

Verteilte Systeme – Übung

Java Remote Method Invocation

Sommersemester 2025

Harald Böhm, Christian Berger, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl für Informatik 4 (Systemsoftware)

<https://sys.cs.fau.de>



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät

Java Remote Method Invocation

Beispiel

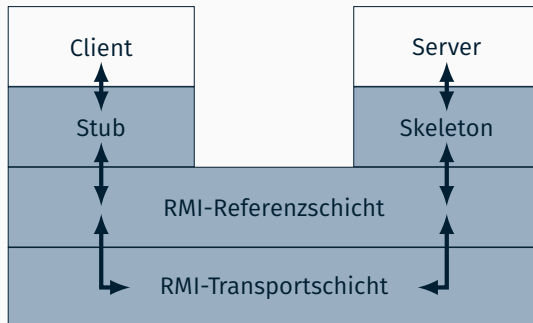
Java Remote Method Invocation

■ Remote Method Invocation (RMI)

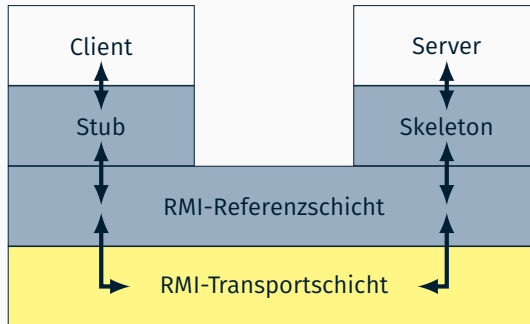
- Aufrufe von Methoden an Objekten auf anderen Rechnern
- *Remote-Referenz*: Transparente Objektreferenz zu entferntem Objekt

```
// Lokaler Aufruf  
localReference.method();  
// Fernaufruf  
remoteReference.method();
```

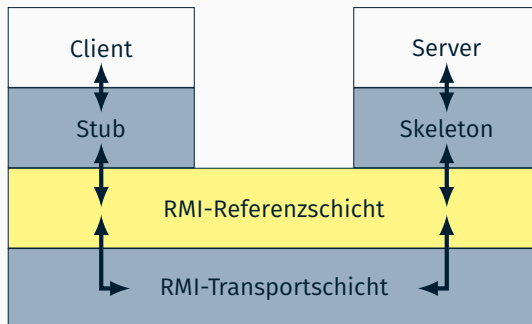
■ Beispiel: Java RMI



- Datenübertragung zwischen Rechnern
- Implementierung
 - Aktueller Standard: Verwendung von TCP/IP-Sockets
 - Generell: Verschiedene Transportmechanismen denkbar



- Verwaltung von Remote-Referenzen
- Implementierung der Aufrufsemantik (Beispiele)
 - Unicast, Punkt-zu-Punkt
 - Strategien zum Wiederaufbau der Verbindung nach einer Unterbrechung

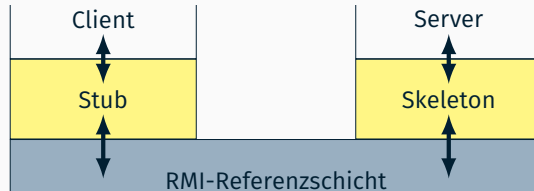


■ Stub

1. erhält einen Objekt-Ausgabe-Strom von der RMI-Referenzschicht
2. schreibt die Parameter in diesen Strom
3. weist die RMI-Referenzschicht an, die Methode aufzurufen
4. holt einen Objekt-Eingabe-Strom von der RMI-Referenzschicht
5. liest das Rückgabe-Objekt aus diesem Strom
6. liefert das Rückgabe-Objekt an den Aufrufer

■ Skeleton

1. erhält einen Objekt-Eingabe-Strom von der RMI-Referenzschicht
2. liest die Parameter aus diesem Strom
3. ruft die Methode am implementierten Objekt auf
4. holt einen Objekt-Ausgabe-Strom von der RMI-Referenzschicht
5. schreibt das Rückgabe-Objekt in diesen Strom



■ Remote-Objekt (entferntes Objekt)

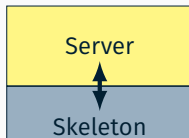
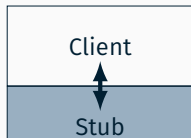
- Kann aus einer anderen Java Virtual Machine heraus genutzt werden
- Erst von außerhalb erreichbar, nachdem es **exportiert** wurde

■ Remote-Schnittstelle

- Beschreibt die per Fernaufruf erreichbaren Methoden des Objekts
- Abgeleitet von `java.rmi.Remote` (Marker-Schnittstelle)
- Einzige Möglichkeit mit Java RMI auf ein entferntes Objekt zuzugreifen

■ Remote-Exception (`java.rmi.RemoteException`)

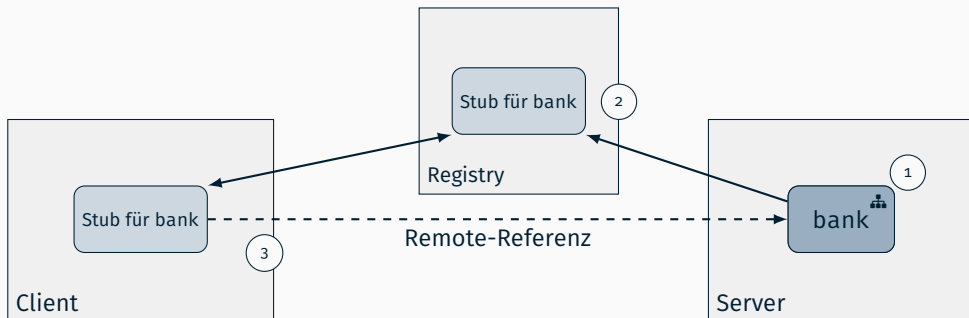
- Muss im throws-Clause jeder Remote-Methode angegeben sein
- Beim Auftreten einer Remote-Exception weiß der Aufrufer nicht, ob die Methode komplett, teilweise oder gar nicht ausgeführt wurde



Java Remote Method Invocation

Beispiel

1. Exportieren
 - Lokales Objekt als Remote-Objekt exportieren
2. Bekannt machen
 - Remote-Objekt über eine Registry bekannt machen
 - (**oder**) Remote-Objekt direkt verschicken
3. Ausführen
 - Fernaufruf auf Remote-Referenz ausführen



■ Geldbetrag VSMoney

```
public class VSMoney implements Serializable {  
    private float amount;  
  
    public VSMoney(float amount) {  
        this.amount = amount;  
    }  
  
    public float getAmount() { return amount; }  
}
```

■ Konto VSAccount (Remote-Schnittstelle)

```
public interface VSAccount extends Remote {  
    public void deposit(VSMoney money) throws RemoteException;  
}
```

■ Bank VSBank (Remote-Schnittstelle)

```
public interface VSBank extends Remote {  
    public void deposit(VSMoney money, VSAccount account) throws RemoteException;  
}
```

- VSBankImpl: Implementierung der Remote-Schnittstelle VSBank
- Exportieren des Remote-Objekts
 - Implizit: Unterklasse von `java.rmi.server.UnicastRemoteObject`

```
public class VSBankImpl extends UnicastRemoteObject implements VSBank {  
    // Konstruktor  
    public VSBankImpl() throws RemoteException { super(); }  
  
    // Implementierung der Remote-Methode  
    public void deposit(VSMoney money, VSAccount account) throws RemoteException {  
        account.deposit(money);  
    }  
}
```

```
VSBank bank = new VSBankImpl();
```

- Explizit: Aufruf von `UnicastRemoteObject.export()`

```
public class VSBankImpl implements VSBank { [...] }
```

```
VSBank b = new VSBankImpl();  
VSBank bank = (VSBank) UnicastRemoteObject.exportObject(b, 0);
```

- Konto-Implementierung VSAccountImpl
 - Implementierung der Remote-Schnittstelle VSAccount
 - Exportieren analog zu VSBankImpl
 - Synchronisation paralleler deposit()-Aufrufe
 - [Auf welchem Rechner erscheint die Bildschirmausgabe?]

```
public class VSAccountImpl implements VSAccount {  
    private float amount;  
  
    public VSAccountImpl(float amount) {  
        this.amount = amount;  
    }  
  
    public synchronized void deposit(VSMoney money) {  
        amount += money.getAmount();  
        System.out.println("New amount: " + amount);  
    }  
}
```

- Namensdienst
 - Bekanntmachen von Remote-Objekten
 - Abbildung von Objektnamen auf Objektreferenzen
- Registry-Schnittstelle

```
public interface Registry extends Remote {  
    public void bind(String name, Remote obj);  
    public Remote lookup(String name);  
    [...]  
}
```

- bind() Zuordnung eines Objekts zu einem eindeutigen Namen
- lookup() Rückgabe der Remote-Referenz zu einem Namen

- Erzeugung und Verbindung zur Registry

```
public class LocateRegistry {  
    public static Registry createRegistry(int port);  
    public static Registry getRegistry(String host, int port);  
    [...]  
}
```

- createRegistry() Erzeugung einer Registry auf dem lokalen Rechner
- getRegistry() Holen einer Remote-Referenz auf eine Registry

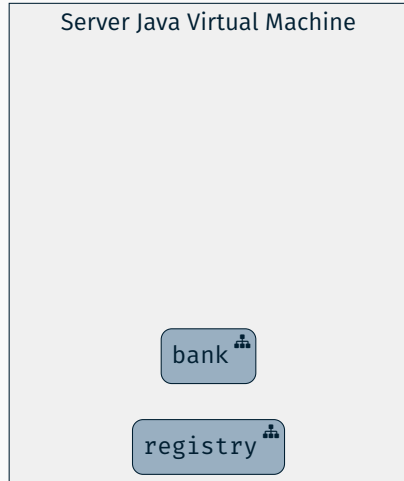
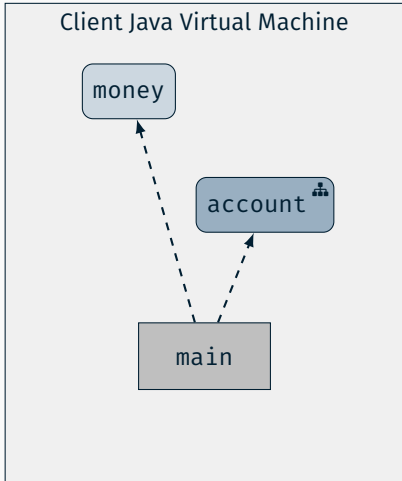
- Server-Implementierung VSBankServer
 - Erzeugen des Remote-Objekts
 - Exportieren des Remote-Objekts
 - Remote-Objekt mittels Registry bekannt machen

```
public class VSBankServer {  
    public static void main(String[] args) throws Exception {  
        // Remote-Objekt erzeugen  
        VSBank bankImpl = new VSBankImpl();  
  
        // Remote-Objekt auf Port 12678 exportieren  
        VSBank bank = (VSBank) UnicastRemoteObject.exportObject(bankImpl, 12678);  
  
        // Remote-Objekt bekannt machen  
        Registry registry = LocateRegistry.createRegistry(12345);  
        registry.bind("bank", bank);  
  
        // Prozess weiterlaufen lassen  
        Thread.sleep(Long.MAX_VALUE);  
    }  
}
```

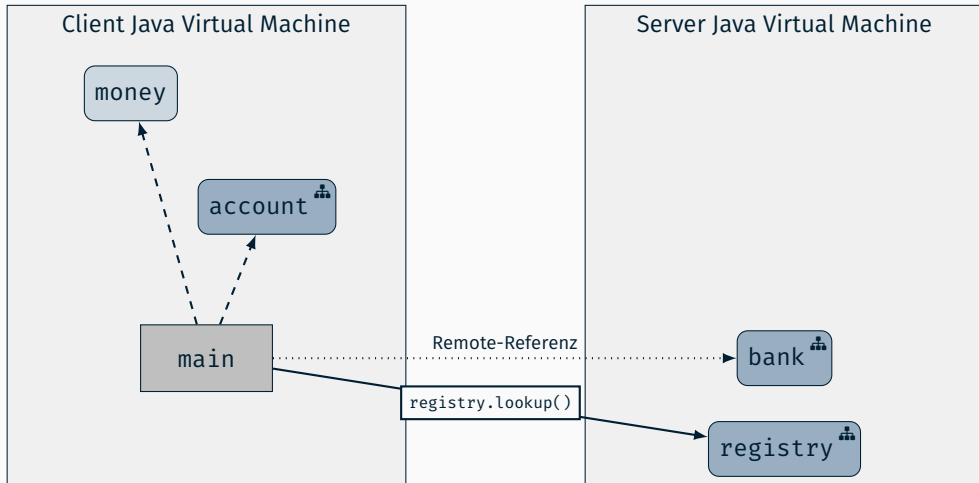
■ Client-Implementierung VSBankClient

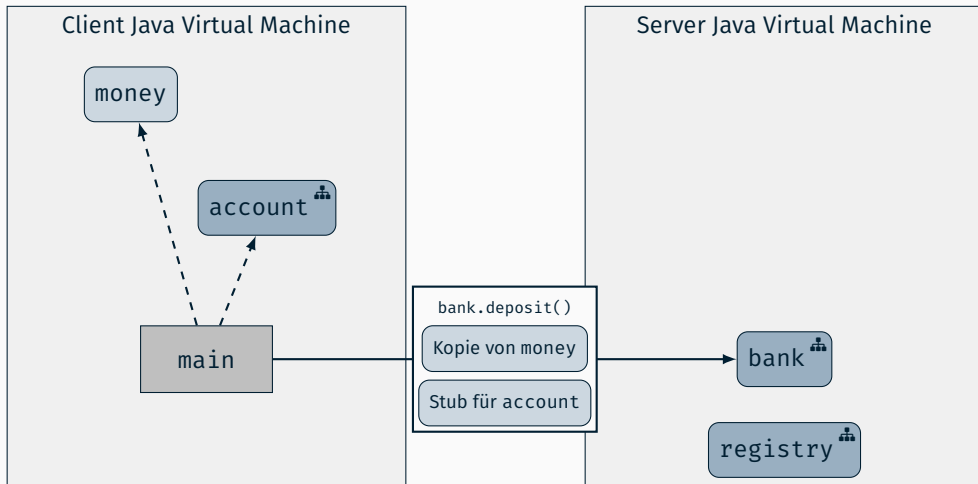
```
public class VSBankClient {  
    public static void main(String[] args) throws Exception {  
        // Geldbetrag-Objekt anlegen  
        VSMoney money = new VSMoney(10.0f);  
  
        // Account anlegen und exportieren  
        VSAccount accountImpl = new VSAccountImpl(100.0f);  
        UnicastRemoteObject.exportObject(accountImpl, 0);  
  
        // Remote-Referenz holen (Annahme: Server auf faui05a)  
        Registry registry = LocateRegistry.getRegistry("faui05a", 12345);  
        VSBank bank = (VSBank) registry.lookup("bank");  
  
        // Geld einzahlen  
        bank.deposit(money, accountImpl);  
  
        // Account freigeben  
        UnicastRemoteObject.unexportObject(accountImpl, true);  
    }  
}
```

- Ausgangssituation vor Registry-Zugriff des Client

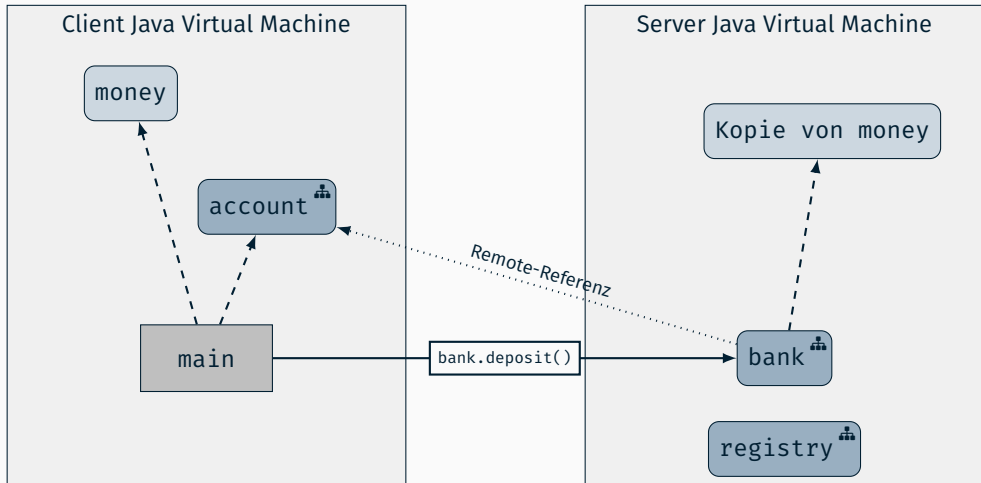


- Remote-Referenz auf bank von Registry holen



■ Methodenaufruf von `bank.deposit()`

- Nach dem Auspacken der Parameter



■ Methodenaufruf von `account.deposit()`