RMI приложения

Основни понятия

- RMI приложения се състои от две отделни програми на клиент и сървър.
- Сървър програмите създават “отдалечени” обекти (remote objects), които са достъпни и чакат клиентите да извикат методи на тези обекти.
- Клиентската програма получава отдалечен достъп до един или повече отдалечени обекти от сървъра и след това извиква методи от тях.

Отдалечени интерфейси, обекти и методи

- Разпределените приложения се изграждат чрез използването на Java RMI, което е изградено от интерфейси и класове.
- Механизмът на Java за отдалечено обръщение към методи (RMI) позволява на обекти от една JVM да викат методи на обекти от друга JVM.

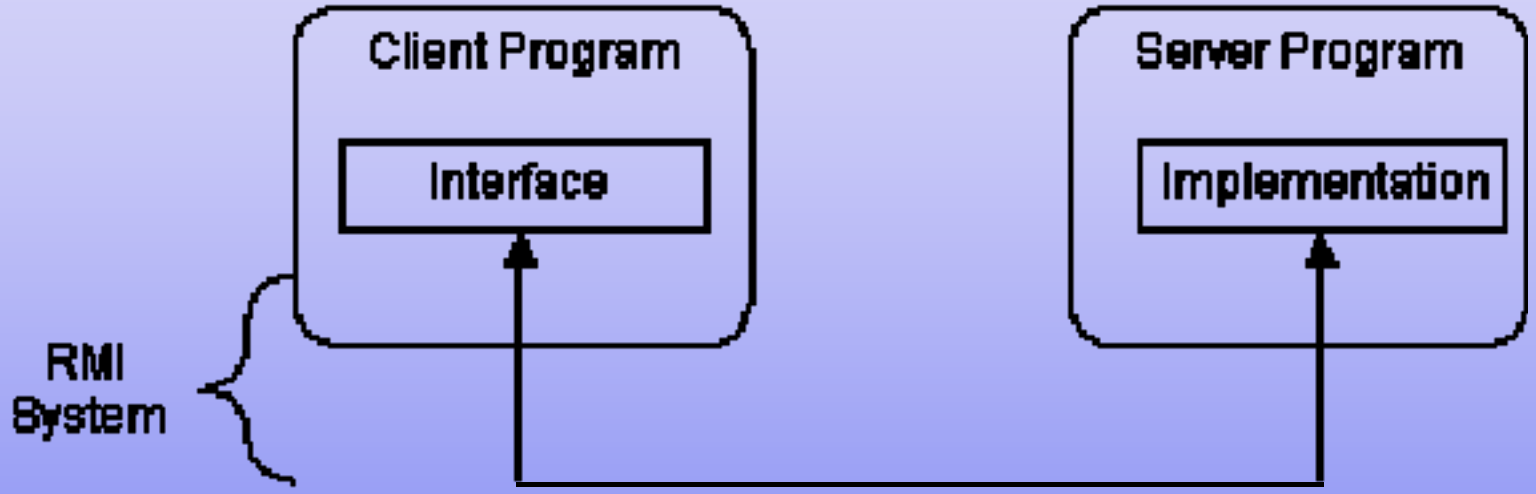
Дефиниции

- **Отдалечен обект** - обект, методите на който могат да бъдат извикани от друга Java машина - потенциално на друг компютър. Обект от този тип се задава чрез един или повече **отдалечени интерфейси**, които представляват Java интерфейси, декларациращи **отдалечените методи**.
- **Отдалечено извикване на методи** (*Remote method invocation* - RMI) - извикването от друга Java машина на метод от отдалечения обект, дефиниран в отдалечения интерфейс. Извикването става по същия начин както в локалната Java машина.

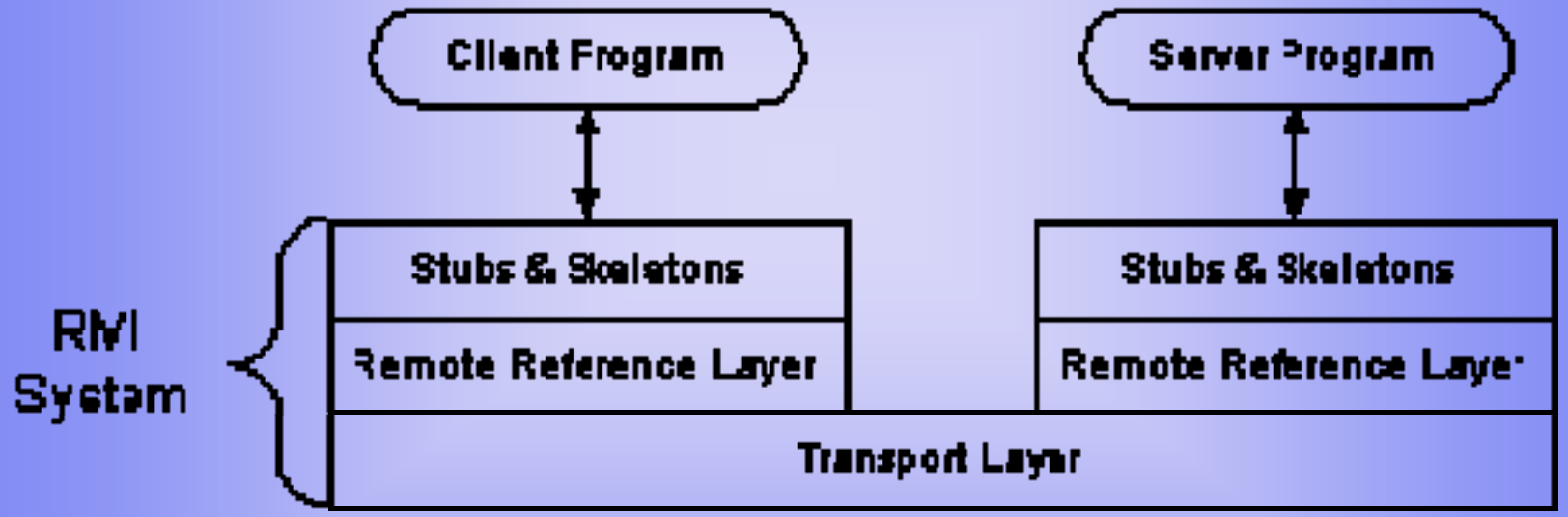
Основни цели

- Интегриране на разпределения обектен модел по естествен начин запазвайки семантиката на Java обектите.
- Еднотипен подход към локалните и отдалечените Java обекти.
- Запазване на проверката на типовете данни осигурявана от виртуалната Java машина
- Поддръжка на сигурността при отдалеченото извикване чрез използване на Java механизмът за сигурност.
- Поддържа връзка между сървъри и аплети.

Архитектура на Java RMI



Архитектура на Java RMI



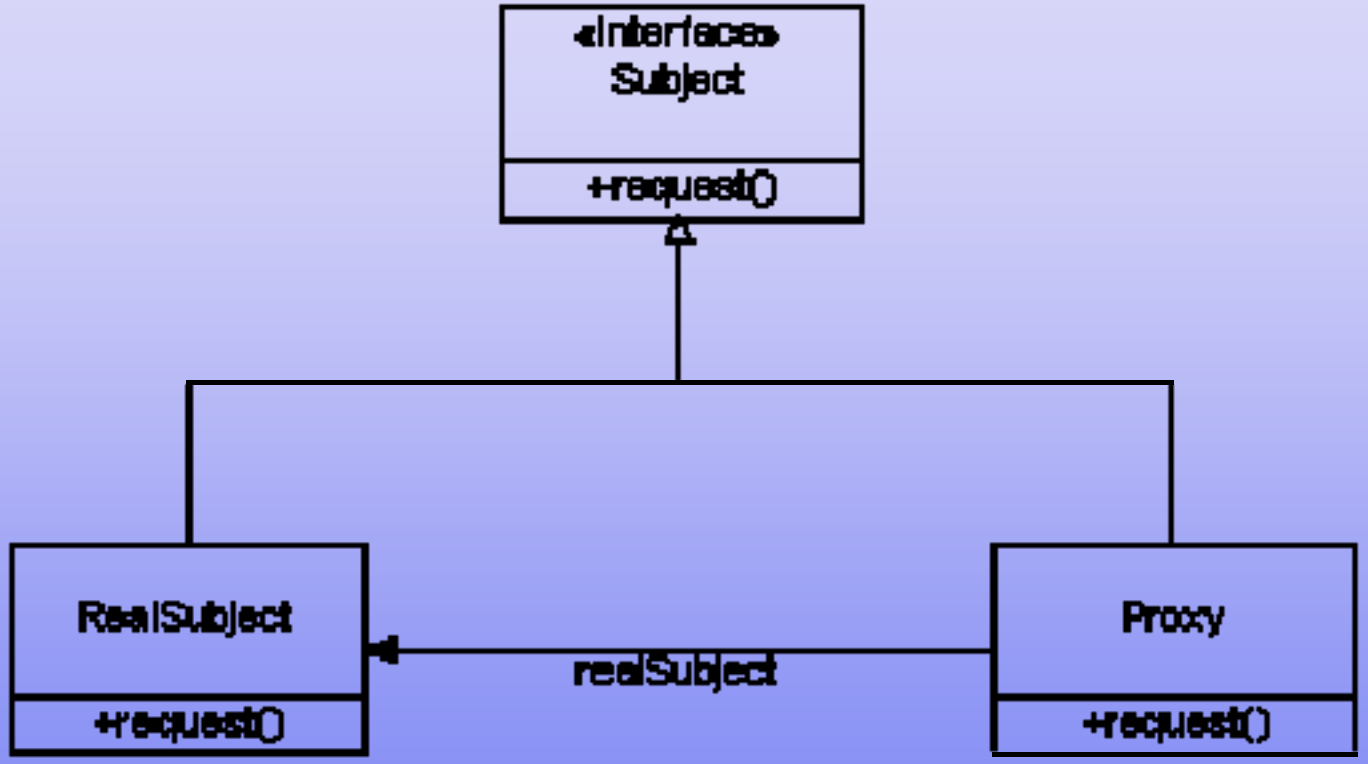
Локализиране на отдалечените обекти

- Това става с вградения механизъм в RMI - rmiregistry, или чрез прехвърляне на псевдоними в процеса на работа

Комуникация между отдалечените обекти: **stubs** и **skeletons**

- Обектът **stub** представлява локалния представител при клиента - заместник (proxy) на отдалечения обект. В RMI, **stub** на отдалечен обект притежава същия интерфейс като този на отдалечения обект.
- В отдалечената JVM, всеки отдалечен обект има съответстващ **skeleton** (ако всички виртуални машини работят в Java 2 среда, skeletons не са необходими). Обектът skeleton прехвърля заявката към актуалният отдалечен метод.

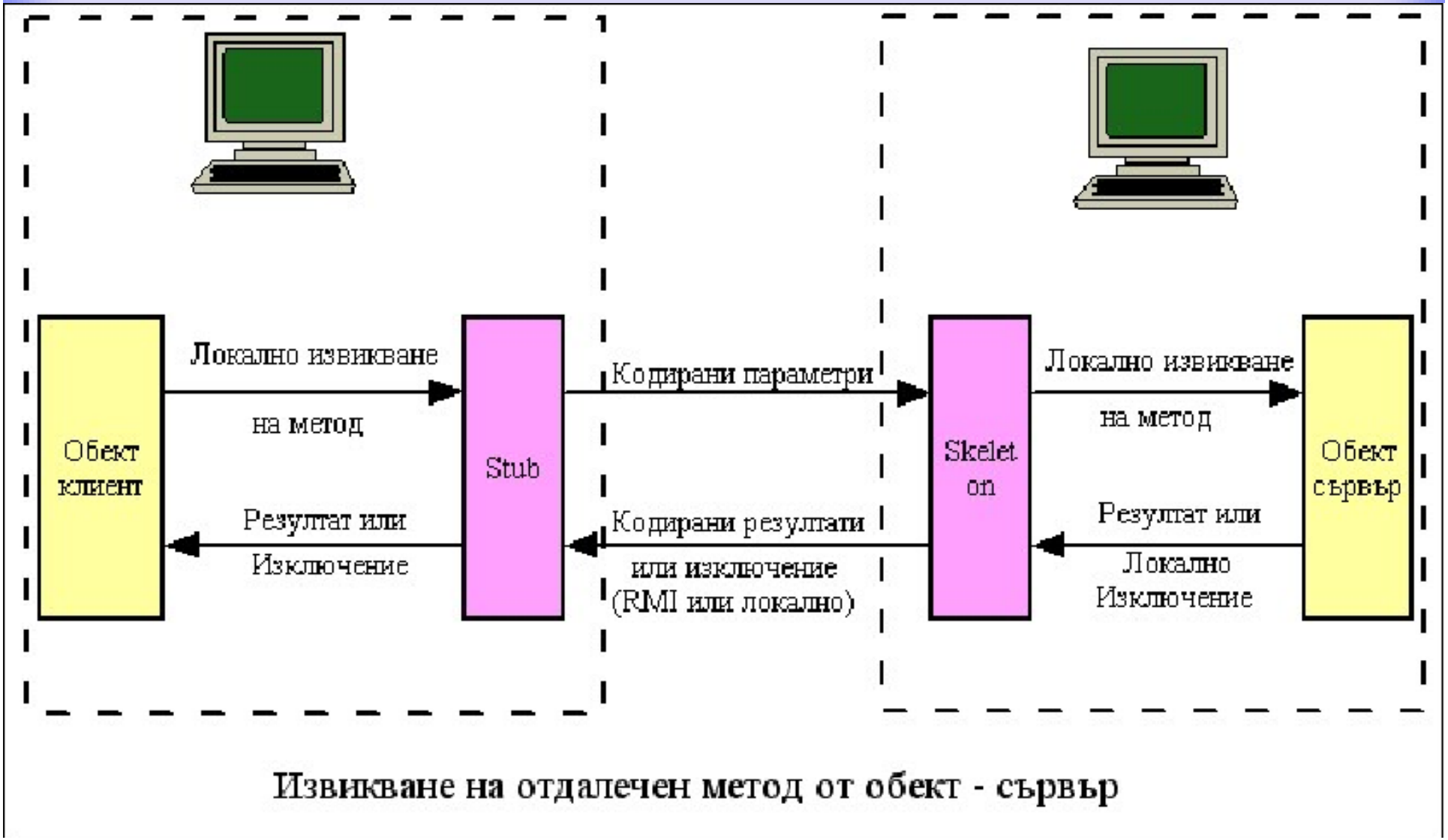
Stub и Skeleton слой



Когато се извиква отдалечен метод, извикването активира метод в обекта **stub**, който

- Инициализира връзка с отдалечената JVM съдържаща отдалечения обект**.**
- Кодира и предава параметрите на отдалечената JVM.
- Изчаква резултата от направеното повикване.
- Декодира получения резултата или изключение.
- Връща получения резултат на извикващия метод.

Пример



Извикване на отдалечен метод от обект - сървър

При получаването на отдалечена заявка, **skeleton:**

- Чете и декодира параметрите на отдалеченото извикване.
- Извиква необходимия локален метод.
- Връща полученият резултата - стойност или изключение.

Remote Reference слой

- Дефинира и поддържа семантиката на RMI връзката.
- Осигурява RemoteRef обект, който представлява линк към отдалечената услуга на имплементирания обект.
- Обектите stub използват метода invoke() в RemoteRef за изпращане на извикването на метода.

Имплементиране

1. Дефиниране на отдалечените интерфейси (интерфейсите на отдалечените обекти).
2. Разработване на отдалечените класове (класове - сървъри), които наследяват отдалечените интерфейси.
3. Разработка на програмата сървър генерираща отдалечените обекти.
4. Разработка на програмата - клиент.
5. Компилиране на класовете.
6. Генериране на stubs и евентуално skeletons.
7. Стартиране на RMI registry.
8. Стартиране на отдалечените обекти.
9. Стартиране на клиента.

Интерфейс

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

Имплементация

```
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;
}
public long sub(long a, long b) throws
            java.rmi.RemoteException {
    return a - b;
}
}
```

Stubs и Skeletons

- `rmic CalculatorImpl`
  - Това създава *CalculatorImpl_Stub.class* и *CalculatorImpl_Skeleton.class*
  - За клиентската машина е са необходими класовете: *Calculator.class*, *CalculatorClient.class* и *CalculatorImpl_Stub.class*
  - За сървъра са необходими: *Calculator.class*, *CalculatorServer.class* и *CalculatorImpl_Skeleton.class*

Host Server

```
import java.rmi.Naming;
public class CalculatorServer {
    public CalculatorServer(){
      try {
       Calculator c = new CalculatorImpl();
       Naming.rebind("rmi://localhost:1099/”+
                              “CalculatorService", c);
      } catch (Exception e) {
         System.out.println("Trouble: " + e);
      }
    }
    public static void main(String args[]) {
          new CalculatorServer();
}
}
```

Client

```
public class CalculatorClient {
     public static void main(String[] args)
{
        try {
           Calculator c = (Calculator)
           Naming.lookup("rmi://localhost
/CalculatorService");
           System.out.println( c.sub(4, 3) );
           System.out.println( c.add(4, 5) );
        } catch (Exception e) {
           System.out.println(e);
        }
     }
}
```

Клас *Naming*

- Класът ***Naming*** наследява ***java.lang.Object*** и осигурява методи за съхраняване и получаване на псевдоними на обекти в регистъра на отдалечените обекти. Всеки метод от класа има като аргумент "name" от тип ***java.lang.String*** в URL формат във вида

  //host:port/name

където

- ***host*** е машината, където се намира регистърът
- ***port*** е номерът на порта, на който регистърът слуша мрежата
- name е обикновен обект от ***java.lang.String***, който задава името на обекта с който той се закачва.
- Ако се пропусне ***host***, се подразбира локалната машина, а ако се пропусне ***port***, се използва подразбиращият се 1099.

Методи

- bind(String name, Remote obj) - Регистрира обекта с зададеното име;
- list(String name) - Връща масив от регистрираните имена
- lookup(String name) - Връща stub на отдалечения обект асоцииран с името
- rebind(String name, Remote obj) - Пререгистрира името на друг обект
- unbind(String name) - Премахва името от регистъра

Ресурси по темата

- http://refg.tu-sofia.bg/AdvJava2/RMI/BgDistObjects.html
- http://java.sun.com/docs/books/tutorial/rmi/index.html
- http://www.javacoffeebreak.com/articles/javarmi/javarmi.html