1) Baixe e execute a aplicação RMI disponível em
http://www.cdk5.net/wp/extra-material/supplementary-material-for-chapter-5

2) Faça um serviço remoto simples que utilize RMI.

Passos para implementação

**1º Passo: Criação de uma interface remota que proporcionará a comunicação entre cliente e servidor (serviço);**

```
import java.rmi.*;
public interface InterfaceRemota extends Remote {
   metodoRemoto1 throws RemoteException;
   ...
   metodoRemotoN throws RemoteException;
}
```

**2º Passo: Criação de uma classe Servente que o lado servidor implementará de acordo com a interface remota. As classes Serventes dão "corpo" ao serviço fornecido pelo servidor;**

```
import java.rmi.*;
public class Servente extends UnicastRemoteObject implements
InterfaceRemota {

   public metodoRemoto1()throws RemoteException{
   //implementacao
   }

   //...
   public metodoRemotoN()throws RemoteException{
   //implementacao
   }
}
```

**3º Passo: Criação da classe Servidor (possui um método main()) e publicação do serviço;**

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class Servidor{
   public static void main(String args[]){
      System.setSecurityManager(new SecurityManager());
      try{
         InterfaceRemota refServente = new Servente();
         InterfaceRemota stub = (InterfaceRemota)
UnicastRemoteObject.exportObject(refServente, 0);
         Naming.rebind("Apelido_do_Servico", refServente);
         System.out.println("Servidor em execucao");
      }catch(Exception e) {
         System.out.println("ShapeList server main " + e.getMessage());
      }
   }
}
```

***4º Passo: Criação da classe Cliente que obterá a referência remota para o objeto que implementa o serviço;***

```
public class ShapeListClient{
   public static void main(String args[]){
      if(System.getSecurityManager() == null){
         System.setSecurityManager(new SecurityManager());
      } else System.out.println("Já há um gerenciador de Seg");
      InterfaceRemota refRemota = null;
      try{
         refRemota =(InterfaceRemota) Naming.lookup("end/apelido");
         System.out.println("Found server");
         refRemota.metodoRemoto1();
         //...
        refRemota.metodoRemotoN();
}
```

***5º Passo: Definição da política de segurança. Java é muito restrito no que diz respeito a comunicação, por questões de segurança, e para conectar uma classe a outra remota é necessário o uso de um arquivo policy, que diga ao JVM quais os serviços disponíveis e permitidos àquela classe, como apenas conexão ou fazer download de algum arquivo/classe.***

*Ex: sem nenhuma restrição*

```
grant{
permission java.security.AllPermission;
};
```

***6º Passo: Execute o servidor de Nomes (rmiregistry), o Servidor e o Cliente.***

Considerando que foi criado o pacote: *caseRemoto*

Compilar os fontes com Javac:
*Execute dentro do diretorio que possui os fontes (/src/caseRemoto):*
*javac -d ../../bin/ *.java*

Coloque a politica no diretorio dos .class (/bin)

Rode o serviço de nomes
*Execute dentro do diretório que possui os .class (/bin):*
*rmiregistry*

Rode o servidor
Execute dentro do diretorio que possui os .class (/bin):
*java -Djava.server.rmi.codebaseile:///caseRemoto/ -Djava.security.policy=policy caseRemoto.Servidor*

Rode o cliente
Execute dentro do diretorio que possui os .class (/bin):
*java -Djava.security.policy=policy caseRemoto.Cliente teste*

## **Data da entrega/apresentação: 22/05/2019 (Sippa)**