Case Study: Java/RMI

Outline

- Goals of RMI
- The Java (RMI) Object Model
- Interface Definitions in Java RMI
- RMI Architecture
  - Presentation Layer Implementation
  - Session Layer Implementation

Goals of RMI

- *In Java 1.0 object communication confined to objects in one Virtual Machine*
- *Remote Method Invocation (RMI) supports communication between different VMs, potentially across the network*
- *Provide tight integration with Java*
- *Minimize changes to Java language/VM*
- *Work in homogeneous environment*

Java Object Model

- *Interfaces and Remote Objects*
- *Classes*
- *Attributes*
- *Operations*
- *Exceptions*
- *Inheritance*

Java Interfaces and Remote Objects

- *Java already includes the concept of interfaces*
- *RMI does not have a separate interface definition language*
- *Pre-defined interface* `Remote`
- ***Remote interfaces** extend* `Remote`
- ***Remote classes** implement remote interfaces*
- ***Remote objects** are instances of remote classes*

Java Remote Interface Example

Interface name *Declare it as remote*

```
interface Team extends Remote {
public:
 String name() throws RemoteException;
 Trainer[] coached_by() throws RemoteException;
 Club belongs_to() throws RemoteException;
 Players[] players() throws RemoteException;
 void bookGoalies(Date d) throws RemoteException;
 void print() throws RemoteException;
};
```

Remote operations

Attributes

- *RMI does not attributes*
- *Attributes must be represented as set and get operations by the designer*
- *Example:*

```
interface Club extends Organization, Remote {
public:
 int noOfMembers() throws RemoteException;
 Address location() throws RemoteException;
 Team[] teams() throws RemoteException;
 Trainer[] trainers() throws RemoteException;
 ...
};
```

Attribute get operations

Combining Classes and Remote Interfaces

```
interface Organization {
 private:
  String name() RemoteException;
};
class Address {
 public:
  String street;
  String postcode;
  String city;
};
interface Club extends Organization, Remote {
 public:
  int noOfMembers() throws RemoteException;
  Address location() throws RemoteException;
  Team[] teams() throws RemoteException;
  Trainer[] trainers() throws RemoteException;
  void transfer(Player p) throws RemoteException;
};
```

Club can return an address object

Club makes name() remotely accessible

Parameter Passing

- *Atomic types are passed by value*
- *Remote objects are passed by reference*
- *Non-Remote objects are passed by value*

```
class Address {
 public:
  String street;
  String postcode;
  String city;
};
interface Club extends Organization, Remote {
 public:
  Address location() throws RemoteException;
  ...
};
```

returns a copy of the address!

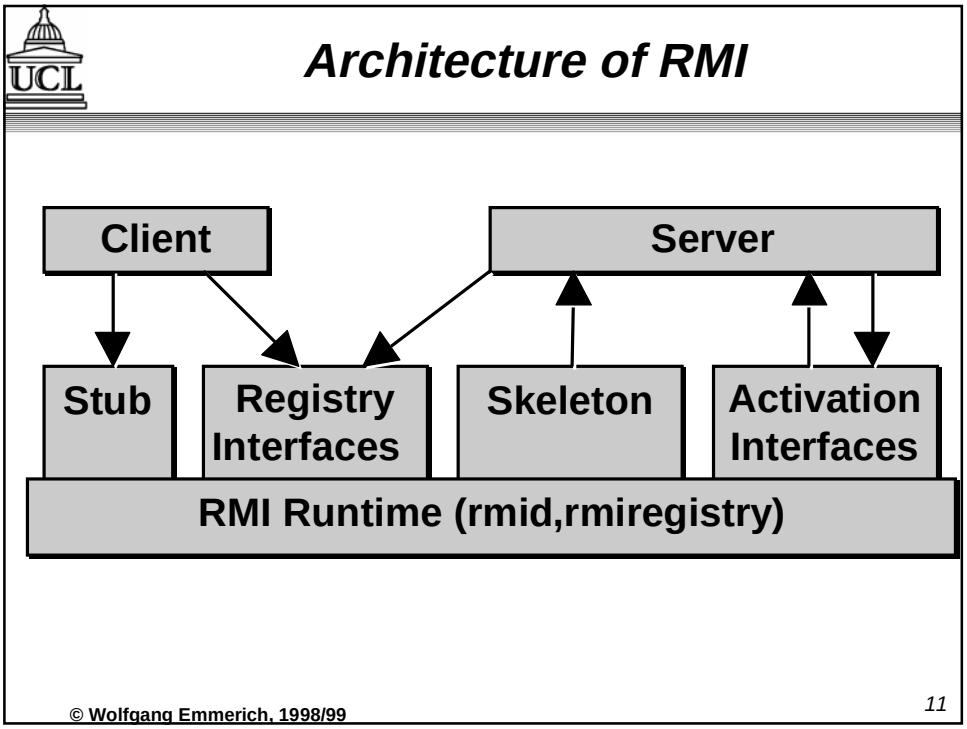
Exception

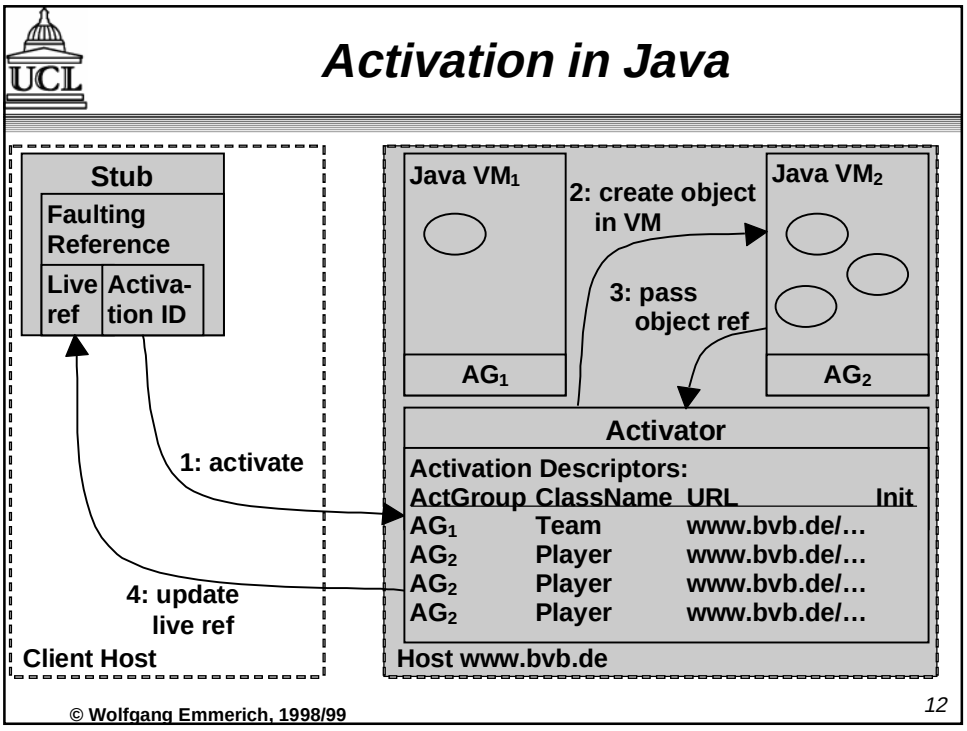
- *Pre-Defined Exception* ***`RemoteException`***
- *Type-Specific Exceptions*
- *Example:*

Type-specific Exception

```
class PlayerBooked extends Exception {};
interface Team extends Remote {
 public:
  ...
  void bookGoalies(Date d) throws
              RemoteException, PlayerBooked;
  ...
};
```

Operation declares that it may raise it


Architecture of RMI
Client
Server
Stub
Registry Interfaces
Skeleton
Activation Interfaces
RMI Runtime (rmid,rmiregistry)
© Wolfgang Emmerich, 1998/99
11


Activation in Java
Stub
Faulting Reference
Live ref
Activa-tion ID
Client Host
1: activate
4: update live ref
Java VM1
AG1
2: create object in VM
3: pass object ref
Java VM2
AG2
Activator
Activation Descriptors:
ActGroup ClassName URL Init
AG1 Team www.bvb.de/…
AG2 Player www.bvb.de/…
AG2 Player www.bvb.de/…
AG2 Player www.bvb.de/…
Host www.bvb.de
© Wolfgang Emmerich, 1998/99
12