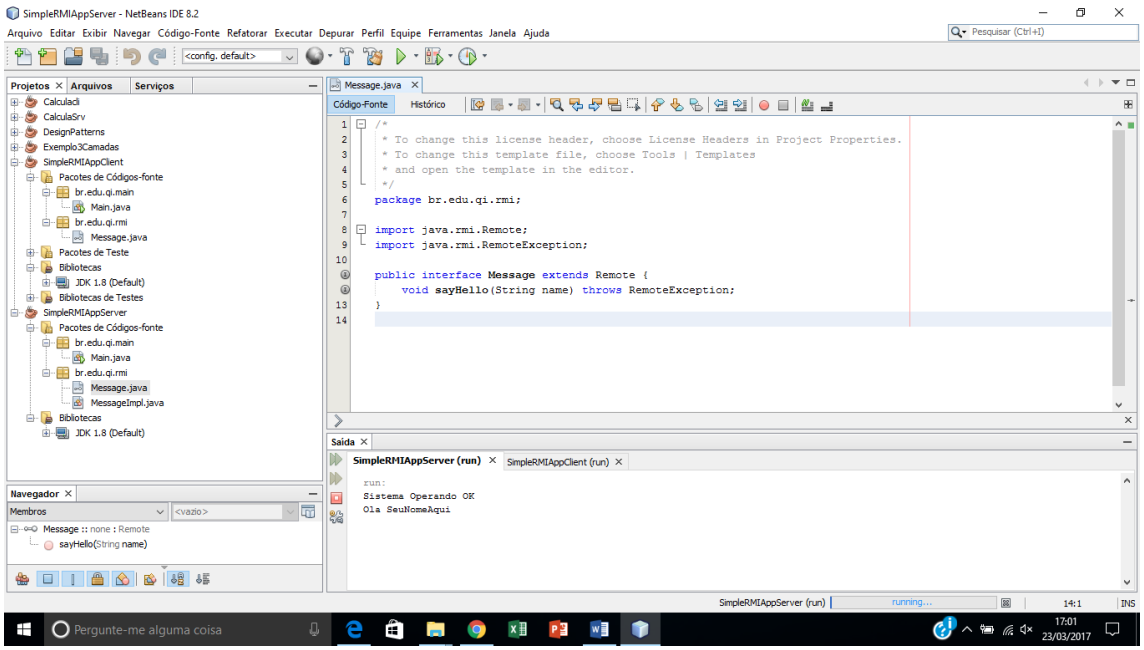
PROJETO EXEMPLO

O Java Remote Method Invocation (Java RMI) permite que o programador crie aplicativos baseados em tecnologia Java baseada em tecnologia Java, nos quais os métodos de objetos Java remotos podem ser chamados de outras máquinas virtuais Java, possivelmente em hosts diferentes. O RMI usa a serialização de objetos para empacotar e desmarcar parâmetros e não trunca tipos, suportando o verdadeiro polimorfismo orientado a objeto

Neste exemplo abaixo, estamos criando 2 projetos diferentes, um será o RMI Server e o outro será o RMI Client.

Primeiramente definimos uma interface simples, esta interface foi criada para invocar métodos e implementações. Esta interface deverá ser criada tanto na aplicação cliente como na aplicação servidor

Interface



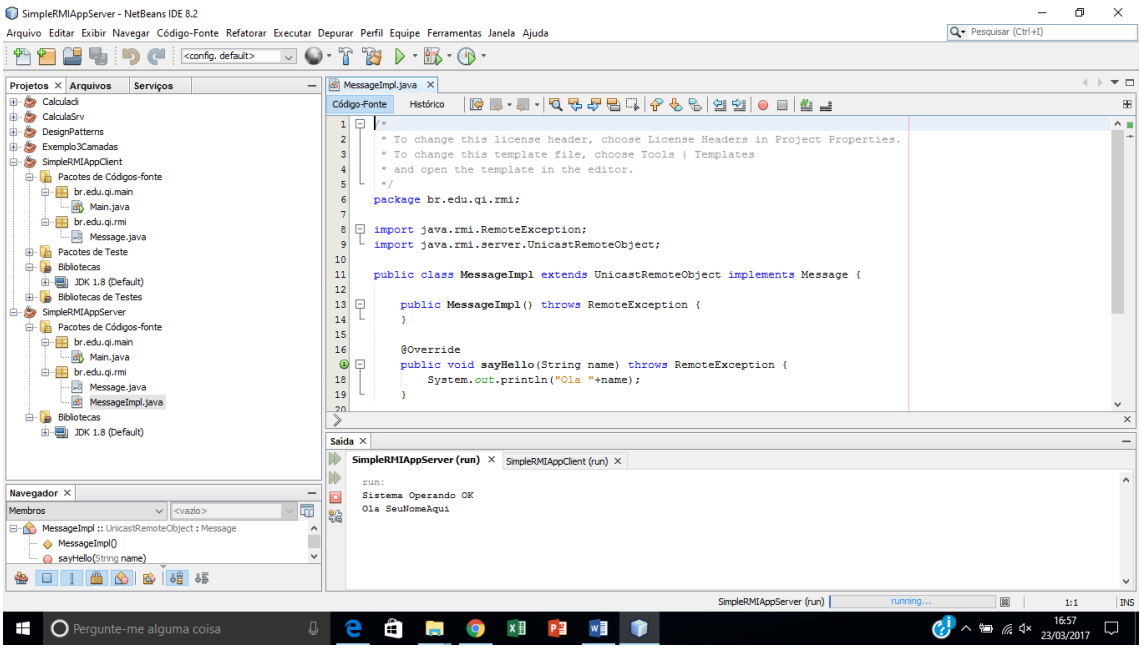
Código a implementar

```
package br.edu.qi.rmi;

import java.rmi.Remote;

import java.rmi.RemoteException;


public interface Message extends Remote {

    void sayHello(String name) throws RemoteException;
```

}

Esta é a implementação para a Interface, esta implementação existe apenas na aplicação servidor. Esta interface é responsável pela exibição da resposta do servidor.

Implementação de Resposta

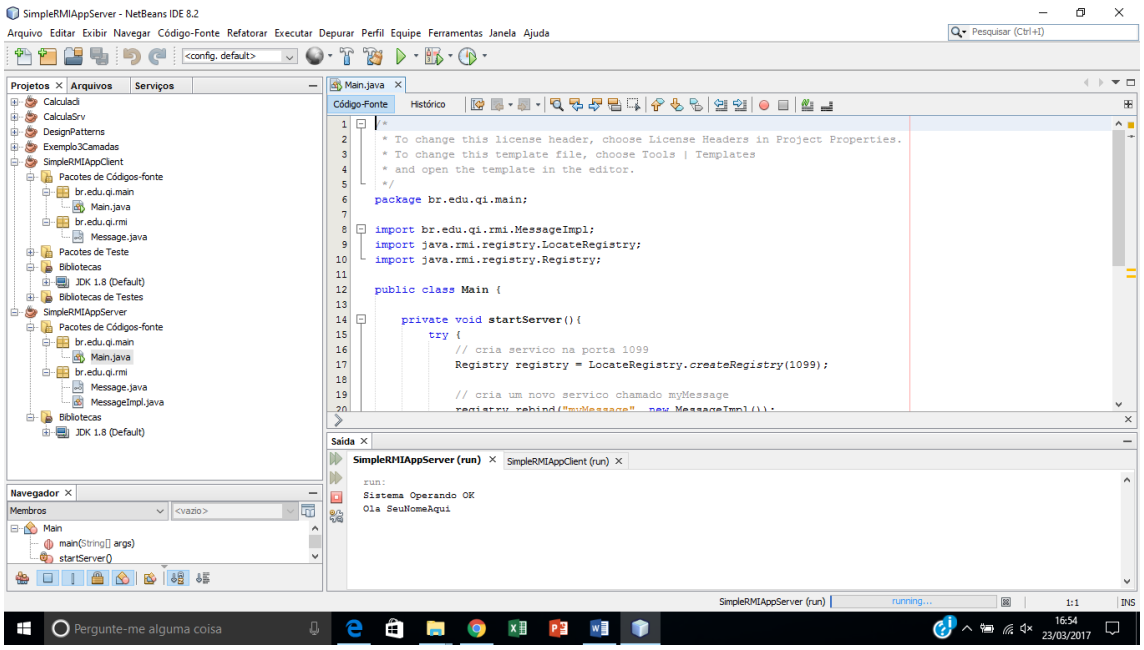


Código a implementar

```
package br.edu.qi.rmi;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;


public class MessageImpl extends UnicastRemoteObject implements Message {
    public MessageImpl() throws RemoteException {
   }
       @Override
   public void sayHello(String name) throws RemoteException {
      System.out.println("Ola "+name);
   }
```

}

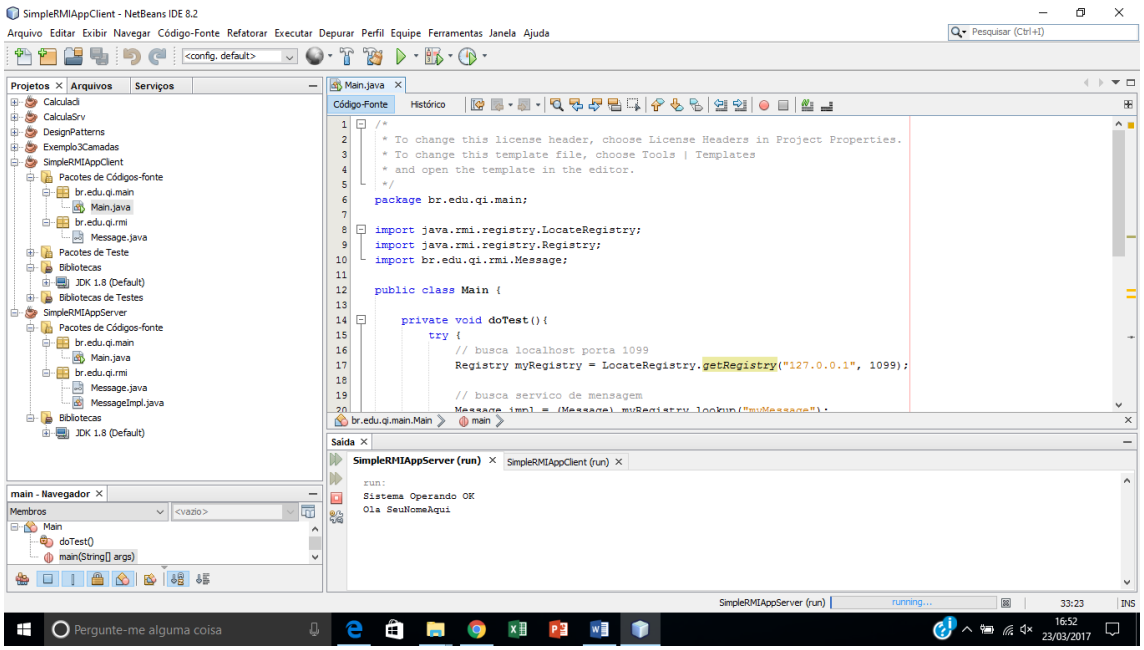
Aplicação SERVIDOR estamos usando a porta 1099




```
package br.edu.qi.main;

import br.edu.qi.rmi.MessageImpl;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;


public class Main {


   private void startServer(){

      try {

         // cria servico na porta 1099

         Registry registry = LocateRegistry.createRegistry(1099);


         // cria um novo servico chamado myMessage

         registry.rebind("myMessage", new MessageImpl());

      } catch (Exception e) {

         e.printStackTrace();
```

```
        }
        System.out.println("Sistema Operando OK");
    }

    public static void main(String[] args) {
        Main main = new Main();
        main.startServer();
    }
}
```

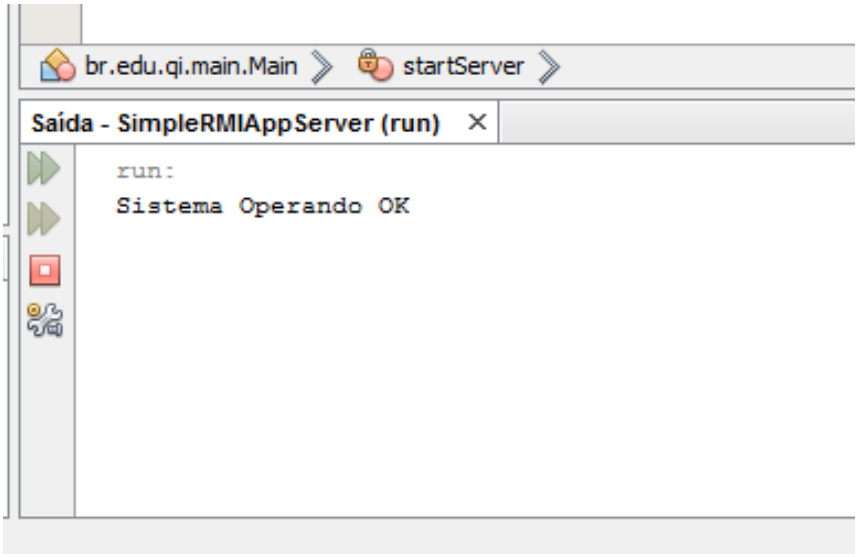
Aplicação Cliente



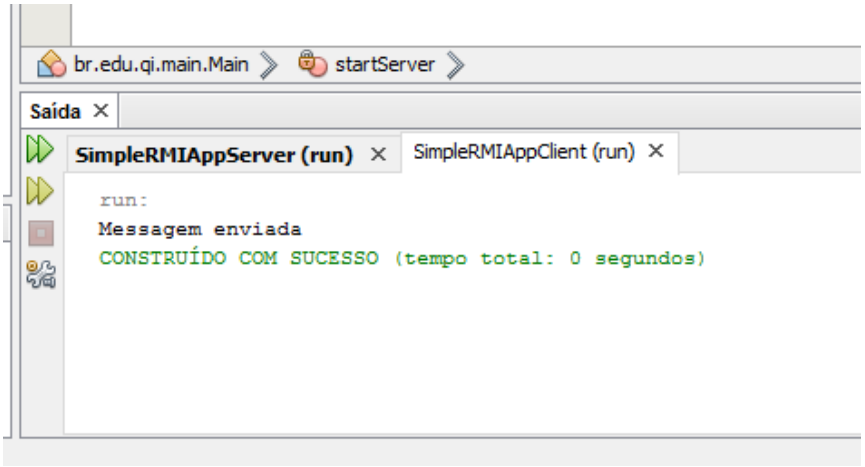

```
package br.edu.qi.main;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import br.edu.qi.rmi.Message;


public class Main {
```

```
    private void doTest(){
        try {
            // busca localhost porta 1099
            Registry myRegistry = LocateRegistry.getRegistry("127.0.0.1", 1099);

            // busca servico de mensagem
            Message impl = (Message) myRegistry.lookup("myMessage");

            // chama o metodo servidor
            impl.sayHello("SeuNomeAqui");

            System.out.println("Messagem enviada");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Main main = new Main();
        main.doTest();
    }
}
```

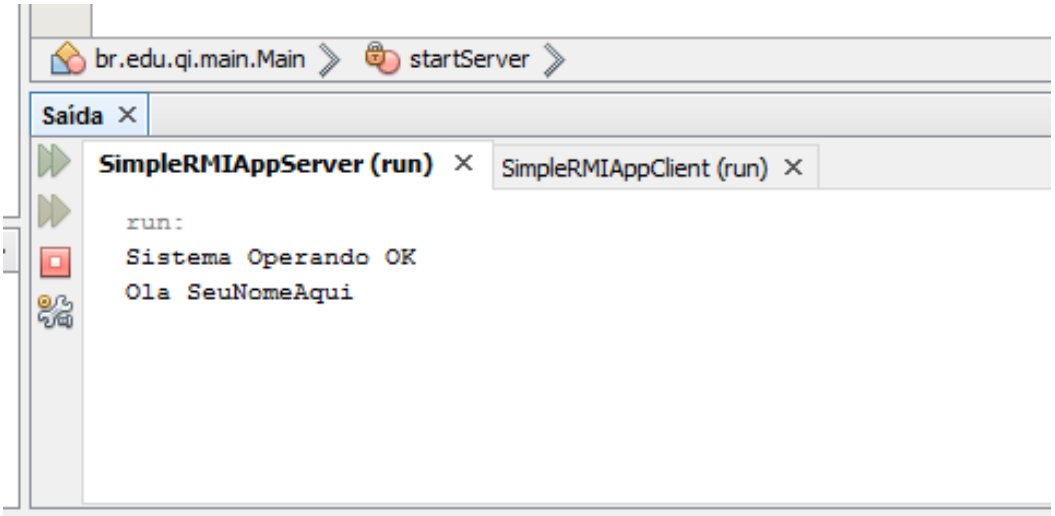
Esta imagem representa o servidor aguardando solicitações



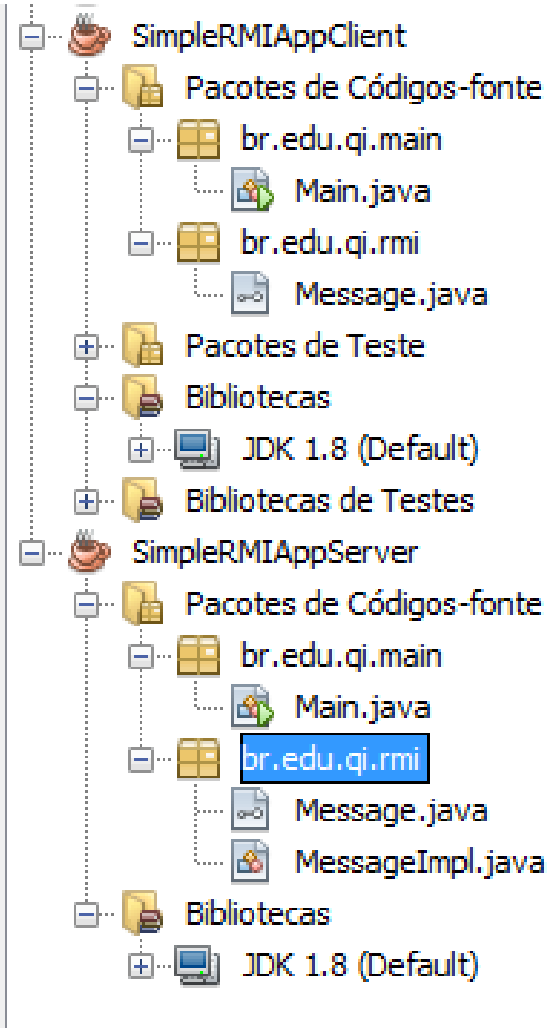
Esta Imagem representa a solicitação do cliente



Esta imagem representa a resposta do servidor



Estrutura do Projeto no NetBeans


SimpleRMIAppClient
Pacotes de Códigos-fonte
br.edu.qi.main
Main.java
br.edu.qi.rmi
Message.java
Pacotes de Teste
Bibliotecas
JDK 1.8 (Default)
Bibliotecas de Testes
SimpleRMIAppServer
Pacotes de Códigos-fonte
br.edu.qi.main
Main.java
br.edu.qi.rmi
Message.java
MessageImpl.java
Bibliotecas
JDK 1.8 (Default)