Implementação

@author Jean Carlo Massami Yamada

Aplicação desenvolvida como trabalho final da disciplina de Computação Distribuída ministrada por Dr.Prof. Milton Hirokazu Shimabukuro do curso de Ciências da Computação da
UNESP(Universidade Estadual Paulista) "Júlio Mesquita Filho" - Câmpus de Presidente Prudente.Com o intuito de concretizar o aprendizado da matéria.O objetivo da aplicação é ter
um servidor que irá invocar métodos remotos (RMI (Remote Method Invocation))
para atender requisições de seus clientes conectados via TCP.

No desenvolvimento foi utilizado a linguagem Java na versão 1.8.0_171, e a IDE NetBeans na versão 8.2.

Dificuldades:

Políticas de segurança:

No desenvolvimento do projeto houveram dificuldades no quesito de política de segurança, um assunto que não aprofundei durante o desenvolvimento.

Teve de ser criado um arquivo “security.policy” para o servidor ter acesso ao RMI.:

```
grant {
        permission java.net.SocketPermission "*:1099", "accept, connect";
        permission java.security.AllPermission "", "";
};
```

Obtido no seguinte link:
https://docs.oracle.com/javase/7/docs/technotes/guides/security/PolicyFiles.html

Interface remota e local:

No desenvolvimento não consegui criar o RMI em um projeto separado do Servidor, portanto foram implementados em um único projeto e então há apenas uma interface. Nas tentativas de criar dois projetos separados, o projeto do servidor informava um erro do tipo que não conseguia localizar o código base, não entendi muito bem.

Desenvolvimento:

Servidor:

No desenvolvimento do servidor foi criada duas Classes:

Server.java:

No Server foi implementado o Runnable para o servidor rodar em outra Thread e assim deixar o interface gráfica independente, podendo até ser requisitado um interrupção do servidor.

Seu método principal é o *public void start(),* onde é instanciado o servidor, e também onde se encontra loop principal que está aceitando conexões com clientes.
Exemplo abaixo onde se destaca os objetos e métodos principais.

```
/*instanciando servidor*/
serverSocket = new ServerSocket(serverPort, 50, InetAddress.getByName(ip));

/*instanciando uma nova conexão*/
Connection connection = new Connection(serverSocket.accept(), portRmi, lookup);
Thread thread = new Thread(connection);

/*iniciando troca de dados entre servidor e client em outra thread*/
 thread.start();
```

Connection.java:

É onde a comunicação entre servidor e cliente acontece.
Também foi implementado a Runnable, pois a comunicação irá rodar em outra Thread, assim tornando a funcionalidade de aceitar conexões do Servidor independente de uma comunicação com o cliente, e claro, permitir que o Servidor tenha vários clientes conectados simultaneamente.
Obs: para cada conexão existe uma referência para o objeto remoto.

Exemplo abaixo onde se destaca os objetos e métodos principais.

o Método principal é o public *void run().*

```
/*obtendo registro ativo*/
registry = LocateRegistry.getRegistry(portRmi);

/*referência do objeto remoto a partir do registro*/
calculator = (Calculator) registry.lookup(lookup);

/*recebendo requisição do cliente*/
data = in.readUTF();

result = calculator.sum(n1, n2);

/*enviando ao cliente resultado*/
out.writeUTF(data);
```

RMI:

No desenvolvimento do RMI foi criada a interface da calculadora, a implementação da calculadora e o servidor RMI.

Interface Calculator.java:

Na interface foi apenas criada a assinatura de métodos de operações matemática básica.

Implementação da interface ImplCalculator.java:

É onde foi implementado a interface Calculator.java, ou seja implementado os métodos matemáticos.

ServerRmi.java:

Aqui é onde é criado o registro em determinada porta e inserindo o objeto remoto com um nome no registro.

Exemplo abaixo onde se destaca os objetos e métodos principais.

O método *public boolean start()*, é onde é criado o registro.

```
/*criando registro*/
registry = LocateRegistry.createRegistry(port);
```

O método *public boolean addBind(String bind),* é onde adiciona um objeto a um registro já criado.

```
/*instanciando objeto*/
 Calculator stub = new ImplCalculator();

/*inserindo objeto no registro*/
registry.rebind(bind, stub);
```

Cliente:

Pode-se dizer que é o mais simples, onde foi implementado o *Client.java* que é o principal.

Client.java:

É onde o cliente se conecta com o servidor e envia requisições.

Exemplo abaixo onde se destaca os objetos e métodos principais.

O Método *public boolean connect()* é onde ocorre a requisição de uma conexão com um servidor.

```
/*requisitando conexão com um servidor*/
socket = new Socket(InetAddress.getByName(ip), port);
```

No método *public boolean start(String operation),* é onde ocorre a troca de dados entre cliente e servidor.

```
/*enviando dados*/
dataOutputStream.writeUTF(operation);

/*recebendo dados*/
String data = dataInputStream.readUTF();
```