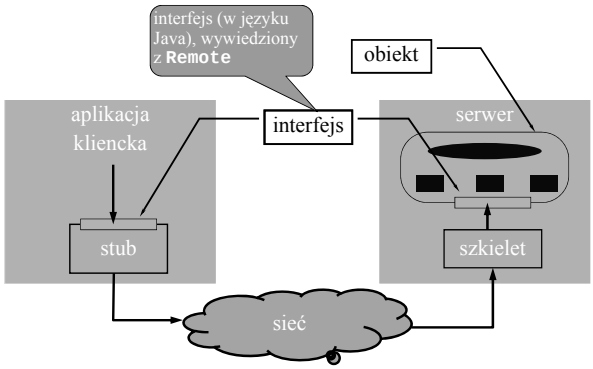
Wywoływanie metod zdalnych

Podejście obiektowe do budowy systemów rozproszonych

Wywoływanie metod zdalnych — model systemu



Obiekt zdalny w środowisku Java

- Mechanizm RMI umożliwia tworzenie obiektów zdalnych (brak bezpośredniego wsparcie dla tworzenia obiektów rozporszonych)
- Jedyna forma zdalnego dostępu polega na wywoływaniu metod wyspecyfikowanych w interfejsie wywiedzionym (dziedziczącym) `java.rmi.Remote`
- Interfejs zdefiniowany jest w języku implementacji
- Obiekt może implementować wiele interfejsów
- Ten sam interfejs może być implementowany przez wiele obiektów
- Interfejs traktowany jest jak typ danych

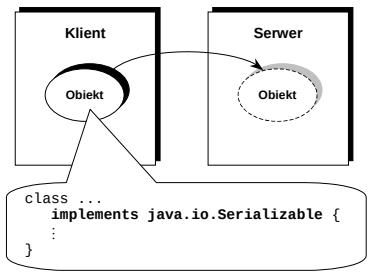
Dostępność obiektu

- Informacja o typie obiektu (czyli o zdalnym interfejsie) dostępna jest w czasie kompilacji.
- Wiązanie obiektu jest jawne i odbywa się w czasie wykonania
- Trwałość obiektu
  - obiekt udostępniany przez `UnicastRemoteObject` ma charakter przejściowy (istnieje tylko w czasie działania serwera)
  - dostępny jest mechanizm obiektów aktywowalnych, ale brak bezpośredniego wsparcia dla utrwalania stanu obiektu

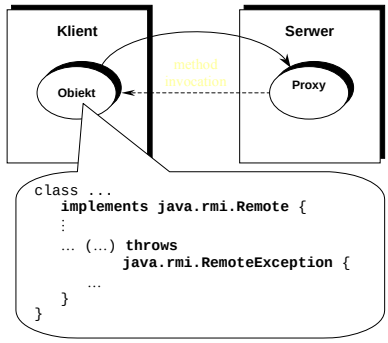
Przekazywanie obiektów jako parametrów

- Obiekty mogą być przekazywane przez wartość (kopię) — konieczna jest deklaracja implementacji interfejsu `java.io.Serializable` w klasie obiektu.
- Obiekty zdalne (implementujące interfejs `Remote`) przekazywane są przez referencję — do zdalnej metody przekazywana jest zdalna referencja, za którą udostępniany jest proxy (stub).
- Brak bezpośredniego wsparcia dla przekazywania parametrów przez kopiowanie i odtwarzanie.

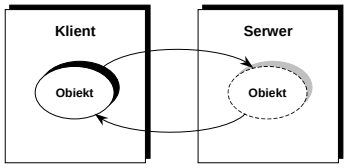
Przekazywanie przez wartość — przykład w Java RMI



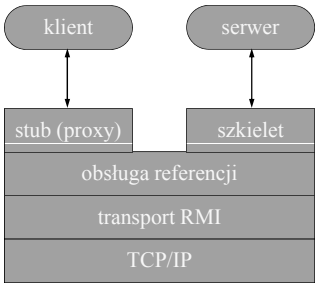
Przekazywanie przez referencję — przykład w Java RMI



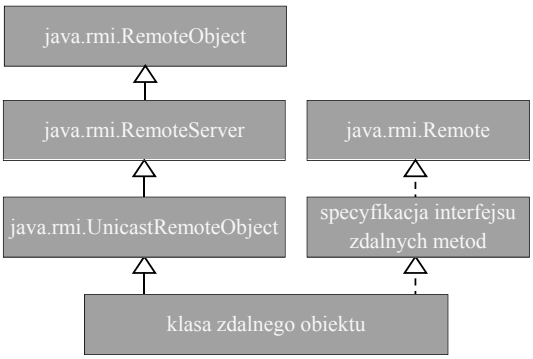
Przekazywanie przez kopiowanie i odtwarzanie



Architektura RMI



Hierarchia klas w definicji zdalnego obiektu



Tworzenia aplikacji rozproszonej w środowisku Java RMI (1)

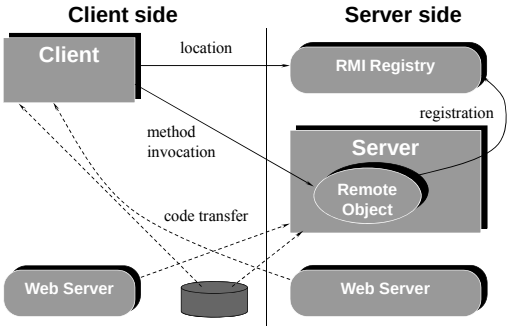
1. Zdefiniowanie i implementacja odpowiednich klas (w szczególności klas dla obiektów dostępnych zdalnie)
   - zdefiniowanie interfejsu pochodnego od `Remote`
   - zdefiniowanie klasy wywiedzionej z klasy `java.rmi.server.UnicastRemoteObject`, implementującej interfejs pochodny od `Remote` lub użycie statycznej metody `exportObject` klasy `UnicastRemoteObject`
2. Kompilacja źródeł (`javac`, `rmic`)
   - `javac xxx.java` ⇨ `xxx.class`
   - `rmic xxx` ⇨ `xxx_Stub.class`, `xxx_Skel.class` (we wczesnych wersjach Javy)

zbędne w Java 5

Tworzenia aplikacji rozproszonej w środowisku Java RMI (1)

3. Udostępnienie wygenerowanego kodu klas
   - wspólny system plików
   - kopia kodów klas w różnych systemach plików
   - udostępnianie kodu przez serwer www
4. Uruchomienie aplikacji
   - uruchomienie `rmiregistry` (name server)
   - uruchomienie serwera: utworzenie zdalnych obiektów i ich rejestracja w `rmiregistry`
   - uruchomienie klienta: zlokalizowanie zdalnych obiektów (odwołanie do `rmiregistry`) i wywoływanie zdalnych metod

Komunikacja między klientem a serwerem



Interfejs `java.rmi.Remote`

Komunikacja pomiędzy serwerem (obiektem) a klientem jest określona przez definicję interfejsu pochodnego od interfejsu `Remote`. Klasa zdalnego obiektu musi implementować ten interfejs

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Compute extends Remote {
   Object executeTask(Task t) throws RemoteException;
}
```

Zdalne udostępnianie obiektu (serwer)

Definicja klasy obiektu:

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

public class ComputeEngine extends UnicastRemoteObject
                           implements Compute {
      public ComputeEngine() throws RemoteException {
         super();
      }
      public Object executeTask(Task t) {
         return t.execute();
      }
```

Utworzenie i rejestracja zdalnego obiektu

```
public static void main(String[] args) {
 if (System.getSecurityManager() == null) {
   System.setSecurityManager(new RMISecurityManager());
 }
 String name = "//host:1099/Compute";
 try {
    Compute engine = new ComputeEngine();
    Naming.rebind(name, engine);
    System.out.println("ComputeEngine ok");
  } catch (Exception e) {
  System.err.println("ComputeEngine excep."
                       + e.getMessage());
      e.printStackTrace();
   } }}
```

Zdalne wywoływanie metod

```
import java.rmi.*;
public class launchComp {
  public static void main(String args[]) {
    if (System.getSecurityManager() == null){
       System.setSecurityManager(
                 new RMISecurityManager());
    }
    try {
      String name = "//"+args[0]+"/Compute";
      Compute comp = (Compute) Naming.lookup(name);
      CompArea task = new CompArea(...);
      BigDecimal arae =
              (BigDecimal) (comp.executeTask(task));
         System.out.println(area);
      } catch (Exception e) {
         ...
      } } }
```

Przekazywanie parametrów

- Jeżeli przekazujemy obiekt przez wartość, to klasa tego obiektu musi implementować interfejs `Serializable`.
- Jeśli przekazujemy obiekt przez referencja, to klasa tego obiektu musi implementować interfejs `Remote`.

```
public interface Task extends
                          java.io.Serializable {
      Object execute();
   }
```

```
public class CompArea implements Task {
   ⋮
}
```

Lokalizowanie zdalnych obiektów

- Rejestr — wiąże z nazwami i udostępnia zdalne obiekty
- Tworzenie rejestru — rmiregistry [<port number>]
- Klasa LocateRegistry — umożliwia tworzenie rejestru (obiekty klasy Registry)
- Klasa Registry — klasa obiektu-rejestru
- Klasa Naming — ułatwia korzystanie z rejestru, umożliwiając jego specyfikację w adresie URL

Klasa `LocateRegistry`

```
static
Registry createRegistry(int port)
                throws RemoteException
static
Registry createRegistry(String host, int port)
                throws RemoteException
static
Registry getRegistry(int port)
                throws RemoteException
static
Registry getRegistry(String host, int port)
                throws RemoteException
```

Klasa `Registry`

"Compute"

```
void bind(String name, Remote obj)
       throws RemoteException,
         AlreadyBoundException, AccessException
void rebind(String name, Remote obj)
       throws RemoteException, AccessException
void unbind(String name)
       throws RemoteException,
         NotBoundException, AccessException
String[] list()
       throws RemoteException, AccessException
Remote lookup(String name)
       throws RemoteException,
         NotBoundException, AccessException
```

Klasa `Naming` (1)

"//host:1099/Compute"

```
static
void bind(String urlName, Remote obj)
  throws RemoteException,
    AlreadyBoundException, AccessException,
    MalformedURLException, UnknownHostException

static
void rebind(String urlName, Remote obj)
  throws RemoteException, AccessException,
     MalformedURLException, UnknownHostException
```

Klasa `Naming` (2)

```
static
void unbind(String urlName)
  throws RemoteException, NotBoundException,
        AccessException, MalformedURLException,
        UnknownHostException
static
String[] list()
  throws RemoteException, AccessException
    MalformedURLException, UnknownHostException
static
Remote lookup(String urlName)
  throws RemoteException, NotBoundException,
        AccessException, MalformedURLException
```