

# Rozproszone systemy internetowe

## JAVA Remote Method Invocation (RMI)

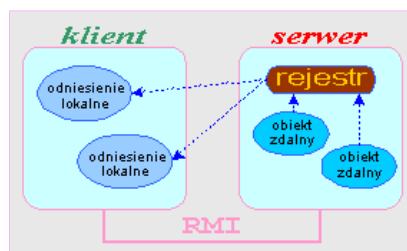
[Help:](#)

[RMI w praktyce](#)

[RMI- step by step \(Eclipse\)](#)

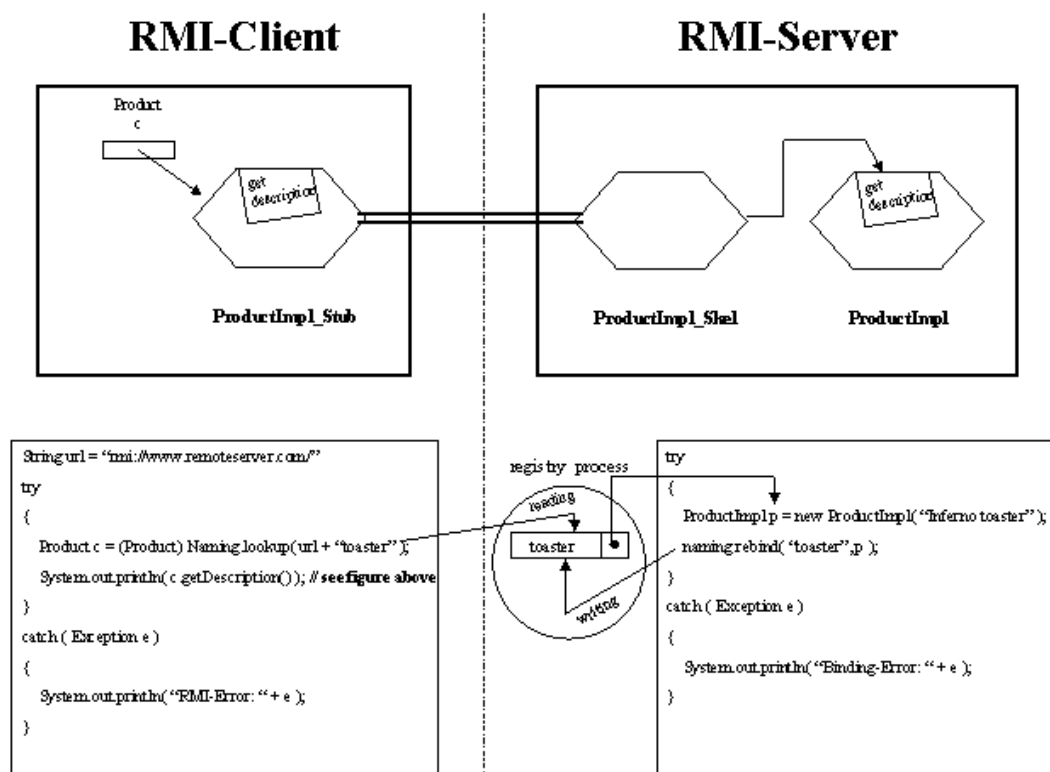
[Getting Started Using Java RMI](#)

**RMI** (ang. **Remote Method Invocation** - zdalne wywołanie metod) to mechanizm umożliwiający zdalne wywołanie metod obiektów. Obiekty te mogą znajdować się w innych maszynach wirtualnych Javy, które mogą znajdować się na innych komputerach.



### Zasada działania:

Obiekty zdalne rejestrowane są pod wybranymi nazwami w serwisie **RMI Registry**. **Aplikacja kliencka** ściąga z RMI Registry tzw. **stub** tego obiektu, który umożliwia komunikację z obiektem zdalnym przy użyciu wyeksportowanych metod w ten sam sposób, jakby chodziło o obiekt lokalny. Rola RMI Registry w tym miejscu kończy się - nie pośredniczy on w komunikacji pomiędzy aplikacją kliencką a obiektem zdalnym. Parametry metod będące obiektami przy wywołaniu zdalnym są serializowane.



# Ćwiczenie 1. Aplikacja echo

Uruchom aplikację Echo w Java RMI (Serwer i klient na tym samym PC)

## 1. Interfejs

Pierwszym krokiem w budowie aplikacji RMI jest konstrukcja **interfejsu** zawierającego metody, które serwer będzie implementował, a klient wywoływał. Interfejs ten będzie później dostępny zarówno dla programu klienta jak i serwera.

Interfejs RMI powinien :

- powinien rozszerzać standardowy interfejs **Remote**
- każda z metod powinna deklarować wystąpienie wyjątku **RemoteException**
- interfejs powinien być **publiczny**

```
package jg.pb.rsi.rmi;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface MyServerInt extends Remote{

    String getDescription(String text) throws RemoteException;

}
```

## 2. Klasa serwisu i serwer

### Implementacja interfejsu

Klasa implementująca interfejs na serwerze powinna dodatkowo dziedziczyć po standardowej klasie **UnicastRemoteObject**. Ze względu na to dziedziczenie powinna też implementować **publiczny konstruktor** deklarujący rzucenie wyjątku **RemoteException**.

### Klasa serwisu:

- implementuje odległy interfejs

```
package jg.pb.rsi.rmi;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class MyServerImpl extends UnicastRemoteObject implements MyServerInt {

    int i = 0;

    protected MyServerImpl() throws RemoteException {

        super();

    }

    @Override
```

```

public String getDescription(String text) throws RemoteException {

    i++;

    System.out.println("MyServerImpl.getDescription: " + text + " " + i);

    return "getDescription: " + text + " " + i;

}}

```

#### **Klasa serwisu:**

- instaluje menadżera bezpieczeństwa obsługującego RMI
- tworzy egzemplarz odległego obiektu i rejestruje go w serwisie RMI

Klasa rejestrująca server w rejestrze RMI (rmiregistry)

```

package jg.pb.rsi.rmi;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

public class MyServerMain

public static void main(String[] args) {

    try {

        System.setProperty("java.security.policy", "security.policy");

        if (System.getSecurityManager() == null) {

            System.setSecurityManager(new SecurityManager());

        }

        //System.setProperty("java.rmi.server.codebase", "file:/C:/Users/Jacek/workspace/
        RMIServer/bin/");

        System.setProperty("java.rmi.server.codebase", "file:/C:/Users/Jacek/NetBeansProjects/
        RMIServer/build/classes/");

        //System.setProperty("java.rmi.server.codebase", "http://192.168.1.102/jaco/");

        System.out.println("Codebase: " + System.getProperty("java.rmi.server.codebase"));

        // LocateRegistry.createRegistry(1099);

        MyServerImpl obj1 = new MyServerImpl();
    }
}

```

```
Naming.rebind("//localhost/ABC", obj1);

System.out.println("Serwer oczekuje ...");

} catch (RemoteException | MalformedURLException e) {

e.printStackTrace();

}}}
```

// Port 1099 jest portem domyślnym dla rmi.

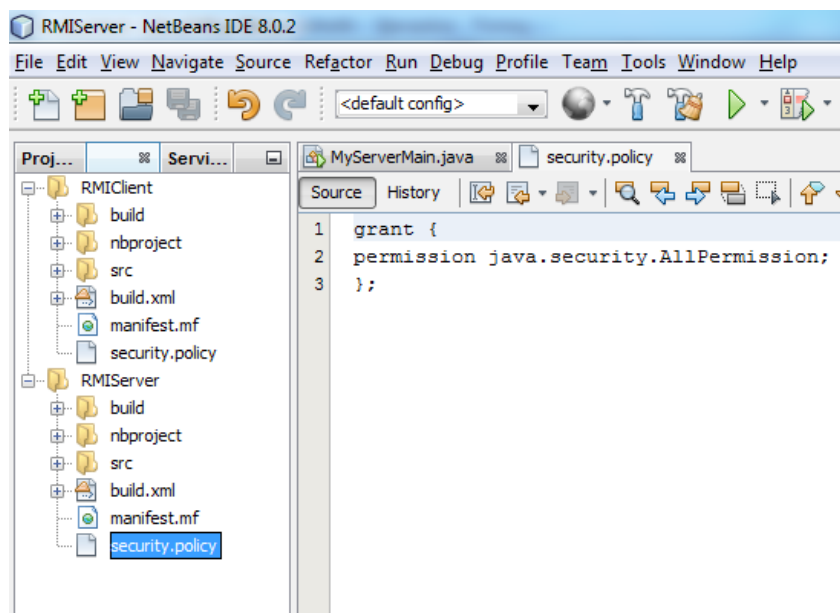
// Rejestr rmi można też uruchomić w programie poleceniem rmiregistry

**LocateRegistry.createRegistry(1099);**

### 3. Uruchamianie serwera

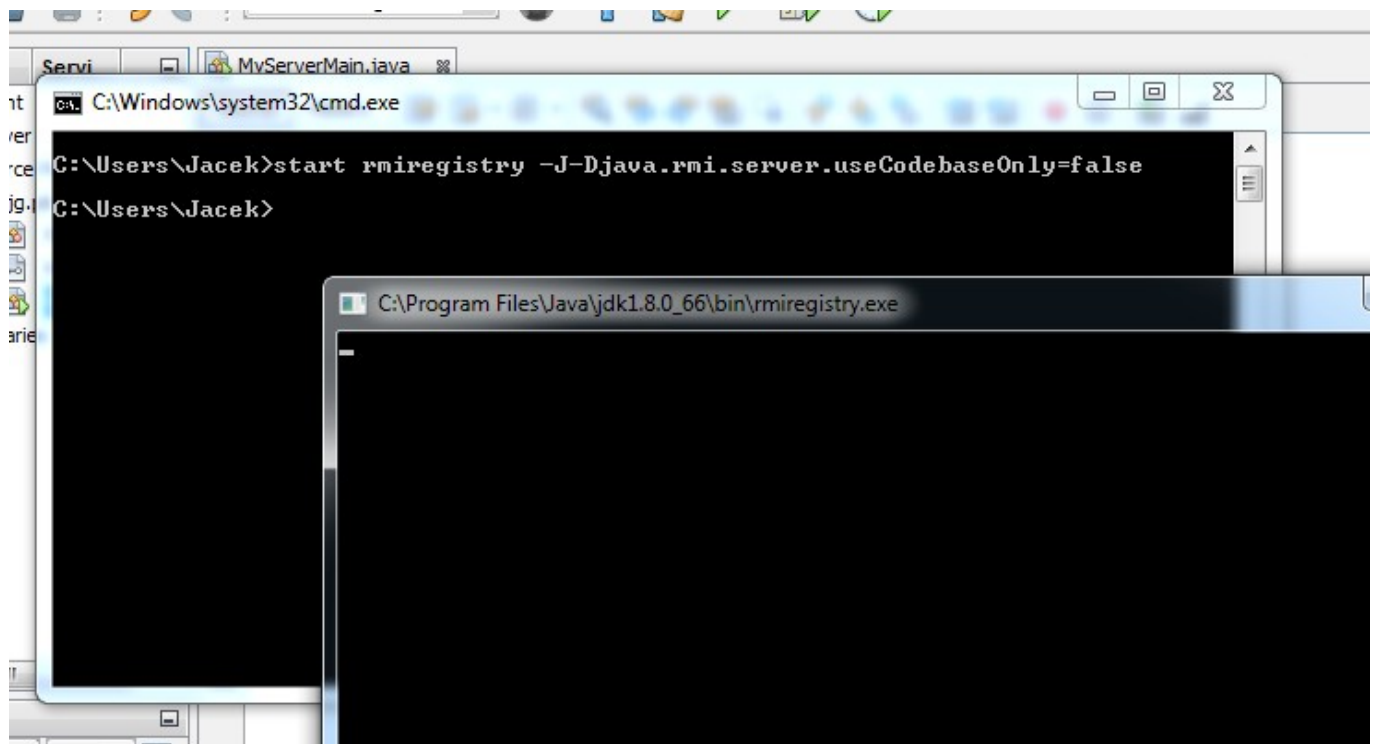
W projekcie serwera (u klienta również) musi być utworzony plik **security.policy** (katalog główny projektu)

- zawiera listę przywilejów dla programu
- przykład pliku zezwalającego na wszelkie operacje



#### 3.1 Uruchomić rmiregistry

alternatywnie można wystartować rmiregistry z katalogu klas (/build/classes) - start rmiregistry



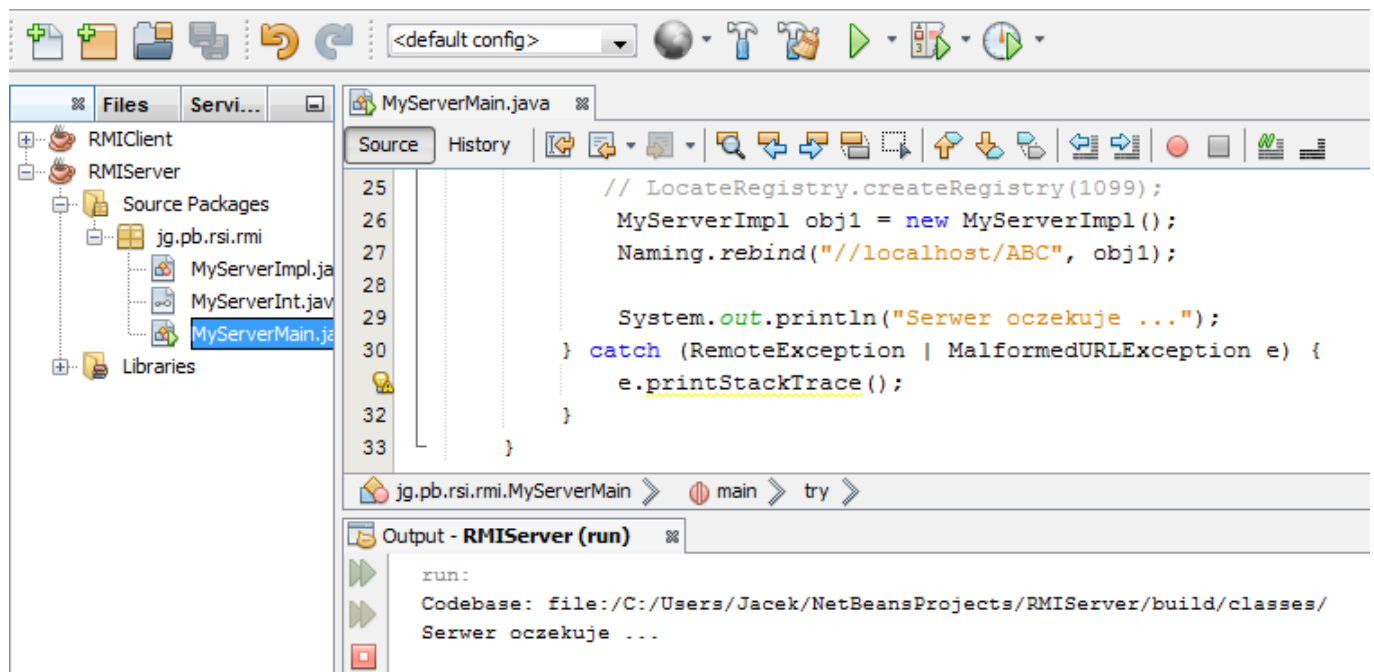
lub w programie (`LocateRegistry.createRegistry(1099);`)

Uruchom - przetestuj 2 warianty aplikacji:

1- rmiregistry uruchamiany z consoli,

2 - rmiregistry uruchamiany z programu serwera.

### 3.2 Uruchomić serwer



Serwer działa- oczekuje na klienta

### 3.3 Jeśli powstaje error

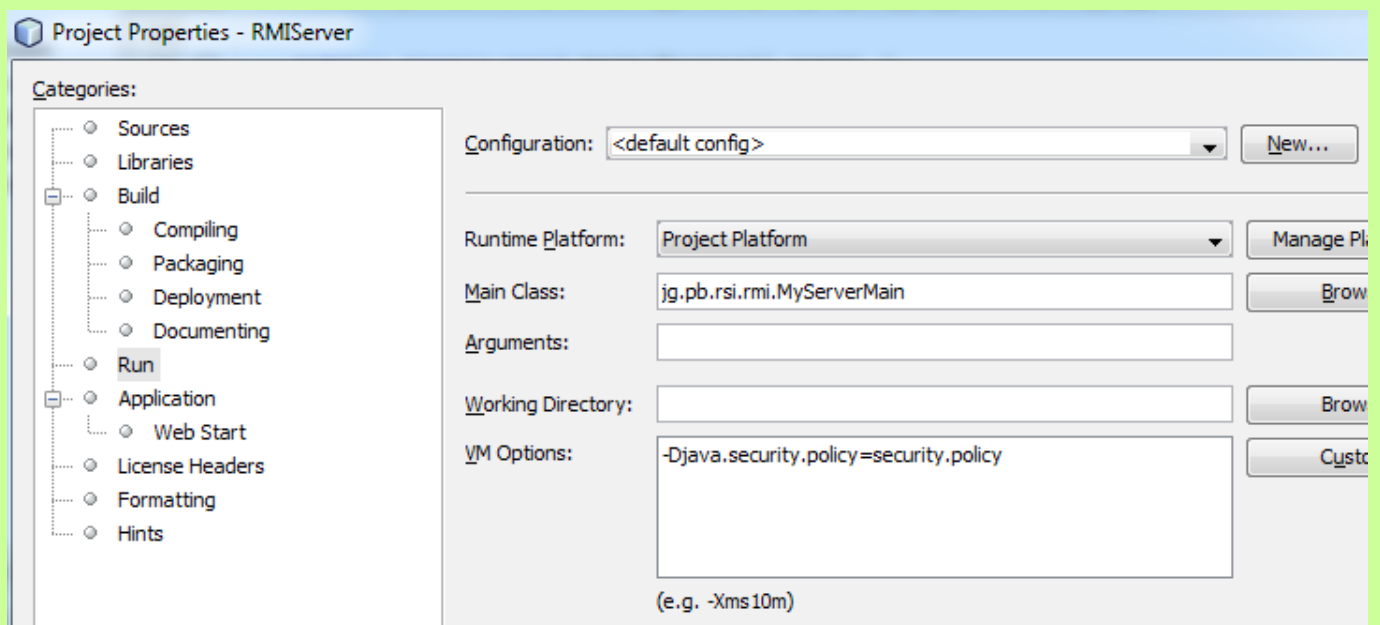
Exception in thread "main" java.security.AccessControlException: access denied

```
("java.util.PropertyPermission" "java.rmi.server.codebase" "write")
at java.security.AccessControlContext.checkPermission(AccessControlContext.java:472)
at java.security.AccessController.checkPermission(AccessController.java:884)
at java.lang.SecurityManager.checkPermission(SecurityManager.java:549)
at java.lang.System.setProperty(System.java:792)
at jg.pb.rsi.rmi.MyServerMain.main(MyServerMain.java:21)
Java Result: 1
```

Oznacza że plik security.policy nie został poprawnie wskazany

opcja dla VM (Virtualnej Maszyny Javy)

```
-Djava.security.policy=security.policy (lub w kodzie
System.setProperty("java.security.policy", "security.policy");)
```



### 3.4 Jeśli powstaje error

```
java.rmi.ServerException: RemoteException occurred in server thread; nested exception
is:
java.rmi.UnmarshalException: error unmarshalling arguments; nested exception is:
java.lang.ClassNotFoundException: jg.pb.rsi.rmi.MyServerInt
```

Oznacza że rmiregistry nie może znaleźć klas interfejsu(jg.pb.rsi.rmi.MyServerInt)

Należy ustawić zmienną wskazującą katalog klas

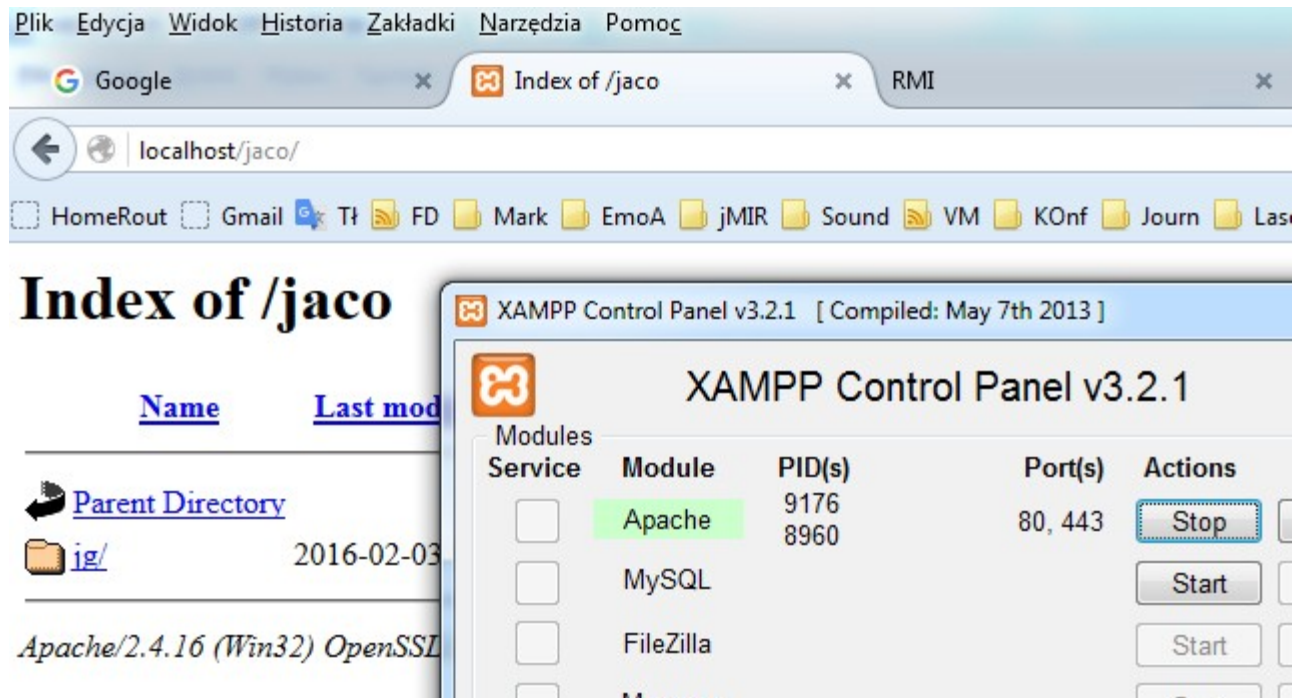
```
-Djava.rmi.server.codebase=file:/C:/Users/Jacek/NetBeansProjects/RMIServer/build/classes/
```

lub w kodzie programu

```
System.setProperty("java.rmi.server.codebase", "file:/C:/Users/Jacek/NetBeansProjects/RMIServer/build/classes/");
```

**3.5 Alternatywnie można umieścić klasę interfejsu na serwerze webowym, np.:**

```
System.setProperty("java.rmi.server.codebase", "http://192.168.1.102/jaco/");
```



### 3. Klient

**Klasa Klienta:**

- instaluje menadżera bezpieczeństwa obsługującego RMI
- odszukuje i pozyskuje odległy interfejs z serwera

```
package jg.pb.rsi.rmi;

import java.rmi.Naming;

public class MyClientMain {

    public static void main(String[] args) {

        System.setProperty("java.security.policy", "security.policy");

        System.setSecurityManager(new SecurityManager());
```

```

try {

MyServerInt myRemoteObject = (MyServerInt) Naming.lookup("//localhost/ABC");

String text = "Hallo :-)";

String result = myRemoteObject.getDescription(text);

System.out.println("Wysłano do servera: " + text);

System.out.println("Otrzymana z serwera odpowiedź: " + result);

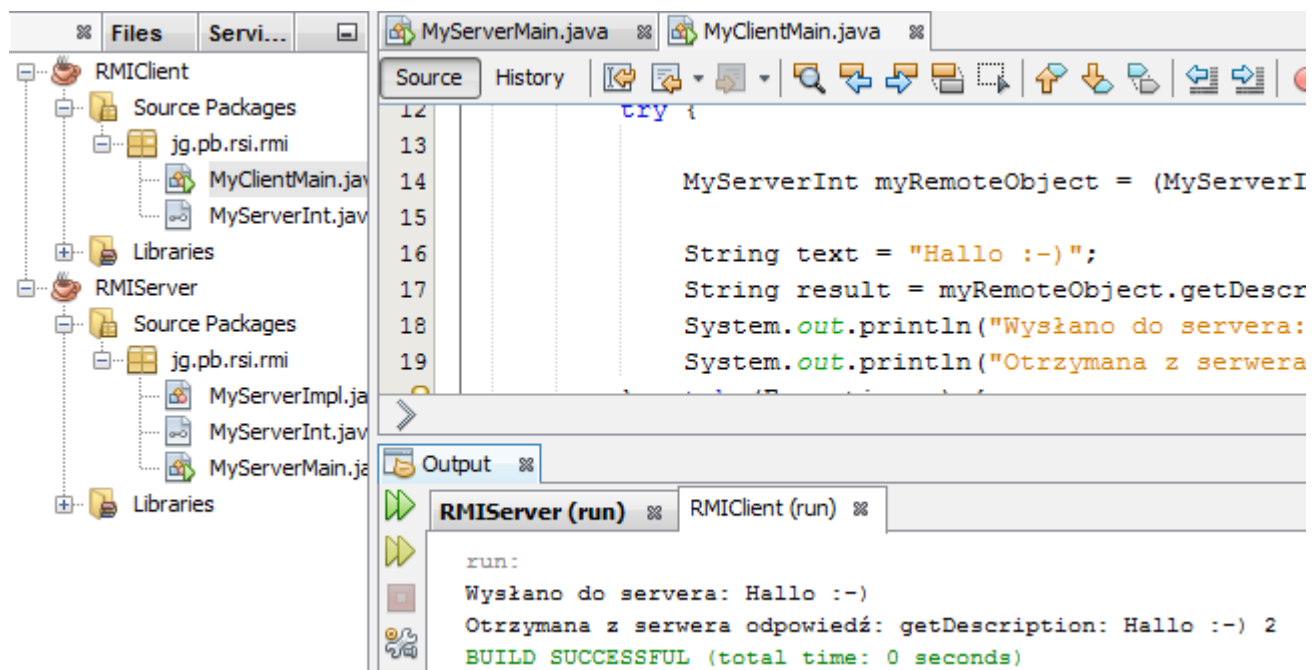
} catch (Exception e) {

e.printStackTrace();

}}

```

#### 4. Uruchamianie:



Klient wysłał String do serwera i otrzymał tekstową odpowiedź.

## Ćwiczenie 2. Kalkulator

Wykonać aplikację kalkulatora realizującego podstawowe operacje matematyczne (+, -, \*, /)

## Ćwiczenie 3. Java RMI na dwóch komputerach

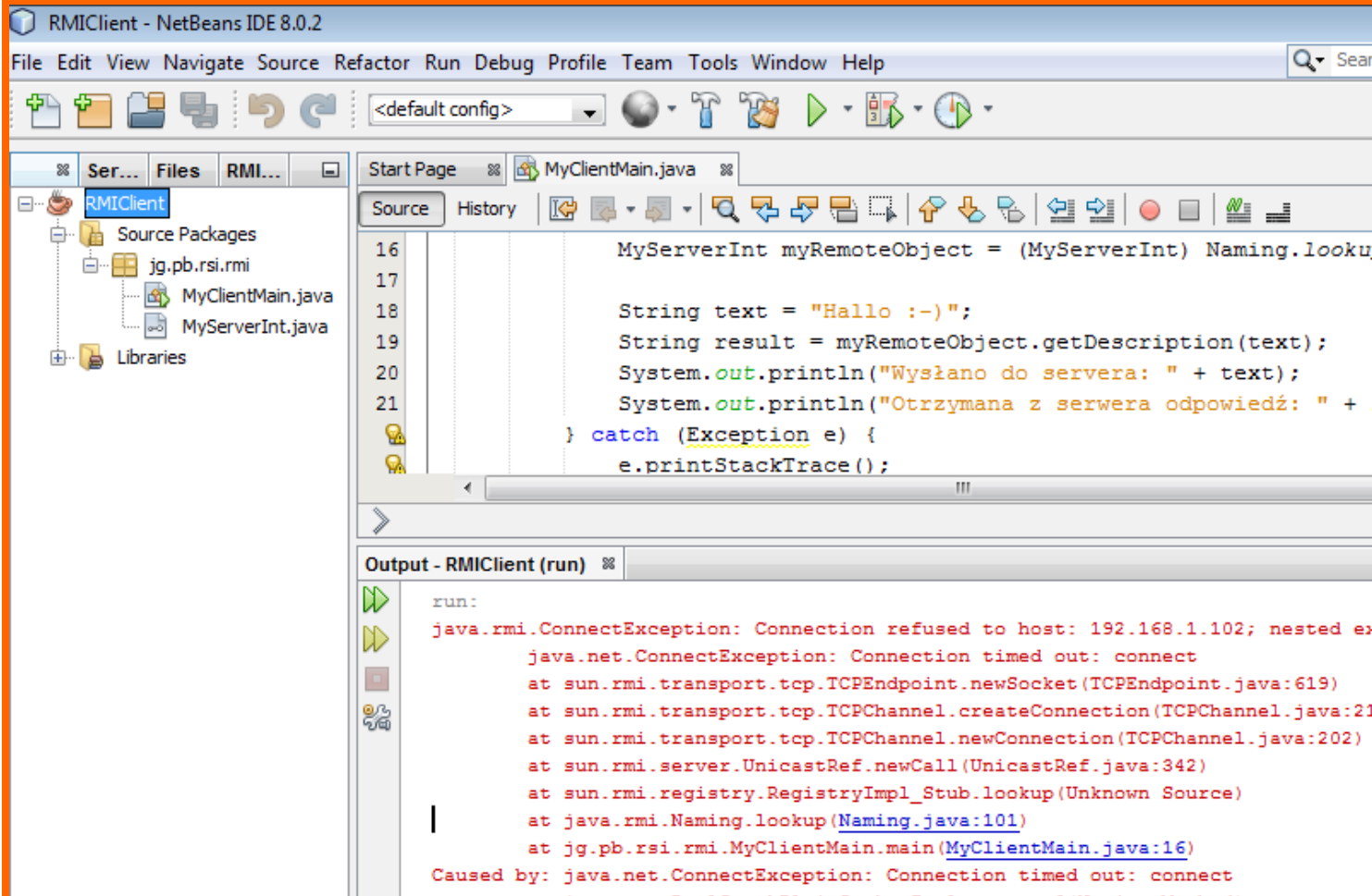
Uruchom aplikację Java RMI na dwóch komputerach (lub przy użyciu maszyny wirtualnej np. VM VirtualBox) – aplikacja klienta na jednym a aplikacja serwera drugim komputerze. Można dobrać się w pary.



Przy problemach można ustawić po stronie servera zmienną „java.rmi.server.hostname”

```
System.setProperty("java.rmi.server.hostname", "xx.xx.xx.xx");
```

W przypadku braku połączenie otrzymamy `ConnectionException` (connection time out..)



RMIClient - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page MyClientMain.java

Source History

```
16 MyServerInt myRemoteObject = (MyServerInt) Naming.lookup("jg.pb.rsi.rmi/MyServerInt");
17
18 String text = "Hallo :-)";
19 String result = myRemoteObject.getDescription(text);
20 System.out.println("Wysłano do servera: " + text);
21 System.out.println("Otrzymana z servera odpowiedź: " + result);
    } catch (Exception e) {
        e.printStackTrace();
    }
```

Output - RMIClient (run)

```
run:
java.rmi.ConnectException: Connection refused to host: 192.168.1.102; nested exception is
    java.net.ConnectException: Connection timed out: connect
    at sun.rmi.transport.tcp.TCPEndpoint.newSocket(TCPEndpoint.java:619)
    at sun.rmi.transport.tcp.TCPChannel.createConnection(TCPChannel.java:211)
    at sun.rmi.transport.tcp.TCPChannel.newConnection(TCPChannel.java:202)
    at sun.rmi.server.UnicastRef.newCall(UnicastRef.java:342)
    at sun.rmi.registry.RegistryImpl_Stub.lookup(Unknown Source)
    at java.rmi.Naming.lookup(Naming.java:101)
    at jg.pb.rsi.rmi.MyClientMain.main(MyClientMain.java:16)
Caused by: java.net.ConnectException: Connection timed out: connect
```

Upewnić się, że rmiregistry jest dostępny w sieci (np. wyłączyć firewall)

RMIServer - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Files Servi... MyServerMain.java

Source History

```
25 // LocateRegistry.createRegistry(1099);
26 MyServerImpl obj1 = new MyServerImpl();
27 Naming.rebind("//localhost/ABC", obj1);
28
29 System.out.println("Serwer oczekuje ...");
```

jg.pb.rsi.rmi.MyServerMain > main > try >

Output - RMIServer (run)

```
run:
Codebase: http://192.168.1.102/jaco/
Serwer oczekuje ...
MyServerImpl.getDescription: Hallo :- ) 1
MyServerImpl.getDescription: Hallo :- ) 2
MyServerImpl.getDescription: Hallo :- ) 3
```

ZoneAlarm

Check Point SOFTWARE TECHNOLOGIES LTD.

Scan Update Tools Help

**YOUR COMPUTER IS AT RISK** Fix Now!

ANTIVIRUS FIREWALL IDENTITY & DATA

**Basic Firewall** Your firewall is not properly set

Fix Now OFF

View Zones | Settings

**Application Control** Blocks dangerous behaviors and unauthorized Internet transmissions

985 programs secured ON

Settings