

# Laboratorium RMI cz. 2

Dominik Radziszowski,  
Sławomir Zieliński

# Rozbudowujemy naszą aplikację – wersja 2

◆ ~/student-426/[nazwisko]

Skopiować kod klienta: ~/lab1/klient/\*

Uruchomić (localhost, port 4260+numer)

◆ O co nam chodzi:

Chcemy wykorzystać możliwość dynamicznego ładowywania stubów przez klienta

# Dynamiczne ładowanie klas

◆ rmiregistry: poza referencją przechowuje informacje gdzie można znaleźć pliki .class

◆ Serwer:

```
java -Djava.rmi.server.codebase=  
    file://c:/...../classes/  
    http://myserver:8080/archiwum.jar  
    ftp://ftp.server/classes/
```

◆ Klient

Klasy lokalne są ładowane w pierwszej kolejności  
– usunąć...

Musimy zainicjować security managera

- Djava.security.manager
- Djava.security.policy

# Rozbudowujemy naszą aplikację – wersja 2

## ◆ Co zmieniamy

Klient nie ma stubów obiektów serwera  
w zmiennej CLASSPATH

WŁASNE MA!

Konfigurujemy parametry uruchomienia

klienta: `-Djava.security.policy=policy.all`

serwera: `-Djava.rmi.server.codebase=...`

W oznaczonym miejscu umieszczamy stuby  
obiektów serwera

w naszym przypadku tylko obiektu `NoteBoardImpl`

# Uruchomienie

# No to siup!

## ◆ Rmiregistry

```
rmiregistry -J-Djava.security.policy=~/.javaAll.policy
```

## ◆ Serwer:

```
java
```

```
-Djava.rmi.server.codebase=file:/home/jasmin/llsr/lab2/serwer/
```

```
-Djava.security.policy=/home/jasmin/llsr/javaAll.policy
```

```
agh.rozproszone.NoteBoardServer
```

## ◆ Klient:

```
java
```

```
-Djava.security.manager
```

```
-Djava.security.policy=/home/jasmin/llsr/javaAll.policy
```

```
agh.rozproszone.NoteBoardClient
```

Uwaga na „/” na końcu ścieżek

# Jak uczynić serwer dynamicznym?

## ◆ Co zmieniamy:

Serwer nie posiada stubów klas klienta

Serwer może korzystać z RMI Class Loadera

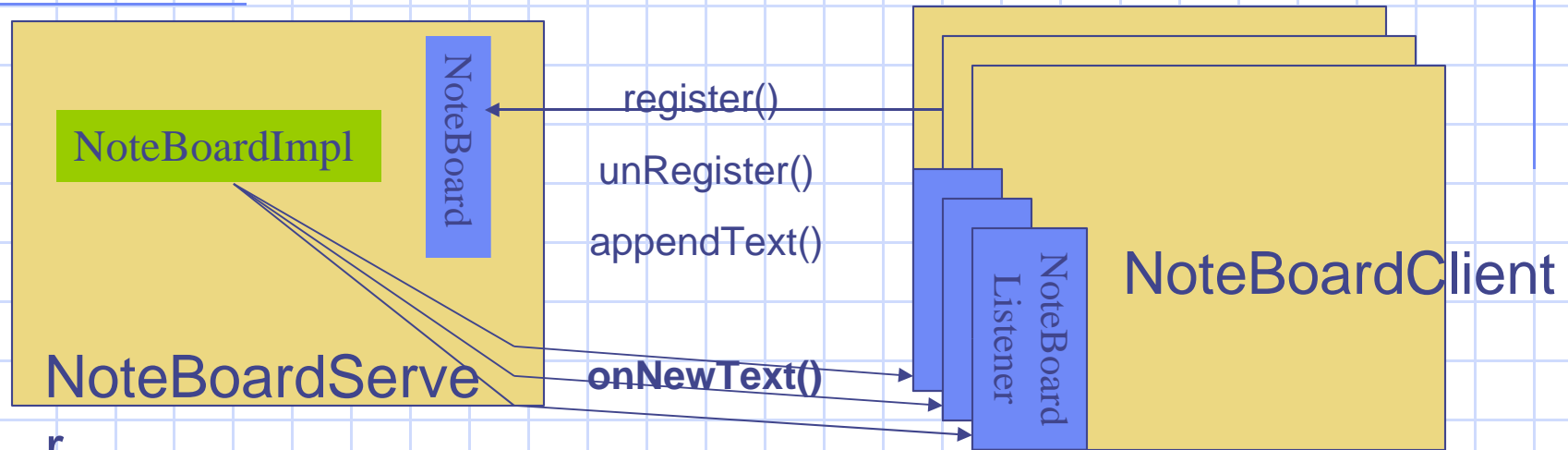
## ◆ Czego nie zmieniamy:

Serwer jest uruchamiany wcześniej, niż klient

## ◆ Co zyskujemy:

Serwer jest mniej uzależniony od zmian w kliencie (i vice versa)

# NoteBoard v2.1



Tak jak poprzednio tylko:

`onNewText(Note note);`

```
public class Note implements Serializable{  
    private String text;  
    public Note(String text) {  
        this.text = text;  
    }  
    public String toString()  
    { return text;  
}
```

**No to siup!**

# SecurityManager

◆ Zamia: -Djava.security.manager

```
//security manager
```

```
if (System.getSecurityManager() == null) {  
    System.setSecurityManager(new RMISecurityManager());  
}
```

◆ Plik policy:

```
grant {  
    permission java.security.AllPermission;  
};  
grant {  
    permission java.net.SocketPermission "*:1024-65535", "connect,accept";  
    permission java.net.SocketPermission "*:80", "connect";  
};  
grant codeBase "file:${java.home}/lib/-" {permission java.lang.RuntimePermission  
    "loadLibrary.*";    permission java.lang.RuntimePermission "accessClassInPackage.*";  
    permission java.lang.RuntimePermission "queuePrintJob";  
}
```



# Activation – jak tego użyć?

- ◆ Dodatkowy konstruktor w NoteBoardImpl

Parametry: ActivationID, MarshalledObject

- ◆ Modyfikacja klasy NoteBoardServer

- ◆ rmid:

<http://java.sun.com/j2se/1.4/docs/tooldo>

- ◆ rmiregistry

# Zmiany w NoteBoardImpl

```
// Dodatkowy konstruktor - wywoływany przez RMID
public NoteBoardImpl(
    ActivationID id,
    MarshalledObject data )
    throws RemoteException
{
    // Rejestrujemy się w „activation system”
    // i eksportujemy referencję na anonimowym porcie
    Activatable.exportObject(this, id, 0);
}
```

# Klasa rejestrująca serwer -1

```
public static void main(String[] args) throws Exception {
    System.setSecurityManager(new RMISecurityManager());
    Properties props = new Properties();
    props.put("java.security.policy", "~/javaAll.policy");

    // Tworzymy Środowisko uruchomieniowe dla NoteBoard
    ActivationGroupDesc.CommandEnvironment ace = null;
    ActivationGroupDesc exampleGroup =
        new ActivationGroupDesc(props, ace);

    // Rejestrujemy utworzone Środowisko
    ActivationGroupID agi =
        ActivationGroup.getSystem().registerGroup(exampleGroup);
}
```

# Klasa rejestrująca serwer -2

```
// Miejsce przechowywania naszych klas
String location = "file:~/dydakt/lab2/serwer";

MarshaledObject data = null;
ActivationDesc desc = new ActivationDesc
(agi, "agh.rozproszone.NoteBoardImpl",
    location, data);
```

# Klasa rejestrująca serwer -3

```
// Rejestrujemy się w RMID
NoteBoard ari =
(NoteBoard)Activatable.register(desc);

// Efekt rejestracji „aktywujący” stub
Naming.rebind("NoteBoard", ari);
System.out.println("Exported NoteBoard");

// i tyle...
System.exit(0);
}
```

# Do dzieła!!!

NoteBoardImpl – zmienic implementacje  
ActivationSetup – stworzyc programik