

Aplikacje RMI

Część 2

Budowa aplikacji rozproszonych

<http://java.sun.com/j2se/1.5.0/docs/guide/rmi/socketfactory/index.html>

I. Implementacja gniazd dziedziczących po Socket i ServerSocket oraz produkcji tych gniazd dla klienta i serwera RMI

- **ustawianie parametrów gniazd**
- **kontrola wywołań metod zdalnych**
- **kontrola połączeń z klientami**
- **szyfrowanie i kompresja danych**
- **implementacja ServerSocket i Socket**
- **implementacja RMIClientSocketFactory**
- **implementacja RMIServerSocketFactory**

1) Zdefiniowanie gniazd i związanych z nimi strumieni wejścia/wyjścia, które pozwalają na szyfrowanie i deszyfrowanie danych

- XorInputStream.java
- XorOutputStream.java
- XorServerSocket.java
- XorSocket.java

2) Wykonanie „fabryki ” gniazd typu ServerSocket i Socket

- **Implementacja XorClientSocketFactory**
- **Implementacja XorServerSocketFactory**

```
package examples.rmisocfac;
import java.io.*;
import java.net.*;
import java.rmi.server.*;
public class XorClientSocketFactory implements
    RMIClientSocketFactory, Serializable
{ private final byte pattern;
  public XorClientSocketFactory(byte pattern_)
    { pattern=pattern_; }
  public Socket createSocket(String host, int port)
    throws IOException
    { return new XorSocket(host, port, pattern); }
  public int hashCode() {return (int) pattern;}
  public boolean equals(Object obj)
    { return (getClass() == obj.getClass() &&
      pattern == ((XorClientSocketFactory) obj).pattern); }
}
```

```
package examples.rmisocfac;
import java.io.*;
import java.net.*;
import java.rmi.server.*;
public class XorServerSocketFactory implements
    RMIServerSocketFactory
{
    private byte pattern;
    public XorServerSocketFactory(byte pattern)
    {
        this.pattern = pattern;
    }
    public ServerSocket createServerSocket(int port)
        throws IOException
    {
        return new XorServerSocket(port, pattern);
    }
    public int hashCode()          { return (int) pattern; }
    public boolean equals(Object obj)
    {
        return (getClass() == obj.getClass() &&
            pattern == ((XorServerSocketFactory) obj).pattern);
    }
}
```

II. Zastosowanie gniazd w aplikacjach RMI

- Wykonanie interfejsu obiektu zdalnego oraz jego implementacji
- Wykonanie aplikacji serwera, który tworzy obiekt zdalny, którego referencję zostawia w rejestrze RMI
- Wykonanie aplikacji klienta, który po otrzymaniu referencji do obiektu zdalnego otrzymuje gniazdo do komunikacji z przydzielonym gniazdem po stronie serwera, zawierającego obiekt zdalny

Interfejs obiektu zdalnego

```
package examples.rmisocfac;
```

```
import java.util.*;
```

```
import java.rmi.*;
```

```
public interface RMI_Interfejs_Wiadomosc extends Remote  
{ public void zapiszWiadomosc(String s)  
                                throws RemoteException;  
  public Date odczytajWiadomosc(String s)  
                                throws RemoteException;  
  public String weString() throws RemoteException;  
}
```


Implementacja obiektu zdalnego

```
package examples.rmisocfac;
```

```
import java.io.*;
```

```
import java.util.*;
```

```
import java.rmi.*;
```

```
import java.rmi.server.*;
```

```
import java.rmi.registry.*;
```

```
public class RMI_Wiadomosc implements  
                    RMI_Interfejs_Wiadomosc
```

```
{
```

```
    String dane;
```

```
    Date data;
```

public void zapiszWiadomosc(String s)

throws RemoteException

```
{ System.out.println(s);  
  data = new Date();  
  System.out.println(data);  
  dane =weString();  
}
```

public Date odczytajWiadomosc(String s)

throws RemoteException

```
{ System.out.println(s);  
  System.out.println(data);  
  System.out.println(dane);  
  return new Date();  
}
```

```
public String weString() throws RemoteException
{
    InputStreamReader wejście =
        new InputStreamReader( System.in );
    BufferedReader bufor =
        new BufferedReader( wejście );
    System.out.print("Podaj wiadomosc: ");
    try
    { return bufor.readLine(); }
    catch (IOException e)
    { System.err.println("Blad IO String");
      return ""; }
}
}
```

Aplikacja serwera

```
package examples.rmisocfac;  
import java.io.*;  
import java.rmi.*;  
import java.rmi.server.*;  
import java.rmi.registry.*;  
  
public class RMI_Server  
{ public static void main(String[] args)  
  { System.setSecurityManager(new RMI_SecurityManager());  
  try  
  { byte pattern = (byte) 0xAC;  
    RMI_ClientSocketFactory RMICsf=  
      new Xor_ClientSocketFactory(pattern);  
    RMI_ServerSocketFactory RMISsf =  
      new Xor_ServerSocketFactory(pattern);  
    RMI_Wiadomosc wiadomosc = new RMI_Wiadomosc();
```

```
RMI_Interfejs_Wiadomosc stub =  
    (RMI_Interfejs_Wiadomosc)  
        UnicastRemoteObject.exportObject(  
            wiadomosc, 0, RMICsf, RMISsf);  
/Utworzenie rejestru i zrejstrowanie referencji do zdalnego obiektu.  
    LocateRegistry.createRegistry(5002);  
    Registry registry = LocateRegistry.getRegistry(5002);  
  
    registry.rebind("RMI_Wiadomosc", stub);  
    System.out.println("Sewer przygotowany do RMI");  
} catch (Exception e)  
    { System.out.println("Blad serwera RMI"+e.getMessage());  
      e.printStackTrace();  
    }  
}  
}
```

Aplikacja klienta

```
package examples.rmisocfac;
```

```
import java.rmi.*;
```

```
import java.rmi.registry.*;
```

```
import java.io.*;
```

```
import java.util.*;
```

```
public class RMI_Klient implements Runnable
```

```
{ RMI_Interfejs_Wiadomosc wiadomosc;
```

```
void Zapiszobiektydopliku(String s) throws RemoteException
{System.out.println(s);
try
{ FileOutputStream plikobiekto =
    new FileOutputStream ("Wiadomosc.obj");
  ObjectOutputStream strumienobiekto =
    new ObjectOutputStream (plikobiekto);
  strumienobiekto.writeObject(wiadomosc);
  strumienobiekto.close();
  System.out.println(
    "\nObiekt wiadomosc zostal zapisany do pliku");
} catch (Exception e)
  { System.out.println (
    "Blad zapisu pliku obiektowego"+e); }
}
```

```
void Odczytajobjektyzpliku(String s)throws RemoteException
{
    try
    { FileInputStream plikobiekto =
        new FileInputStream ("Wiadomosc.obj");
      ObjectInputStream strumienobiekto =
        new ObjectInputStream (plikobiekto);
      RMI_Interfejs_Wiadomosc wiadomosc_nowa =
        (RMI_Interfejs_Wiadomosc)strumienobiekto.readObject();
      System.out.println(
        "\nObiekt wiadomosc zostal odczytany z pliku");
      if (wiadomosc_nowa!=null)
        wiadomosc_nowa.odczytajWiadomosc(s);
      strumienobiekto.close();
    } catch (Exception e)
    { System.out.println ("Blad odczytu pliku obiektowego"+e);}
}
```



```
public void run()
{ System.setSecurityManager(new RMISecurityManager());
try
{ Registry registry = LocateRegistry.getRegistry(5002);
  wiadomosc = (RMI_Interfejs_Wiadomosc)
    registry.lookup("RMI_Wiadomosc");
  System.out.println("Klient przygotowany do RMI");
  wiadomosc.zapiszWiadomosc(
    "\nKlient uzywa metody zapiszWiadomosc obiektu z serwera za pomoca RMI");
  Date data = wiadomosc.odczytajWiadomosc(
    "\nKlient uzywa metody odczytajWiadomosc obiektu z serwera za pomoca RMI" );
  System.out.println("Data otrzymana z serwera: "+data);
  Zapiszobiektydopliku(
    "\nKlient zapisuje do pliku obiekt z serwera dostepny za pomoca RMI");
  Odczytajobjektyzpliku(
    "\nTo zawartosc obiektu odczytana przez klienta z pliku za pomoca serializacji");
}
}
```

catch (Exception e)

```
{ System.out.println("Blad klienta RMI: "  
                    +e.getMessage());  
  e.printStackTrace();  
}
```

public static void main(String []args)

```
{ RMI_Klient klient = new RMI_Klient();  
  Thread t= new Thread(klient);  
  t.start();  
}
```

III. Uruchomienie aplikacji RMI

- Kompilacja obiektu zdalnego, aplikacji serwera i klienta oraz systemu „fabryki gniazd”
- Kompilacja i utworzenie architektury RMI dla obiektu zdalnego za pomocą kompilatora rmic
- Przygotowanie systemu bezpieczeństwa `java.policy` do obsługi RMI
- Uruchomienie serwera z określeniem uprawnień za pomocą pliku `policy.txt`
- Uruchomienie aplikacji klienta z określeniem uprawnień za pomocą pliku `policy.txt`

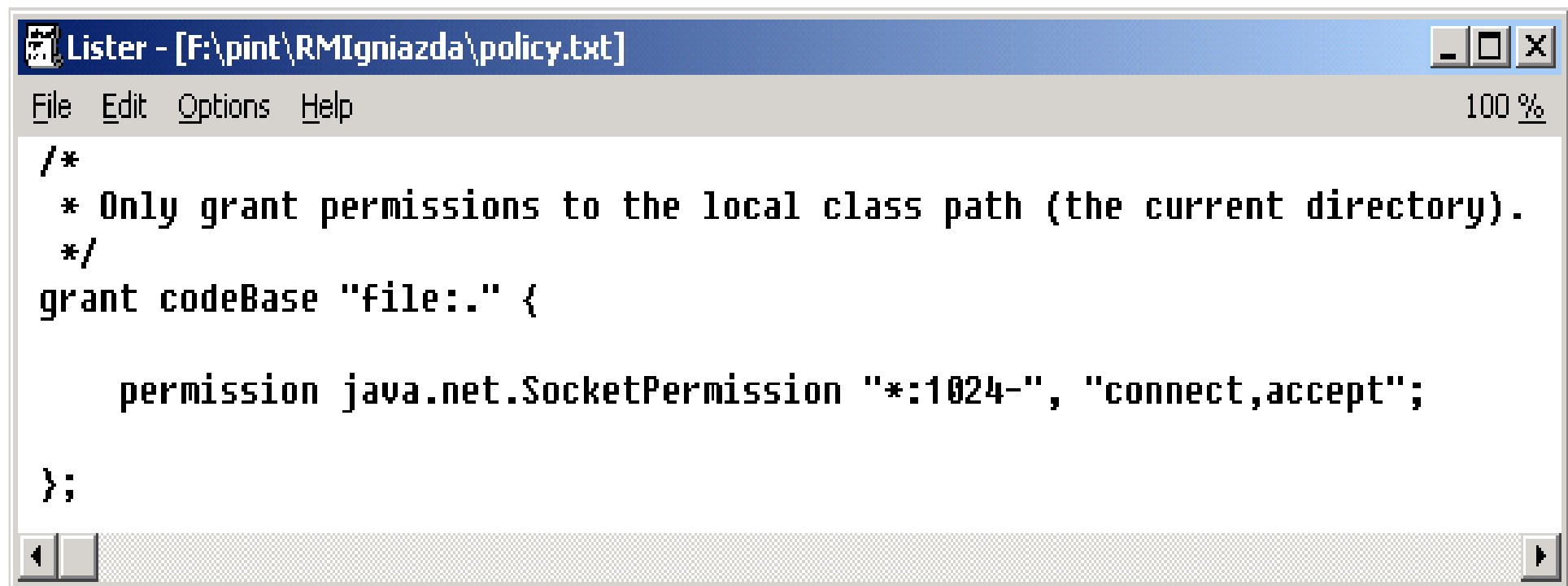
Pliki wsadowe do uruchamiania poszczególnych części systemu RMI

```
Lister - [F:\pint\RMIgniazda\rmi\RMI3\RMI___2_1.bat]
File Edit Options Help 100 %
set classpath=
d:\jdk1.5\bin\rmic examples.rmisocfac.RMI_Wiadomosc
pause
```

```
Lister - [F:\pint\RMIgniazda\rmi\RMI3\RMI___2_2.bat]
File Edit Options Help 100 %
set classpath=
d:\jdk1.5\bin\java -Djava.security.policy=policy.txt
examples.rmisocfac.RMI_Server
pause
```

```
Lister - [F:\pint\RMIgniazda\rmi\RMI3\RMI___2_3.bat]
File Edit Options Help 100 %
set classpath=
d:\jdk1.5\bin\java -Djava.security.policy=policy.txt
examples.rmisocfac.RMI_Klient
pause
```

Określenie warunków bezpieczeństwa działania aplikacji RMI



A screenshot of a Notepad window titled "Lister - [F:\pint\RMigniazda\policy.txt]". The window has a menu bar with "File", "Edit", "Options", and "Help", and a status bar showing "100 %". The text content is as follows:

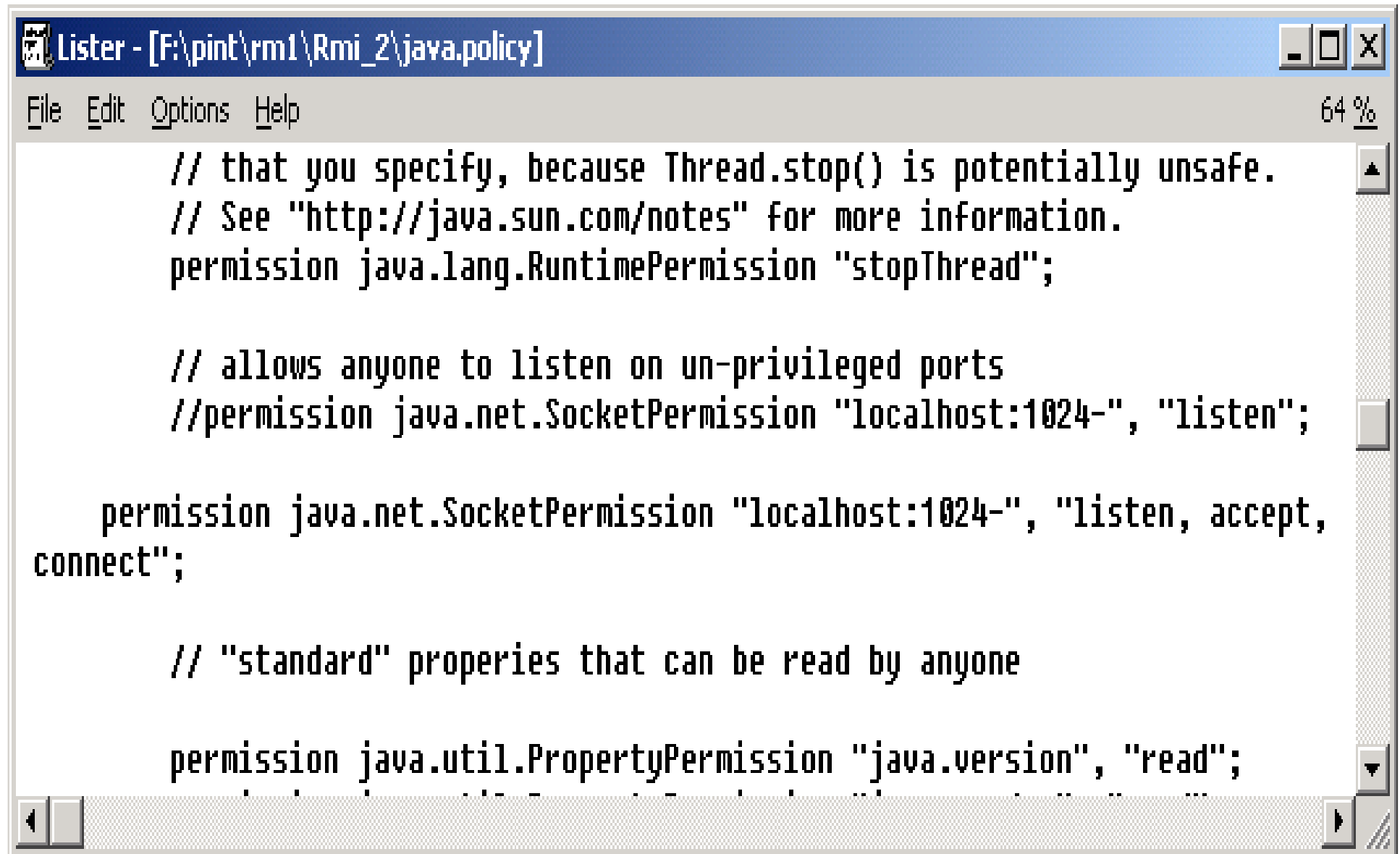
```
/*
 * Only grant permissions to the local class path (the current directory).
 */
grant codeBase "file:." {
    permission java.net.SocketPermission "*:1024-", "connect,accept";
};
```



A screenshot of a Notepad window titled "Lister - [F:\pint\RMigniazda\rmi\RMi3\policy.txt]". The window has a menu bar with "File", "Edit", "Options", and "Help", and a status bar showing "100 %". The text content is as follows:

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

Przygotowanie systemowego pliku java.policy



The image shows a Notepad window titled "Lister - [F:\pint\rm1\Rmi_2\java.policy]". The window contains the following text:

```
File Edit Options Help 64 %  
  
// that you specify, because Thread.stop() is potentially unsafe.  
// See "http://java.sun.com/notes" for more information.  
permission java.lang.RuntimePermission "stopThread";  
  
// allows anyone to listen on un-privileged ports  
//permission java.net.SocketPermission "localhost:1024-", "listen";  
  
permission java.net.SocketPermission "localhost:1024-", "listen, accept,  
connect";  
  
// "standard" properties that can be read by anyone  
  
permission java.util.PropertyPermission "java.version", "read";
```

C:\WINNT\System32\cmd.exe

F:\pint\RMIgniazda\rmi\RMI3>set classpath=

F:\pint\RMIgniazda\rmi\RMI3>d:\jdk1.5\bin\rmic examples.rmisocfac.RMI_Wiadomosc

F:\pint\RMIgniazda\rmi\RMI3>pause

Naciśnij dowolny klawisz, aby kontynuować . . .

C:\WINNT\System32\cmd.exe

F:\pint\RMIgniazda\rmi\RMI3>set classpath=

F:\pint\RMIgniazda\rmi\RMI3>d:\jdk1.5\bin\java -Djava.security.policy=policy.txt
examples.rmisocfac.RMI_Klient

Klient przygotowany do RMI

Data otrzymana z serwera: Thu Jun 01 16:24:45 CEST 2006

Klient zapisuje do pliku obiekt z serwera dostepny za pomoca RMI

Obiekt wiadomosc zostal zapisany do pliku

Obiekt wiadomosc zostal odczytany z pliku

F:\pint\RMIgniazda\rmi\RMI3>pause

Naciśnij dowolny klawisz, aby kontynuować . . .

C:\WINNT\System32\cmd.exe

F:\pint\RMIgniazda\rmi\RMI3>set classpath=

F:\pint\RMIgniazda\rmi\RMI3>d:\jdk1.5\bin\java -Djava.security.policy=policy.txt
examples.rmisocfac.RMI_Klient

Klient przygotowany do RMI

Data otrzymana z serwera: Thu Jun 01 16:24:44 CEST 2006

Klient zapisuje do pliku obiekt z serwera dostepny za pomoca RMI

Obiekt wiadomosc zostal zapisany do pliku

Obiekt wiadomosc zostal odczytany z pliku

F:\pint\RMIgniazda\rmi\RMI3>pause

Naciśnij dowolny klawisz, aby kontynuować . . .

C:\WINNT\System32\cmd.exe

F:\pint\RMIgniazda\rmi\RMI3>set classpath=

F:\pint\RMIgniazda\rmi\RMI3>d:\jdk1.5\bin\java -Djava.security.policy=policy.txt
examples.rmisocfac.RMI_Server

Sewer przygotowany do RMI

Klient uzywa metody zapiszWiadomosc obiektu z serwera za pomoca RMI

Thu Jun 01 16:23:08 CEST 2006

Podaj wiadomosc: ala

Klient uzywa metody zapiszWiadomosc obiektu z serwera za pomoca RMI

Thu Jun 01 16:24:34 CEST 2006

Podaj wiadomosc: ola

Klient uzywa metody odczytajWiadomosc obiektu z serwera za pomoca RMI

Thu Jun 01 16:24:34 CEST 2006

ola

Klient uzywa metody odczytajWiadomosc obiektu z serwera za pomoca RMI

Thu Jun 01 16:24:34 CEST 2006

ola

To zawartosc obiektu odczytana przez klienta z pliku za pomoca serializacji

Thu Jun 01 16:24:34 CEST 2006

ola

To zawartosc obiektu odczytana przez klienta z pliku za pomoca serializacji

Thu Jun 01 16:24:34 CEST 2006

ola