Remote Method Invocation

Лекция 17

УНЦ «Инфоком»
Самара
2013

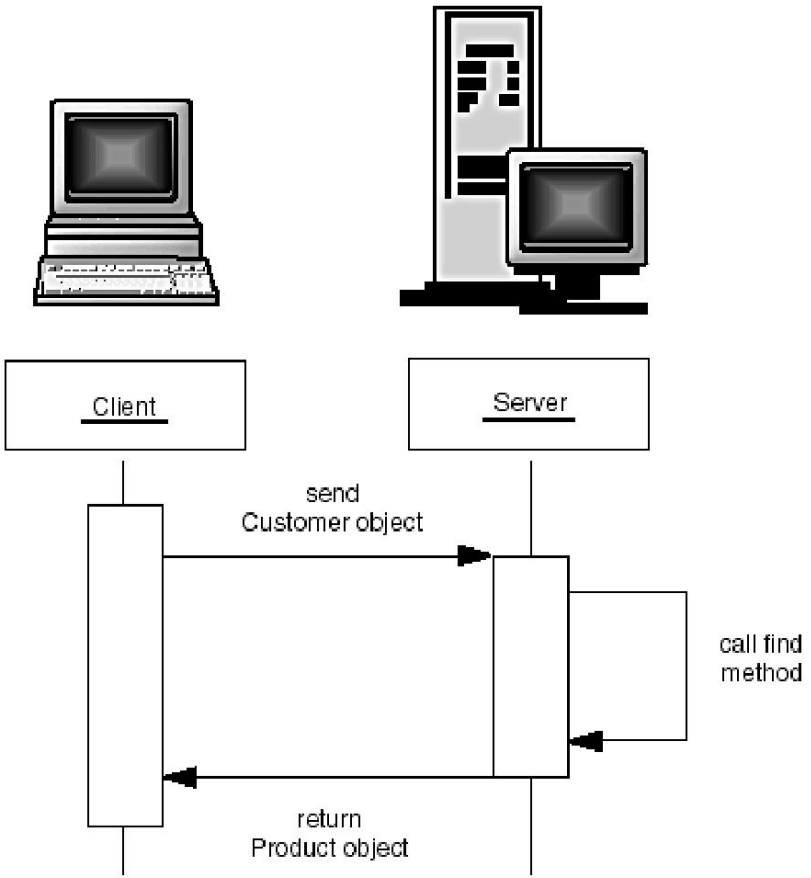
NetCracker®

План лекции

- Общие принципы RMI
- Элементы распределенной системы RMI
- Порядок разработки и запуска RMI-приложений
- Нововведения в Java5

Remote Method Invocation

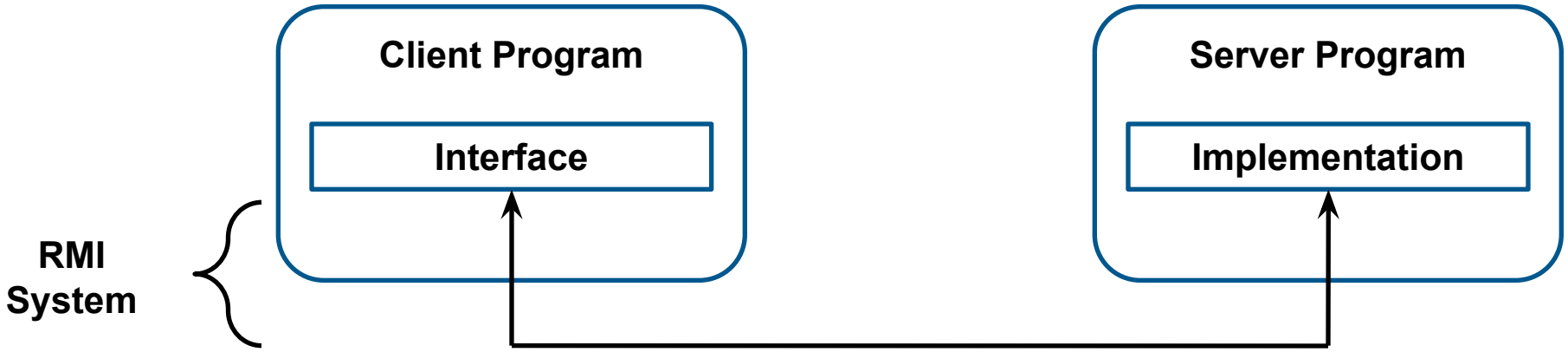
- Основной принцип: **определение** поведения и **реализация** этого поведения считаются разными понятиями
- RMI дает возможность разделить и выполнить на разных JVM код, **определяющий** поведение, и код, **реализующий** поведение



Remote Method Invocation

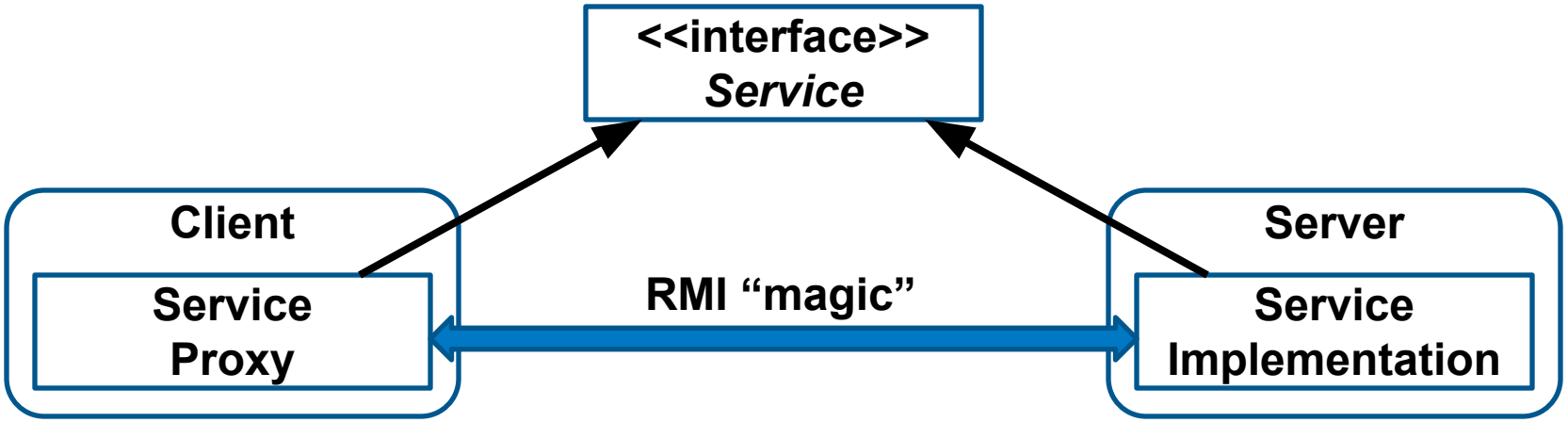
- В RMI удаленная служба определяется при помощи интерфейса Java
- Реализация удаленной службы кодируется в классе, реализующем интерфейс
- Ключ к пониманию RMI:
  - интерфейсы определяют поведение
  - классы определяют реализацию

RMI: принцип действия



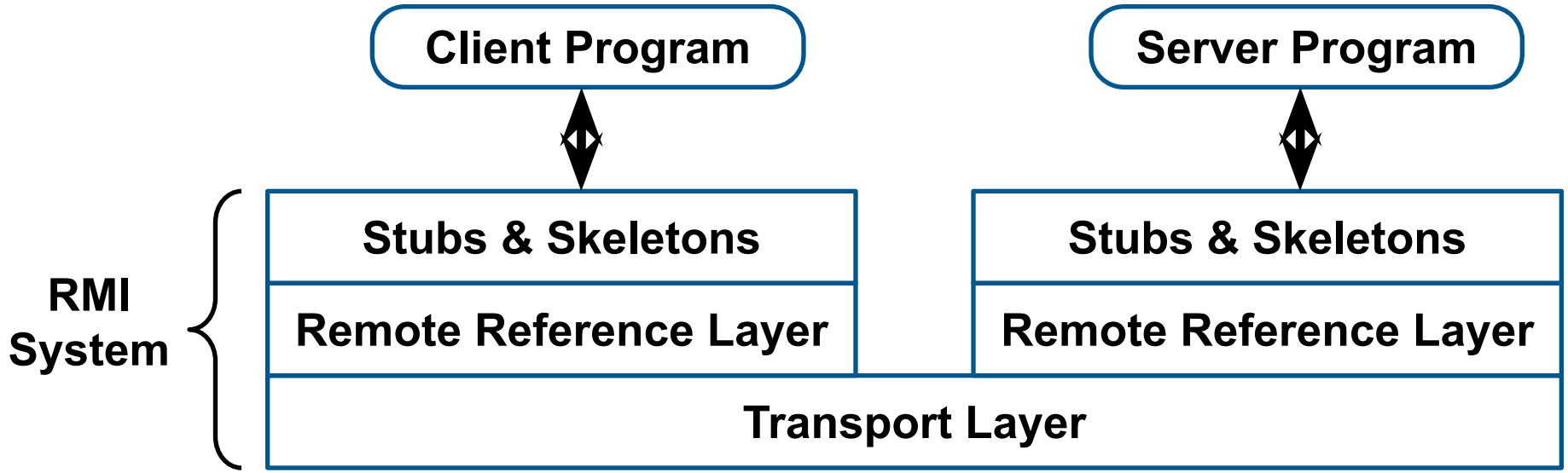
- Интерфейсы Java не содержат исполняемого кода
- RMI поддерживает два класса, реализующих один и тот же интерфейс:
  - первый класс является реализацией поведения и исполняется на сервере
  - второй класс работает как промежуточный интерфейс для удаленной службы и исполняется на клиентской машине

RMI: принцип действия



- Клиентская программа вызывает методы прокси-объекта, RMI передает запрос на удаленную JVM и направляет его в реализацию объекта
- Любые возвращаемые из реализации значения передаются назад в прокси-объект и затем в клиентскую программу

Уровни архитектуры RMI



- Уровень заглушки и скелета
- Уровень удаленной ссылки
- Транспортный уровень

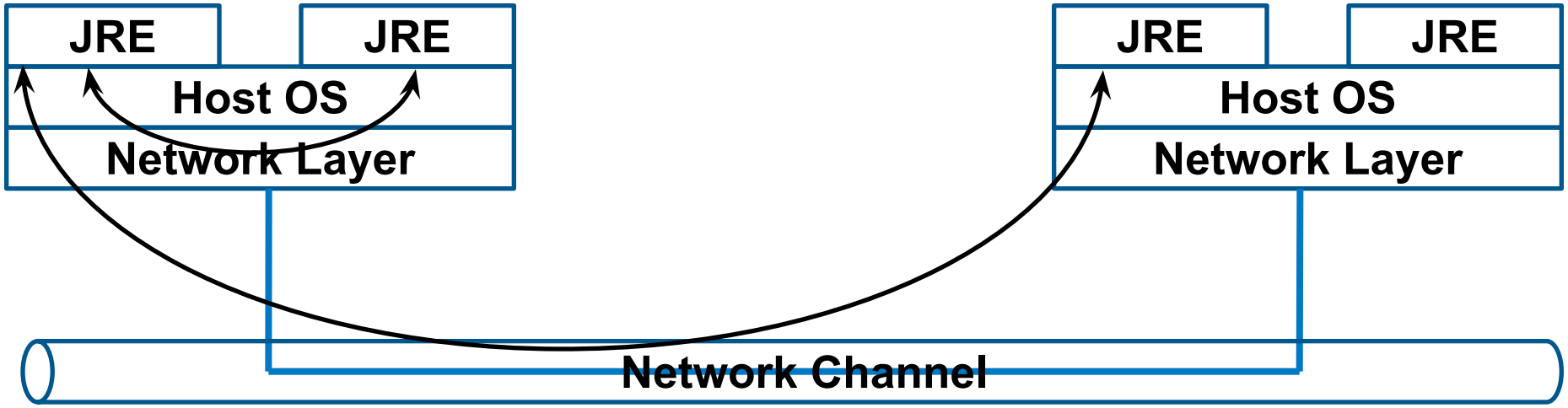
Уровень заглушки и скелета

- Непосредственно с ним взаимодействует разработчик
- Перехватывает вызовы методов, произведенные клиентом при помощи ссылки типа интерфейса, и переадресует их в удаленную службу RMI
- Основан на образце проектирования Proxy (Заместитель)

Уровень удаленной ссылки

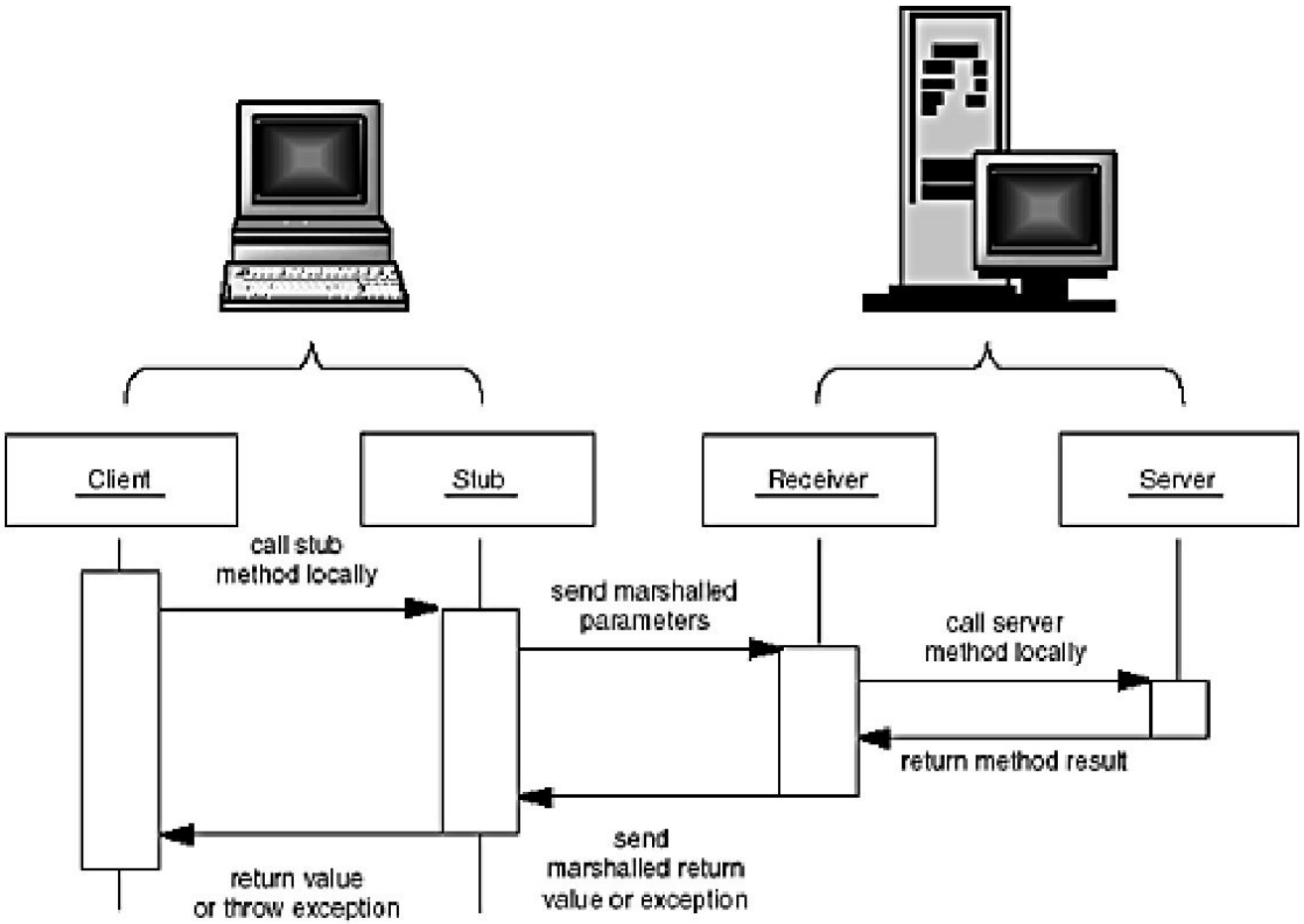
- Удаленная ссылка (remote reference)
  - Может включать в себя адрес компьютера, адрес приложения и адрес собственно объекта
  - Ссылка на удаленный объект должна быть получена в начале работы с этим объектом с помощью **службы именования**
- Этот уровень понимает, как интерпретировать и управлять ссылками на удаленные объекты

Транспортный уровень



- Основан на соединениях TCP/IP между сетевыми машинами
- Обеспечивает основные возможности соединения и некоторые стратегии защиты от несанкционированного доступа
- Поддерживаются протоколы RMI-JRMP и RMI-IIOP

Вызов удаленного метода



Действия при вызове удаленного метода

Заглушка

- Высылает серверу пакет с идентификатором удаленного объекта, описанием вызываемого метода и упакованными параметрами
- Получает пакет от сервера, распаковывает результат

Получатель

- Разбирает параметры (unmarshaling)
- Находит объект
- Вызывает нужный метод
- Получает и упаковывает результат (marshaling)
- Отсылает пакет заглушке

Передача параметров

Аргументы методов и возвращаемое значение могут быть следующих типов:

- **Простые типы**
- **Объектные типы**
- **Удаленные объектные типы**

Параметры простых типов

- Когда в качестве параметра в удаленный метод передается простой тип данных, RMI передает их по значению
- RMI делает копию значения простого типа и передает ее в удаленный метод
- Если метод возвращает значение простого типа, также используется передача по значению
- Значения передаются между JVM в стандартном, машинно-независимом формате; это позволяет JVM, работающим на разных платформах, надежно взаимодействовать друг с другом

Параметры объектных типов

- RMI передает между JVM сам объект, а не ссылку на него, т.е. объект передается по значению
- Когда удаленный метод возвращает объект, в вызывающую программу передается копия объекта
- Для передачи состояния объекта RMI использует сериализацию: состояние объекта преобразуется в набор байтов, пересылаемых по сети

Параметры удаленных объектных типов

- При передаче в качестве параметра или возвращаемого значения ссылки на заглушку удаленного объекта сериализация не используется
- Вместо этого передается удаленная ссылка
- Получатель получает для работы локальную ссылку на заглушку удаленного объекта
- Это еще один способ получить ссылку на удаленный объект

Синтаксис вызова

- Синтаксис вызова такой же, как и при локальном вызове

```
centralWarehouse.getQuantity("SuperSucker 100 Vacuum Cleaner”);
```

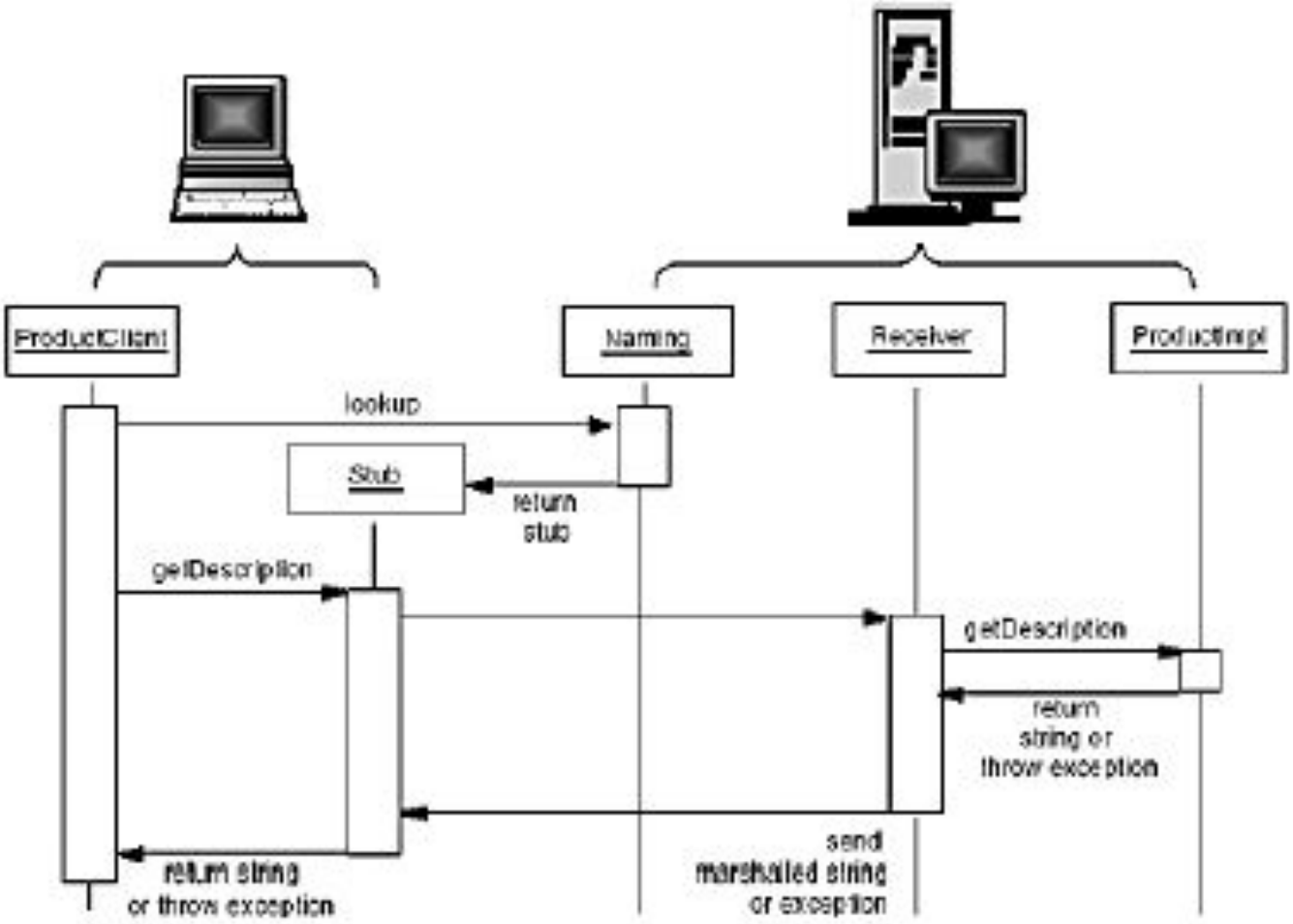
- Используются ссылки интерфейсных типов

```
interface Warehouse extends Remote {
  int getQuantity(String description) throws RemoteException;
  ...
}
```

Динамическая загрузка классов

- Класс заглушки должен быть доступен клиенту
- RMI-клиенты могут сами динамически загружать классы заглушек
- Также могут быть загружены дополнительные классы, необходимые для передачи параметров
- Для обеспечения корректности применяется менеджер безопасности (security manager)

Пример работы



Именование удаленных объектов

- Как клиент находит удаленный объект RMI?
- **Клиенты находят удаленные объекты, используя службу имен или каталогов**
- RMI может использовать различные службы, включая Java Naming and Directory Interface (JNDI)
- RMI включает в себя простую службу – реестр RMI (rmiregistry)
- Реестр RMI работает на каждой машине, содержащей объекты удаленных служб и принимающей запросы на обслуживание (по умолчанию используется порт 1099)

На стороне сервера

- Программа сервера создает удаленный объект, создавая локальный объект, реализующий нужную функциональность
- Затем программа экспортирует этот объект в RMI
- Как только объект экспортирован, RMI создает службу прослушивания, ожидающую соединения с клиентом и запроса к объекту
- После экспорта сервер регистрирует объект в реестре RMI, используя публичное имя

На стороне клиента

- Доступ к реестру RMI обеспечивается через статический класс **`Naming`**
- Он предоставляет метод **`lookup()`**, который клиент использует для запросов к реестру
- Метод принимает URL, указывающий на имя хоста и имя требуемой службы
- URL принимает следующий вид: rmi://<host_name> [:<name_service_port>] /<service_name>
- Метод возвращает удаленную ссылку на объект

Основные элементы распределенной RMI-системы

- Интерфейс удаленного объекта
- Класс, реализующий удаленный объект
- Файлы классов stub'a и skeleton'a.
- Программа серверной части
- Служба именования RMI
- Провайдер файлов классов (HTTP- или FTP- сервер)
- Программа-клиент

Соглашения именования классов

Без суффикса `Product`	Удаленный интерфейс
Суффикс Impl `ProductImpl`	Серверный класс, реализующий интерфейс
Суффикс Server `ProductServer`	Программа на сервере, создающая и регистрирующая объекты
Суффикс Client `ProductClient`	Клиентская программа, вызывающая удаленные методы
Суффикс _Stub `ProductImpl_Stub`	Класс заглушки, автоматически генерируется программой rmic
Суффикс _Skel `ProductImpl_Skel`	Класс скелета, автоматически генерируется программой rmic; нужен для RMI 1.1

Порядок разработки серверной части

- Определение интерфейса удаленного объекта
- Написание класса, реализующего этот интерфейс
- Создание программы серверной части, которая реально создает объект и регистрирует его
- Запуск специального компилятора (`rmic`), автоматически создающего код для заглушки

Описание интерфейса Product

```
import java.rmi.*;

/**
 * The interface for remote product objects.
 */
 public interface Product extends Remote
 {
  /**
   * Gets the description of this product.
   * @return the product description
   */
  String getDescription() throws RemoteException;
}
```

Реализация интерфейса ProductImpl

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

public class ProductImpl extends UnicastRemoteObject
                         implements Product {
  public ProductImpl(String n) throws RemoteException {
    name = n;
  }

  public String getDescription() throws RemoteException{
    return "I am a " + name + ". Buy me!";
  }

  private String name;
}
```

Сервер ProductServer

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

public class ProductServer {
  public static void main(String args[]) {
    try {
      System.out.println("Constructing server implementations...");
      ProductImpl p1 = new ProductImpl("Blackwell Toaster");
      ProductImpl p2 = new ProductImpl("ZapXpress Microwave Oven");
      System.out.println("Binding server implementations to registry...");
      Naming.rebind("toaster", p1);
      Naming.rebind("microwave", p2);
      System.out.println ("Waiting for invocations from clients...");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

Порядок работы клиентской части

- Запуск менеджера безопасности (Security Manager)
- Поиск удаленного объекта
- Вызов какого-либо метода удаленного объекта

Клиент ProductClient

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

public class ProductClient {
  public static void main(String[] args) {
    System.setProperty("java.security.policy", "client.policy");
    System.setSecurityManager(new RMISecurityManager());
    String url = "rmi://localhost/";
    // change to "rmi://yourserver.com/"
    try {
      Product c1 = (Product)Naming.lookup(url + "toaster");
      Product c2 = (Product)Naming.lookup(url + "microwave");
      System.out.println(c1.getDescription());
      System.out.println(c2.getDescription());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

Файл политики безопасности

- Определяет права на доступ к различным ресурсам
- Используется менеджером безопасности
- Необходимо любому загружаемому коду с любого места разрешить:
  - Соединяться или принимать соединения по непривилегированным портам (> 1024) с любого хоста
  - Подключаться к порту 80 (HTTP-порт)

```
grant
{
  permission java.net.SocketPermission
  "*:1024-65535", "connect";
  permission java.net.SocketPermission
  "*:80", "connect";
};
```

Разделение кода для распределенного приложения

Server

- Папка, где располагается сервер
- Не должна быть доступна клиенту
- Должна содержать, как минимум, следующие файлы:
  - `ProductServer.class`
  - `ProductImpl.class`
  - `Product.class`
  - `ProductImpl_Stub.class`

Разделение кода для распределенного приложения

Client

- Папка, где располагается клиент
- Должна содержать, как минимум, следующие файлы:
  - `ProductClient.class`
  - `client.policy`
- Если интерфейс удаленного объекта известен заранее, также должна содержать файл:
  - `Product.class`

Разделение кода для распределенного приложения

Download

- Содержит классы, используемые с данного сервера
- Классы из нее могут быть загружены клиентом динамически
- Указывается как значение переменной `java.rmi.server.codebase`
- Должна содержать, как минимум, следующие файлы:
  - `Product.class`
  - `ProductImpl_Stub.class`

Запуск серверной части

- Запуск программы RMIRegistry

```
UNIX: rmiregistry &

Windows: start rmiregistry
```

- Запуск программы сервера удаленного объекта

```
UNIX: java -Djava.rmi.server.codebase=file:classDir/
        ProductServer &

Windows: start java -Djava.rmi.server.codebase=file:classDir/
           ProductServer
```

Запуск клиентской части

- Файл политики безопасности должен быть доступен менеджеру безопасности

- Запуск производится как запуск обычного приложения Java

```
java ProductClient
```

Нововведения Java5

- Стала необязательной компиляция заглушек с помощью rmic
- Расширились возможности службы именования
- Немного изменился подход к регистрации объекта на сервере
- Общие принципы и порядки разработки и работы приложений сохранились

Интерфейс и реализация в стиле Java5

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

public interface Hello extends Remote {
  String sayHello() throws RemoteException;
}
```

```
public class HelloImpl implements Hello {
  public HelloImpl() {}

  public String sayHello() {
    return "Hello, world!";
  }
}
```

Сервер в стиле Java5

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

public class HelloServer {
  public static void main(String args[]) {
    try {
      HelloImpl obj = new HelloImpl();
      Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
      LocateRegistry.getRegistry();
      registry.bind("Hello", stub);
      System.err.println("Server ready");
    }
    catch (Exception e) {
      System.err.println("Server exception: " + e.toString());
    }
  }
}
```

Спасибо за внимание!

NetCracker®

Дополнительные источники

- Хорстманн, К.С. Java2. Библиотека профессионала. Том 2. Тонкости программирования [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010. – 816 с.
- Grosso, W. Java RMI [Текст] / William Grosso. – O'Reilly, 2001. – 572 с.
- Harold, E.R. Java Network Programming [Текст] / Elliotte Rusty. – O'Reilly, 2004. – 504 с.
- Remote Method Invocation home [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html, дата доступа: 21.10.2011.
- Trial: RMI [Электронный ресурс]. – Режим доступа: http://download.oracle.com/javase/tutorial/rmi/index.html, дата доступа: 21.10.2011.
- jGuru: Remote Method Invocation (RMI) [Электронный ресурс]. – Режим доступа: http://java.sun.com/developer/onlineTraining/rmi/RMI.html, дата доступа: 21.10.2011.