
SUMÁRIO

1. INTRODUÇÃO	1
2. IMPLEMENTAÇÃO	1
2.1. SERVIDOR	1
2.1.1. InterfaceServidor.java	2
2.1.2. ServidorMat.java	2
2.1.3. ArrancaServidor.java	3
2.2. CLIENTE	3
2.2.1. InterfaceServidor.java	3
2.2.2. Cliente.java	4
3. EXECUÇÃO	4
4. DIFICULDADES ENCONTRADAS	5
5. LINK NO GITHUB	5
REFERÊNCIA BIBLIOGRÁFICA	5

1. INTRODUÇÃO

Este relatório descreve a implementação de um programa utilizando o método RMI. RMI é uma sigla correspondente a *Remote Method Invocation*. Trata-se de uma interface para a criação de aplicativos distribuídos em JAVA. Nesta aplicação, realiza-se a conexão entre dois computadores distintos, por meio de uma arquitetura cliente-servidor. O cliente da aplicação passa ao servidor três valores numéricos - chamados de “a”, “b” e “c”, correspondentes aos coeficientes de uma equação de segundo grau, do tipo $ax^2 + bx + c = 0$ - para que o servidor realize a tarefa de, por meio da fórmula de Bháskara, encontrar as raízes da equação de segundo grau correspondente.

Nesta implementação, *cliente* e *servidor* rodam na mesma máquina, em terminais distintos.

2. IMPLEMENTAÇÃO

A implementação da aplicação deu-se em ambiente Windows. Por se tratar de uma aplicação JAVA, faz-se necessária a instalação do JAVA no computador. As subseções a seguir descrevem características dos arquivos de código que compõem a aplicação.

2.1. SERVIDOR

O lado servidor da aplicação é composto por três arquivos distintos: *ArrancaServidor.java*,

InterfaceServidor.java, *ServidorMat.java*. A seguir, detalham-se suas características.

2.1.1. InterfaceServidor.java

Este arquivo é composto apenas pelo cabeçalho da função *bhaskara*, que é implementada em *ServidorMat.java*.

```
public String bhaskara(double a, double b, double c) throws  
RemoteException;
```

2.1.2. ServidorMat.java

Neste arquivo, além da função homônima, que apenas informa que o servidor foi instanciado, está também a função *bhaskara*. É essa a função, presente no lado do servidor, que executa, de fato, o que o cliente solicita. Para isso, recebe em seus parâmetros o valor de *a*, *b* e *c*; em seguida, verifica se o valor de delta é positivo e, caso não seja, informa da impossibilidade de calcular a equação; por fim, efetua os cálculos de cada uma das duas raízes da equação e retorna esse valor, de maneira organizada.

```
public ServidorMat() throws RemoteException  
{  
    System.out.println("Novo Servidor instanciado...");  
}
```

```
public String bhaskara(double a, double b, double c) throws  
RemoteException  
{  
    double delta, x1, x2, raiz_delta;  
    delta = (b*b)-4*a*c; //Cálculo do valor de delta  
    if (delta < 0){ //Verifica se negativo. Se for, avisa que é  
impossível calcular.  
        return "Valor de delta eh inferior a 0. Impossivel  
calcular.";  
    }  
    else{  
        raiz_delta = Math.sqrt(delta);  
        x1 = ((b*(-1))-(raiz_delta))/2*a;  
        x2 = ((b*(-1))+(raiz_delta))/2*a;  
        System.out.println("Valores recebidos do cliente: " + a, b,  
c);  
        System.out.println("Equação: (" + a + ")x^2 + (" + b + ")x +  
(" + c + ") = 0");  
        return "x1 = " + x1 + " || x2 = " + x2;  
    }  
}
```

```
}  
}
```

2.1.3. ArrancaServidor.java

É neste arquivo que está a classe `public static void main(String argv[])` da aplicação, portanto, conforme será abordado na seção 3, é aqui que dá-se a execução do servidor. O código, apresentado a seguir, associa o servidor à porta (aqui, a 1099) e também o associa a um nome para que o cliente da aplicação seja capaz de acessá-lo.

```
public static void main(String argv[])  
{  
    try  
    {  
        System.out.println("Subindo servidor...");  
        InetAddress IP = InetAddress.getLocalHost();  
        System.setProperty("java.rmi.server.hostname",  
IP.getHostAddress());  
        System.out.println(IP.getHostAddress());  
        Registry r =  
java.rmi.registry.LocateRegistry.createRegistry(1099); // Registra a  
porta da aplicação  
        Naming.rebind("ServidorMat_1", new ServidorMat()); //  
Associa o servidor a um nome para o acesso do cliente  
    }  
    catch (Exception e)  
    {  
        System.out.println("Ocorreu um problema no arranque do  
servidor.\n"+e.toString());  
    }  
}
```

2.2. CLIENTE

Do lado do cliente, são dois os arquivos: *InterfaceServidor.java*, *Cliente.java*. A seguir, detalham-se suas características.

2.2.1. InterfaceServidor.java

Idêntico ao arquivo homônimo presente na parte do servidor, este arquivo é composto apenas pelo cabeçalho da função *bhaskara*, que é implementada em *ServidorMat.java*.

```
public String bhaskara(double a, double b, double c) throws  
RemoteException;
```

2.2.2. Cliente.java

Em `Cliente()`, executa-se a conexão com o servidor, passando o endereço e a porta da conexão. Verifica-se o sucesso dessa conexão; caso ocorra, chama-se a função `bhaskara` por meio da interface de comunicação entre este cliente e o servidor que realizará os cálculos solicitados.

```
public Cliente()  
{  
    System.out.println("Executando Cliente... \n");  
    try  
    { // Acessa o servidor de nomes para localização das  
      funções remotas  
        msi = (InterfaceServidorMat)  
Naming.lookup("rmi://192.168.56.1:1099/ServidorMat_1");  
    }  
    catch (Exception e)  
    {  
        System.out.println("Falhou a execucao do  
Cliente.\n"+e);  
        System.out.println("Certifique se a aplicacao no  
servidor esta em execucao.\n");  
        System.exit(0);  
    }  
}
```

Adicionalmente, neste arquivo, em sua *main*, instancia-se o `Cliente` e faz-se as leituras dos três valores da equação para cálculo.

3. EXECUÇÃO

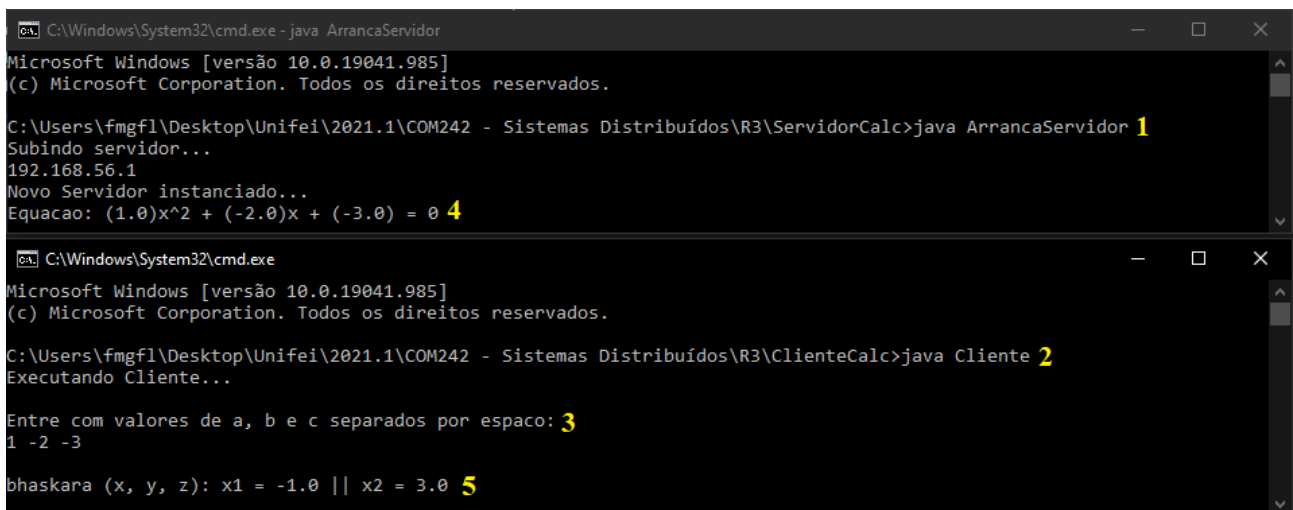
Os arquivos de *cliente* e de *servidor* estão separados em duas pastas distintas. Para execução da aplicação, deve-se abrir uma interface do *cmd* em cada uma dessas pastas. É importante garantir que nenhum *firewall* esteja bloqueando a execução do programa.

Após abrir o terminal, deve-se compilar todos os arquivos de código das pastas. Isso é feito por meio do comando `javac nomeArquivo.java`. Em seguida, no terminal aberto na pasta do servidor, compila-se `rmic ServidorMat`. Por fim, executa-se, em suas respectivas pastas, nesta ordem, os seguintes comandos: `java ArrancaServidor` e `java Cliente`.

Visualize, na Figura 1, o resultado de uma execução. Verifique a ordem de execução de

acordo com a numeração indicada na figura em cor amarela. A janela superior corresponde ao servidor e a janela inferior ao cliente.

1. O primeiro passo, após as compilações supracitadas, é a execução do servidor. Na janela, nas linhas inferiores à linha de execução, mostra-se o servidor sendo subido, seu IP e a mensagem de que foi instanciado.
2. Em seguida, executa-se o cliente. A aplicação retorna o aviso de que ele está em execução;
3. Do lado do cliente, solicita-se a inserção de três valores numéricos para a equação;
4. Do lado do servidor, mostra-se os valores recebidos;
5. Cliente recebe o resultado do trabalho do servidor: as raízes da equação informada.



```
C:\Windows\System32\cmd.exe - java ArrancaServidor
Microsoft Windows [versão 10.0.19041.985]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\fmgfl\Desktop\Unifei\2021.1\COM242 - Sistemas Distribuídos\R3\ServidorCalc>java ArrancaServidor 1
Subindo servidor...
192.168.56.1
Novo Servidor instanciado...
Equacao: (1.0)x^2 + (-2.0)x + (-3.0) = 0 4

C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.19041.985]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\fmgfl\Desktop\Unifei\2021.1\COM242 - Sistemas Distribuídos\R3\ClienteCalc>java Cliente 2
Executando Cliente...

Entre com valores de a, b e c separados por espaço: 3
1 -2 -3

bhaskara (x, y, z): x1 = -1.0 || x2 = 3.0 5
```

Figura 1: Demonstração da execução do código.

4. DIFICULDADES ENCONTRADAS

Durante a confecção do trabalho, houve alguns pequenos atritos com o JDK, facilmente resolvidos. O *firewall* também apresentou alguns problemas, que foram igualmente solucionados sem maiores dificuldades.

5. LINK NO GITHUB

Os códigos desta aplicação podem ser encontrados no Github por meio do seguinte link: <https://github.com/fmgflavio/RMI.git>.

REFERÊNCIA BIBLIOGRÁFICA

Material sobre RMI disponibilizado no Classroom desta disciplina de Sistemas Distribuídos pelo Prof. Dr. Rafael Frinhani.