinas, **mas não nesta**, acarretará desconto.

Submeta **os dois arquivos** no ambiente de ensino.
Não deixar a tarefa em Modo Rascunho.
Clique no botão **Enviar Tarefa por Definitivo**.

Atenção para as penalidades no exercício

Quando o exercício recair exatamente numa dessas situações ou acumular erros, **os descontos abaixo serão aplicados.**

Situação de descontos cumulativos	Desconto
Classes (ou uma delas) não possuem um comentário indicando sua autoria.	10,00
Entregou o exercício com *package* diferente.	15,00
Entregou o exercício em formato ZIP ou RAR ou outro, que é pedido nas outras disciplinas, **mas não nesta.**	15,00
Foi utilizada porta baixa ou porta reservada – o programa só funciona com a troca da porta.	40,00
Foi utilizado um IP de LAN – e não um endereço do escopo 127/8 – o programa só funciona com a correção para o *localhost*.	40,00
O programa não fecha o servidor após o final, bloqueando a porta numa segunda tentativa de rodar o programa.	40,00
Somente servidor funciona e entra no estado bloqueante, cliente não compila ou nem funciona.	60,00
Servidor não retorna mensagem para cliente nem retorna mensagem para o console, mas o cliente funcionou enviando o objeto CPF.	60,00
Existe a conexão entre o servidor e cliente, mas o CPF é parcialmente validado (ex. só verifica número de dígitos) ou não é validado em nenhum momento.	60,00
O código não compila. O código é incompreensível.	100,00
Cópia de outro aluno, não importando se tudo está correto.	100,00

Apêndice

Dica 01: Para verificar o **CPF** no servidor, você precisa apenas do método **`main`** e dois outros métodos **`boolean`**. Lembre-se que os métodos **`boolean`** retornam valores (**`return`**) verdadeiros ou falsos. Com valores **`boolean`**, pode-se definir a resposta adequada sobre determinado dado recebido.

Dica 02: No cliente, você pode implementar a interação com o usuário via console ou via janela gráfica, tanto para a entrada de dados quanto para exibição da resposta do servidor.

Dica 03: Se você tiver muita dificuldade de implementar os métodos pedidos, você poderá realizar uma pesquisa para estudar como se implementa o código, utilizando os seguintes recursos: [A] [B] [C] [D]

Com tudo o que foi explicado aqui, seguindo o algoritmo de cálculo e as dicas, você consegue facilmente implementar sem ajuda de pesquisa, porém se você sentir muita dificuldade na implementação ou necessitar de alguma dica a mais, você pode reutilizar código que aparece em um dos links acima. No entanto, você não poderá implementar uma terceira classe para dentro do seu programa. A validação terá que acontecer no arquivo **Servidor.java**, que é perfeitamente possível. Pesquise e adote o código que achar mais otimizado, mas não viole o enunciado do exercício utilizando classe externa tal como a classe *ValidaCpf.java*, ou semelhante.

O algoritmo é relativamente simples e pode ser implementado como já mencionado, com 1 método **`main`**, e dois métodos **`boolean`**.

Dica 04: Não é necessário o tratamento de erros na entrada de dados, bastando apenas uma sequência numérica de CPF, porém, o aluno permanece à vontade para tratar a entrada de dados como bem entender: recebendo pontos ou hifens, do mesmo modo como se fosse implementar um programa real para uso em um escritório, realizando todos os tratamentos. O tratamento é opcional.