Trabalho Técnico

Sistema e Software:

Windows 10;

Eclipse IDE for Java Developers (includes Incubating components)

Version: 2020-06 (4.16.0)

Build id: 20200615-1200

Intervenções:

**Prática 1**

Ao executar o arquivo *CalculadoraClientHTTP.java*, na linha 32 (“writer.write("oper1=15&oper2=15&operacao=3"); //1-somar 2-subtrair 3-dividir 4-multiplicar” ), são passados os argumentos da operação e o tipo de operação.

O comentário está trocado. Para hoje a operação 3 está multiplicando, ver Figura 1.1 para resultado no console, e a operação 4 está dividindo, ver Figura 1.2 para resultado no console. A execução está diferente do que está colocado no comentário.

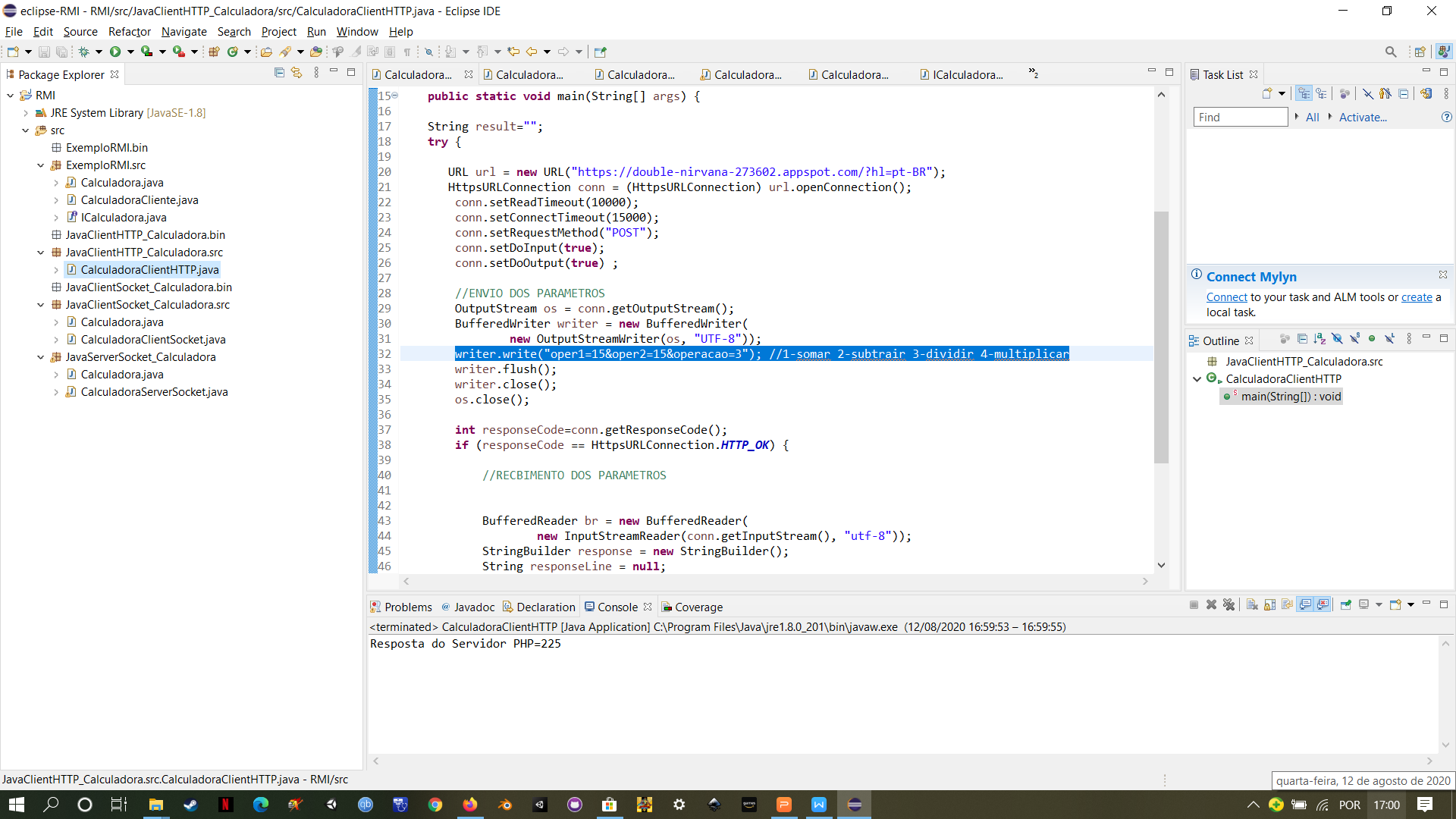


Figura 1.1

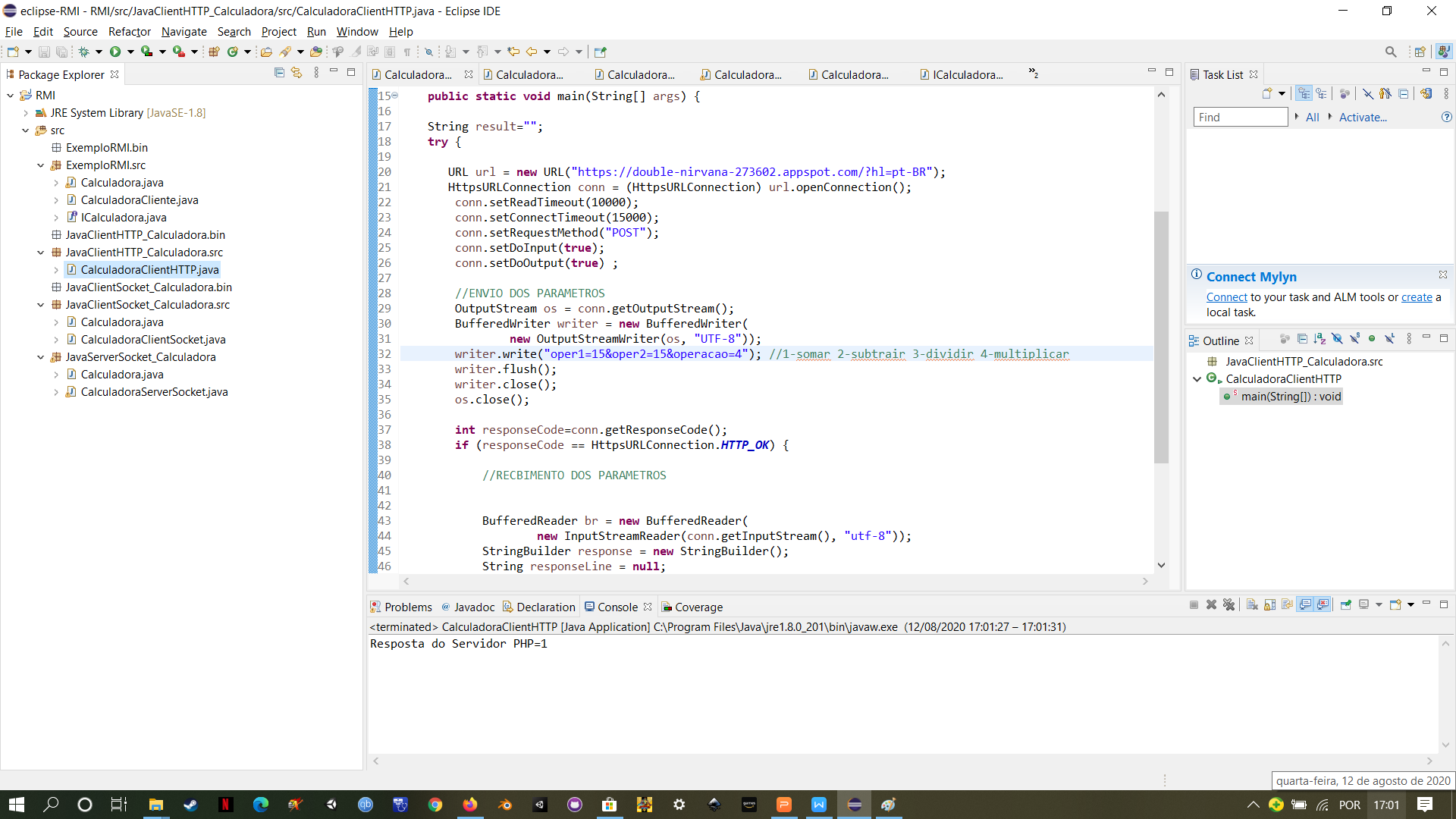


Figura 1.2

Adicionei um loop aonde o valor de “operacao” vai de 1 a 4. Assim consigo testar todos os casos. Ver Figura abaixo.

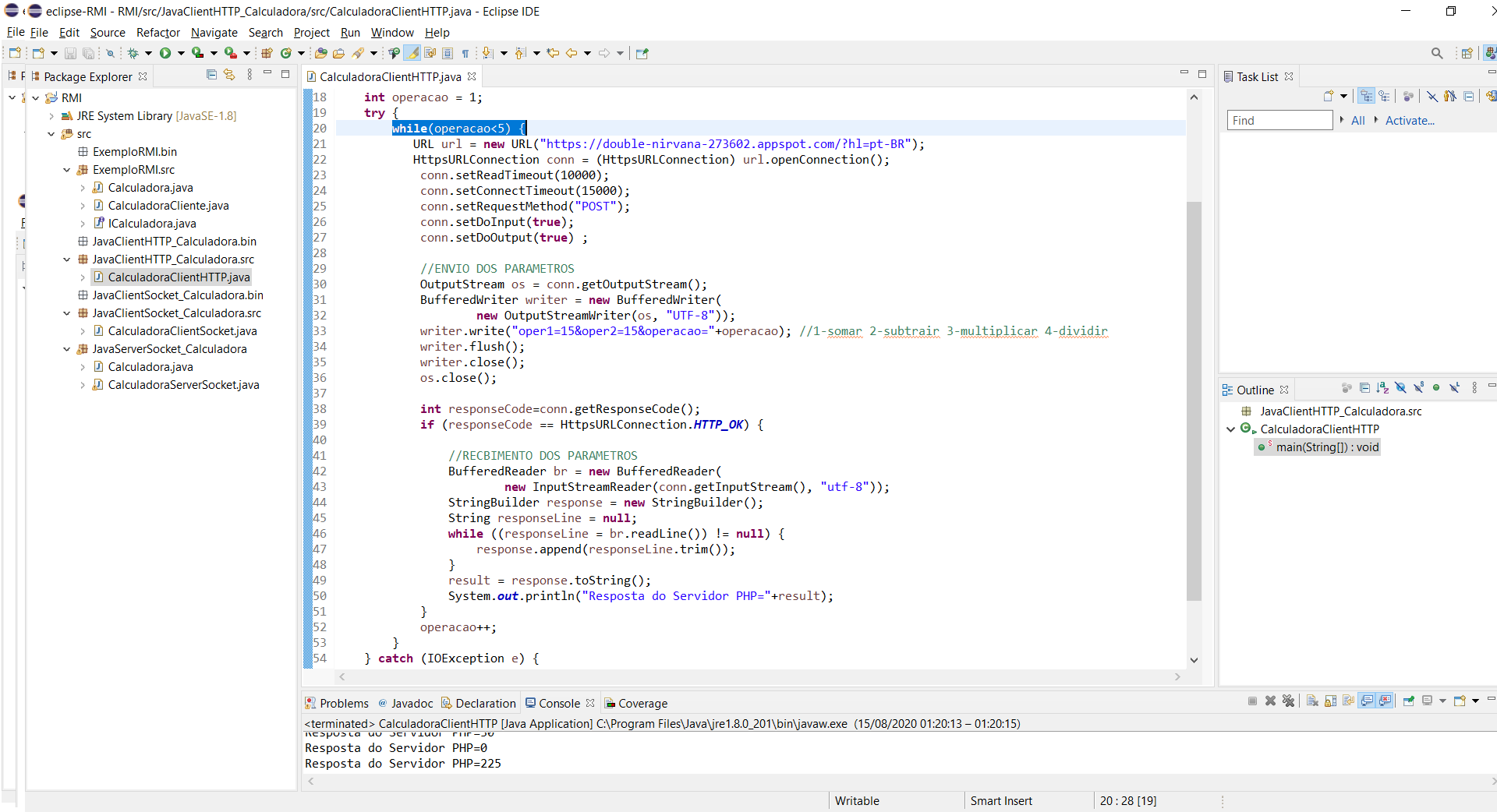


Figura 2

Exemplo de RMI

Com relação ao exemploRMI. A proposta foi evoluir a calculadora. Para isto, dentro da classe *Calculadora.java*, serão adicionadas mais 4 métodos: sub(subtração), mult(multiplicação), div(divisão) e para bônus um método exp(exponenciação).

No classe *ICalculadora.java*, que serve de interface remota para que o cliente identifique quais métodos estão disponíveis na classe *Calculadora.java*.

Dentro da classe Calculadora.java, está a figura 3, com a inserção dos contadores e métodos extras.

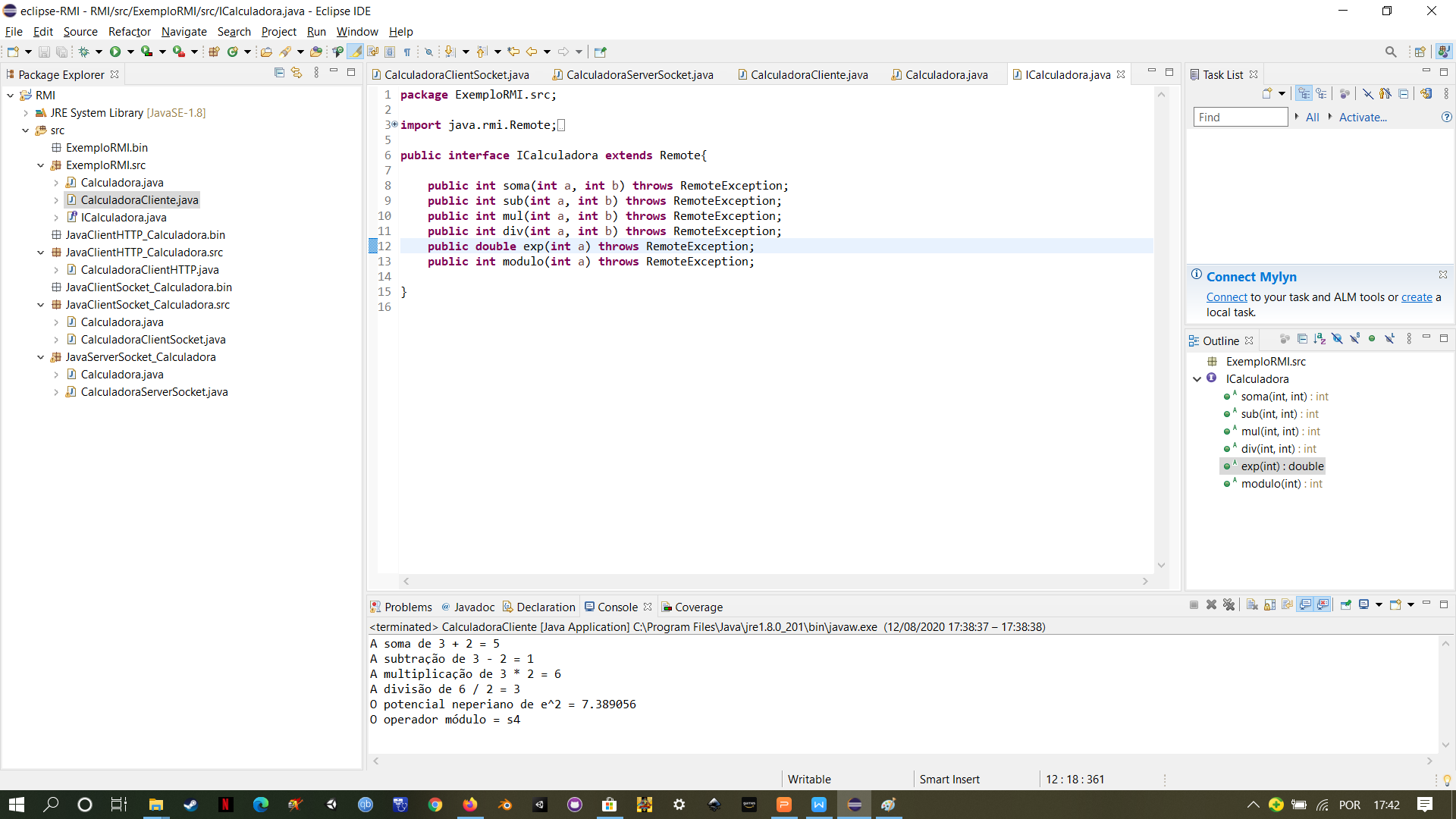


Figura 3

Para a classe ICalculadora.java, a adição das assinaturas que estarão disponíveis do Calculadora.java. Ver figura 4.

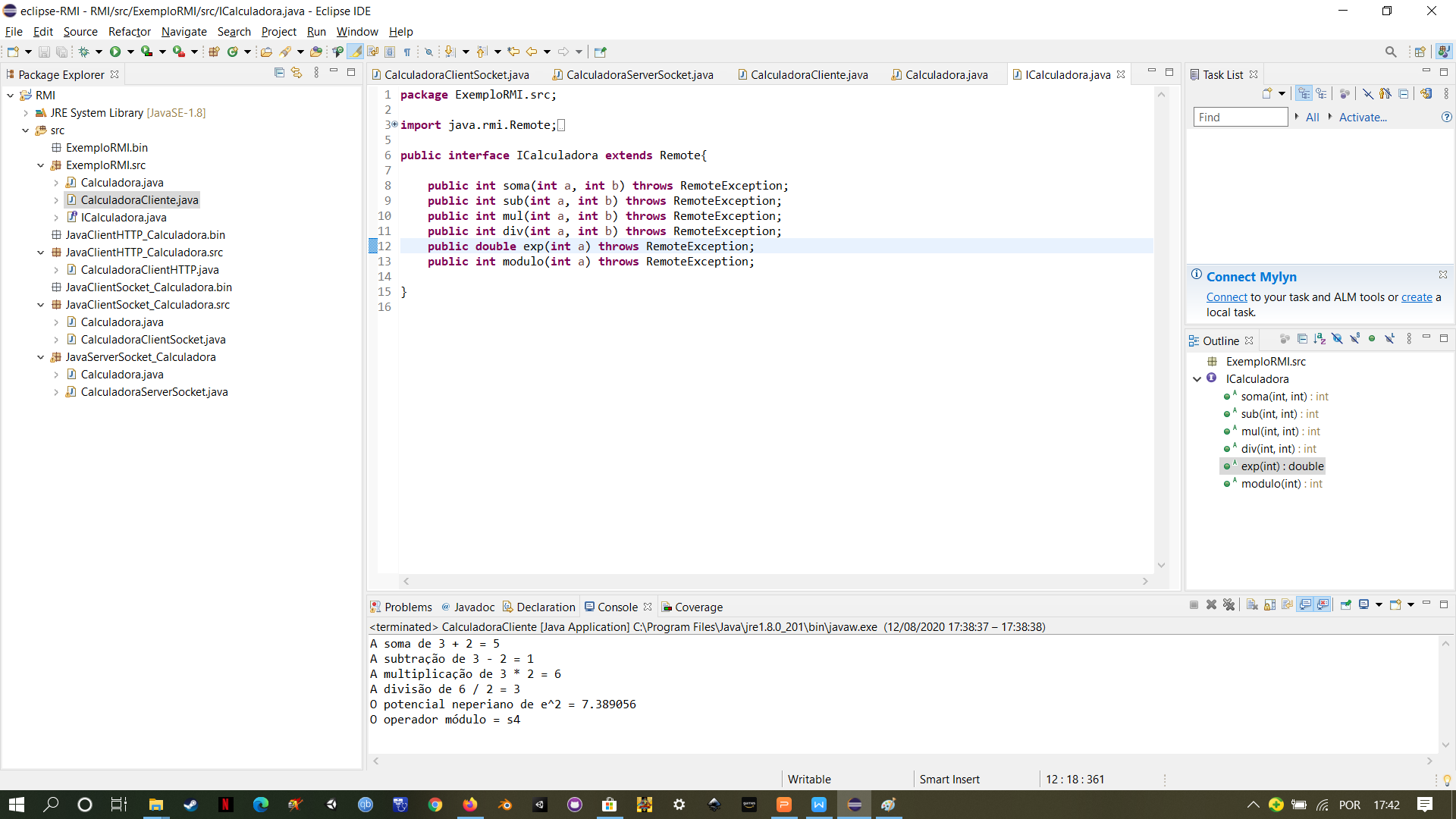


Figura 4

O cliente apenas adicionado linhas de testes para todas as funcionalidades. Ver Figura 5.

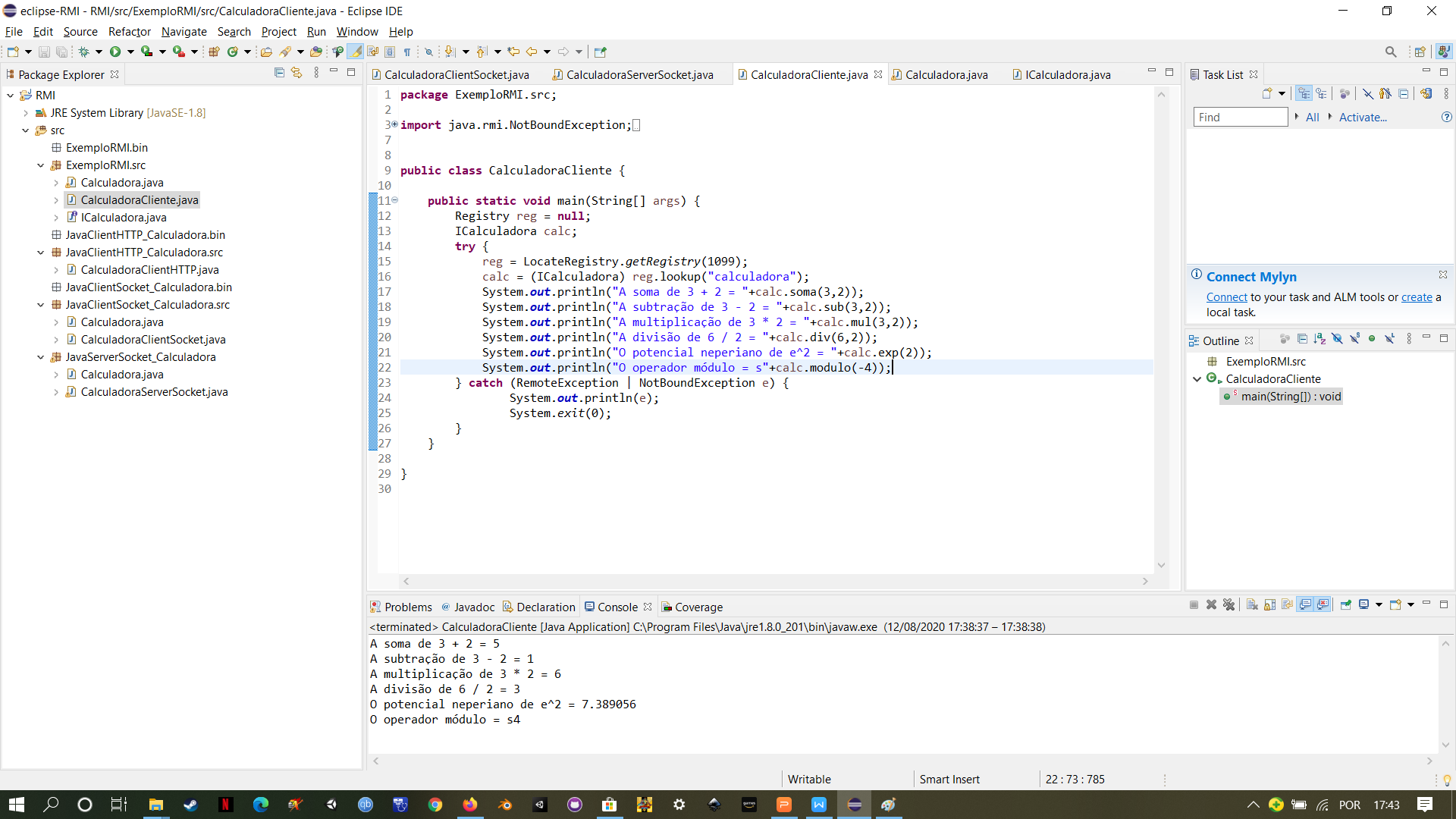


Figura 5

Exemplo de Socket

Com relação ao exemploRMI. A proposta foi evoluir a calculadora. Para isto, dentro da classe *Calculadora.java*, serão adicionadas mais 4 métodos: sub(subtração), mult(multiplicação) e div(divisão).

Primeiro ponto na infra foi remover a classe Calculadora.java do package de cliente, já que ele se comunica via Stream dentro do socket.

No *CalculadoraClientSocket.Java*, inseri um loop(**while**) no qual incrementa operacao de 1 a 4, assim com os mesmos operadores consigo testar todos as funcionalidades da calculadora.

O meu endereço IP é setado manualmente, mas ao rodar o servidor pela primeira vez, no log ele já mostra o endereço que está pendurado o server. Segue Figura 6.

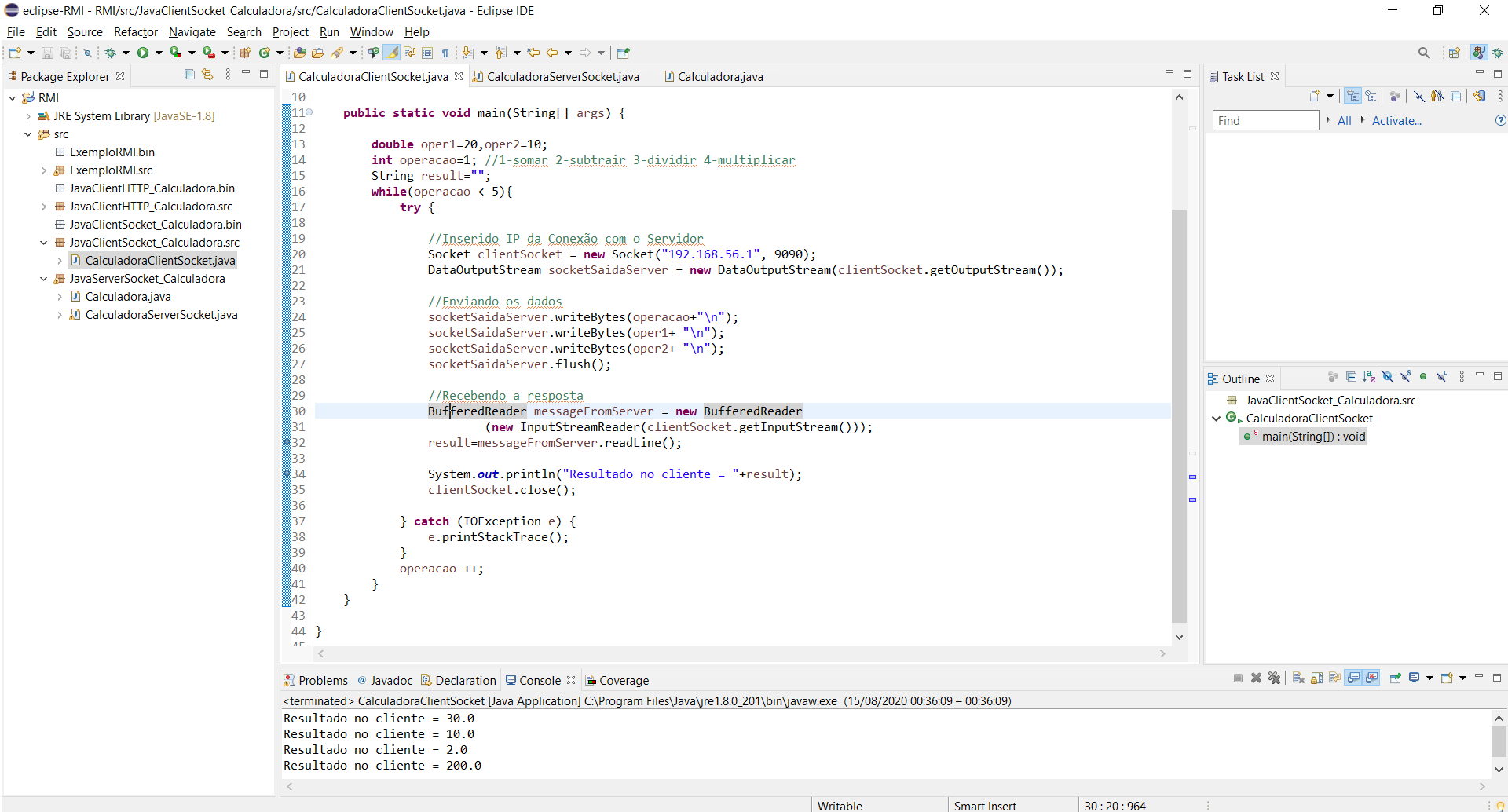


Figura 6

No *JavaServerSocket\_Calculadora.Java*, há a necessidade de inicializar o ServerSocket, Ao rodar pela primeira vez, estava inicializando no IP 0.0.0.0. Então mudei método para ServerSocket(PORTA,escuta de backlog,InetAddress), ver Figura 7. Com ele, através do InetAddress, consigo pegar o valor do IP Local e assim instanciar o ServerSocket sem problemas.

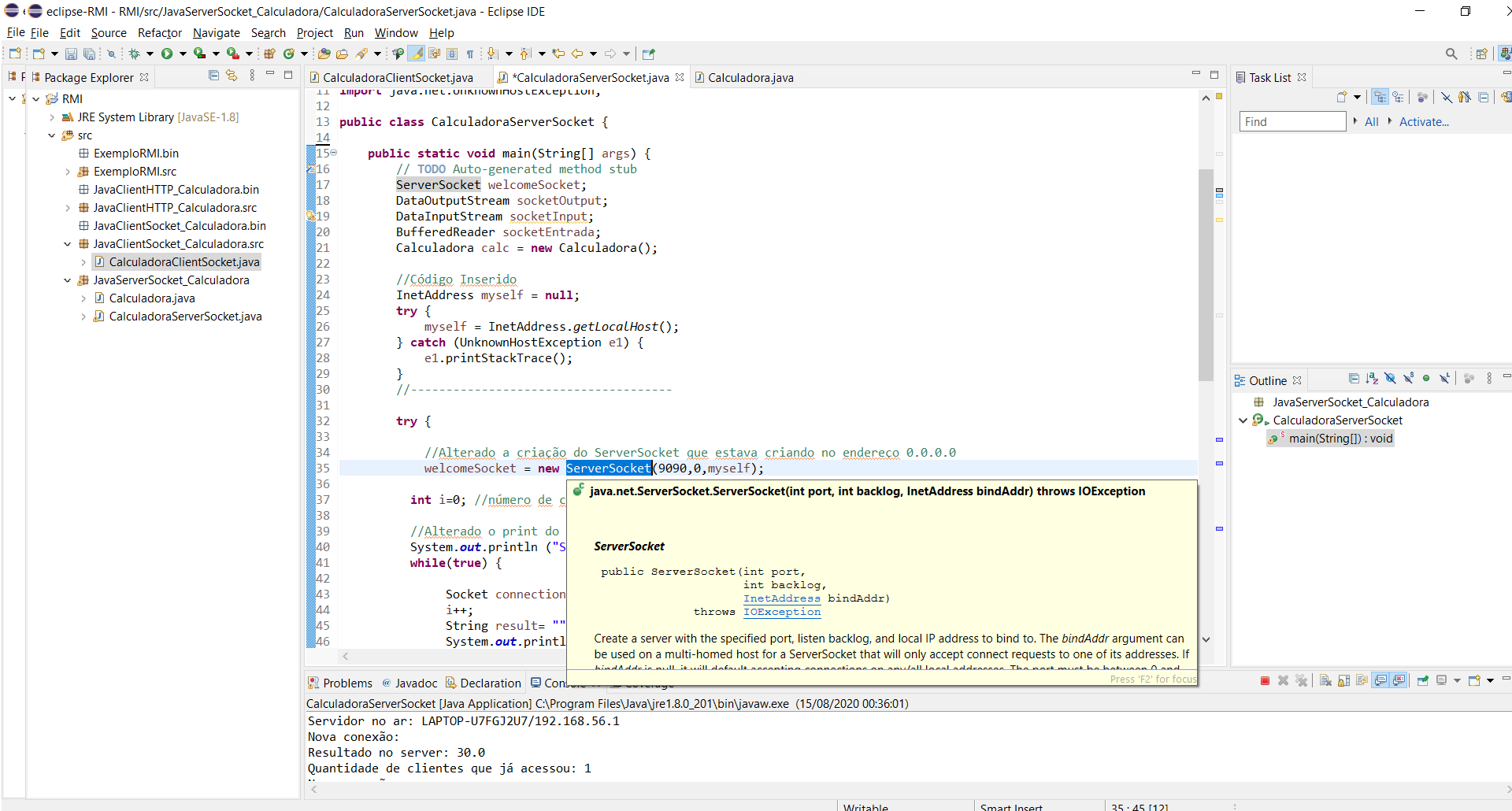


Figura 7

Seguindo evolução do projeto, a variável i, se tornará a quantidade de pessoas que acessaram o server. Alterei o log “System.***out***.println ("Servidor no ar: "+welcomeSocket.getInetAddress().toString());” para exibir o IP Local que está o servidor.

A linha de instrução, “Socket connectionSocket = welcomeSocket.accept();”, ficará esperando uma conexão, só a partir daí, a linha “i++” irá incrementar e os buffers serão lidos e salvos em variáveis.Ver FIgura 8.

Inseri uma estrutura condicional **if**, nela comparo os valores da variável operação. De acordo com a escolha o método é chamado do objeto “calc”. Ver Figura 8.

Foi adicionado alguns logs no console para controle como por exemplo: “(System.***out***.println ("Resultado no server: "+result);” e “System.***out***.println ("Quantidade de clientes que já acessou: "+i);”, ver figura 8.

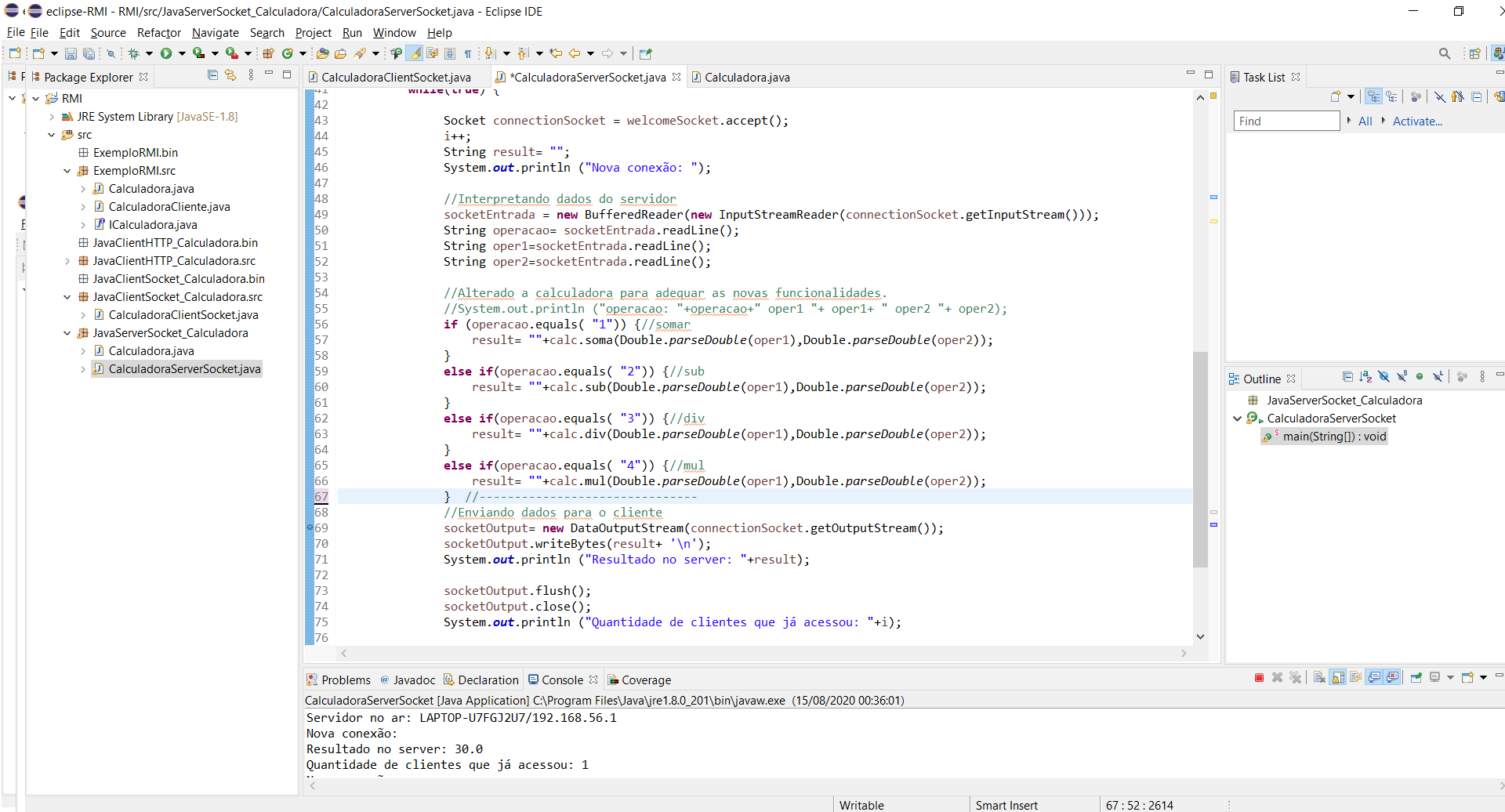


Figura 8