DISTRIBUIRANI SISTEMI

RMI

RMI

REMOTE METHOD INVOCATION

Java RMI

RMI = Remote Method Invocation

• Omogućava poziv metoda objekta koji pripada drugoj JVM (Java Virtual Machine) i to:

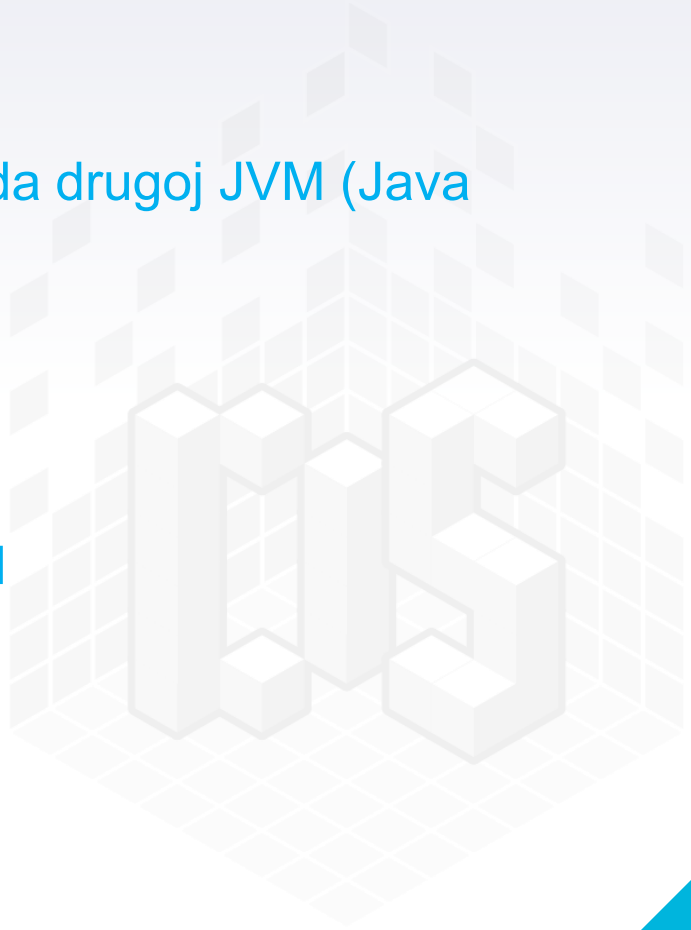
– ili na udaljenom računaru

– ili na istom računaru, različiti procesi

• Svaki proces se izvršava na drugoj JVM

• Različiti adresni prostor za svaki proces/JVM

RMI obezbeđuje objektno-orjentisani RPC



Java RMI

- U osnovi RPC princip
  - Poziv udaljenog metoda
  - Prosleđivanje argumenata metodu
  - Primanje rezultata izvršenja udaljenog metoda
- RPC evolucija
  - RPC bez OO koncepta
  - CORBA
  - RMI

Procesi učesnici

- **Klijent**
  - Proces koji poziva metod udaljenog objekta
- **Server**
  - Proces koji sadrži udaljeni objekat
  - Za server je to lokalni objekat
- **Registar objekata (rmiregistry)**
  - Server imena koji povezuje objekte sa imenima,
  - Server po startovanju registruje svoje objekte u registru objekata preko simboličkog imena (preko URL)

Procesi učesnici

URL ime:

rmi://hostname:port/name

e.g.:

rmi://crapper.pk.org:12345/MyServer

• Klijent pre poziva metoda mora prvo da kontaktira registar objekata da bi mogao da pristupi udaljenom objektu

Java RMI Interfejsi i klase

- Kao podrška opisanoj RMI komunikaciji služe:
  - Remote interfejs
  - Stub i skeleton
  - Remote klasa (objekat)

Remote interfejs

- Specificira metode kojima se može udaljeno pristupati
- Implementiran od strane remote klase čija instanca postaje udaljeni objekat
- Nasleđuje **java.rmi.Remote** interface

Klijent stub

- Klijent Stub
- Nalazi se u adresnom prostoru klijenta
- Igra ulogu proxy-ja remote objekta
- Implementira Remote interfejs
- Klijent poziva metode Klijent stub-a lokalno

Klijent stub

- Klijent stub obezbeđuje konekciju sa remote objektom
- Šalje argumente i prima rezultate od remote objekta
- Izvodi marshaling i unmarshaling

Server skeleton

- RMI Skeleton
- Nalazi se u adresnom prostoru servera
- Prima argumente poslate od strane klijent stuba i šalje rezultat istom
- Izvodi marshaling i unmarshaling
- Određuje koji metod remote objekta će biti pozvan

Remote klasa (objekat)

- Implementacija remote interfejsa
- Remote objekat mora biti izvezen (exported) da bi spreman da prima pozive od udaljenih klijenata
- Izvoz je omogućen nasleđivanjem klase **java.rmi.server.UnicastRemoteObject**

Arhitektura RMI sistema

- Definicija interfejsa za udeljene servise (metode)- remote interfejs
- Implementacija udaljenih servisa (metoda)-remote klasa
- Stub i skeleton
- Server koji obezbeđuje udaljene servise
- RMI naming servisi koji omogućavaju klijentu da nađe udaljene sevise
- Provajder fajlova (HTTP ili FTP server)
- Klijent program koji poziva udaljene servise

Koraci u razvoju Rmi aplikacije (1)

- Definisati remote interfejs
  - Interfejs mora da nasleđuje **java.rmi.Remote** interfejs
  - Definisati metode koji se pozivaju udaljeno
  - Metodi moraju biti public i da bacaju **java.rmi.RemoteException**
  - Tip podataka koji se prosleđuje kao parametar ili povratna vrednost iz metoda mora biti tipa Remote interfejsa (ne implementacije tog interfejsa) ili tipa klase koja implementira Serializable

Koraci u razvoju Rmi aplikacije (2)

- Napisati klasu koja implemetira remote interfejs (za svaki remote interfejs)
  - Nasleđuje **UnicastRemoteObject** klasu
- Obezbediti implementaciju svakog metoda interfejsa
- Napisati server klasu
  - Sadrži main() metod
  - Instacirati i exportovati remote objekat
  - Registrovati stub za remote objekat u registru objekata

Koraci u razvoju Rmi aplikacije(3)

- Napisati klijent klasu
  - Sadrži main() metod
  - Za dato ime dobiti referencu (instancu stuba) na objekat iz registra objekata (rmiregistry)
  - Pozivati udaljene metode

1.Kreiranje Remote Interfejsa

- Remote interfejs mora da nasleđuje **java.RMI.Remote**.
- Svaki metod u interfejsu se deklariše tako da može da baca izuzetke **RemoteException**

Calculator.java

```
public interface Calculator extends java.rmi.Remote {
    public long add(long a, long b) throws java.rmi.RemoteException;
    …
}
```

2.Implementacija Remote interfejsa

Remote klasa koja implementira Remote interfejs, treba da nasleđuje **java.rmi.server.UnicastRemoteObject** klasu da bi se povezala na RMI sistem. Objekti ove klase postoje u adresnom prostoru servera i mogu se pozivati udaljeno. Ako ovi objekti imaju metode kojih nema u interfejsu oni se mogu pozivati lokalno Poziv udaljenog metoda može da bude neuspešan pa svaki metod ove klase mora da u deklaraciji ima **throws java.rmi.RemoteException;**

CalculatorImpl.java

```
public class CalculatorImpl extends java.rmi.server.UnicastRemoteObject
implements Calculator {
  public CalculatorImpl() throws java.rmi.RemoteException {
    super();
  }
  public long add(long a, long b) throws java.rmi.RemoteException {
    return a + b;
  } …
}
```

3. Stub i skeleton

• U starim verzijama Jave, RMI kompajler (rmic) se koristio za generisanje stuba i skeletona.

Rmic Ime_remote_klase

• U verziji 1.5, ove klase se generišu dinamički i rmic se ne mora koristiti.

4. Host server

U okviru ovog programa obavlja se kreiranje i registrovanje remote objekta u registru objekata (rmiregistry) koji se npr. nalazi na lokalnom hostu i “osluškuje” na portu 1099. Prilikom kreiranja objekta poziva se **UnicastRemoteObject** konstruktor u okviru koga se proizvoljan port pridružuje kreiranom objektu i on će osluškivati pozive na tom portu.

Registrovanje se obavlja pozivom metoda klase **java.rmi.Naming**: **Naming.rebind**(object_name, stub) gde je object_name String koji imenuje udaljeni objekat, a stub je je stub instanca za dati remote objekat (tj. referenca na remote objekat)

4. Host server

Object_name je URL formatirani string oblika

rmi://computerName:port/objectName

gde computerName i port ukazuju na lokaciju i port regsitra objekata (rmiregistry). Ako su izostavljeni, localhost i 1099 port se podrazumevaju

U okviru registrovanja u registru objekata ime object_name se povezuje sa klijent stub instancom stub_object koja sadrži informaciju o hostu i portu na koji će se upućivati pozivi metoda za dati objekat. (mogu da sadrže i informacije potrebne za dobijanje koda klijent stuba)

4. Host server

CalculatorServer.java

```
import java.rmi.Naming;
public class CalculatorServer {
  public CalculatorServer() {
    try {
       Calculator c = new CalculatorImpl();
       Naming.rebind("rmi://localhost:1099/CalculatorService", c);
       …
    }
  …
}
```

5.Klijent

Klijent je program koji poziva udaljeni metod. Da bi to uradio klijent koristi referencu na udaljeni objekat (object handle). Za to je potrebno znati:

1. Adresu mašine na kojoj se izvršava rmiregistry u kome se registruje udaljeni objekat
2. Port na kome se rmiregistry izvršava (ako je 1099 može se izostaviti)
3. Ime objekta u rmiregistry-ju Klijent u u ovu svrhu koristi lookup metod klase **java.rmi.Naming**. Povratna vrednost iz ovog metoda je instanca klijent stuba i mora biti cast-ovana u tip remote interfejs

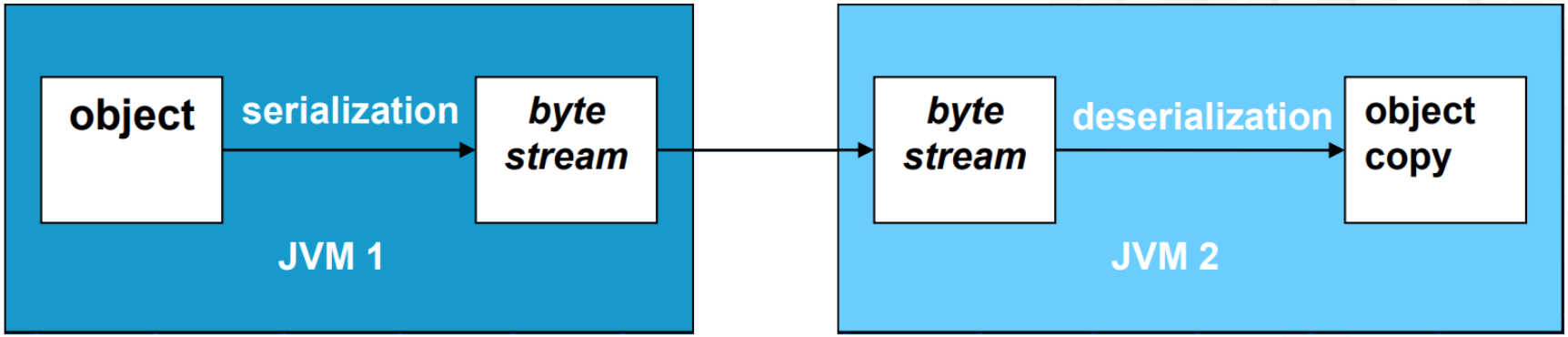
`CalculatorClient.java`

```
…
Calculator c = (Calculator) Naming.lookup( "rmi://localhost/CalculatorService");
System.out.println( c.sub(4, 3) );
System.out.println( c.add(4, 5) );
System.out.println( c.mul(3, 6) );
System.out.println( c.div(9, 3) );
…
```

Tipovi klasa u RMI

Remote klasa- instance ove klase se mogu koristiti ili lokalno ili udaljeno. U potonjem slučaju ovoj klasi se pristupa preko reference na objekat (klijent stuba). CalculatorImpl je primer remote klase.

Serializable klasa (implementira Serializable interfejs koji nema metode) čije instance se mogu kopirati iz jednog adresnog prostora u drugi, kao što je prikazano na slici



Prenos parametara

• Ako se serializable objekat prosledjuje kao parametar (ili povratna vrednost) poziva udaljenog metoda, onda se vrednost objekta prenosi iz jednog adresnog prostora u drugi. Podaci primitivnih tipova i lokalni objekti se prenose kao serializable.

• Semantika “prenos-po-vrednosti”

• Ako se objekat remote klase prosledjuje kao parametar (ili povratna vrednost) onda se referenca na objekat prenosi iz jednog iz jednog adresnog prostora u drugi

• Semantika “prenos-po-referenci” U primeru Calculator prenose se podaci primitivnih tipova, tj. po vrednosti

PRIMERI...