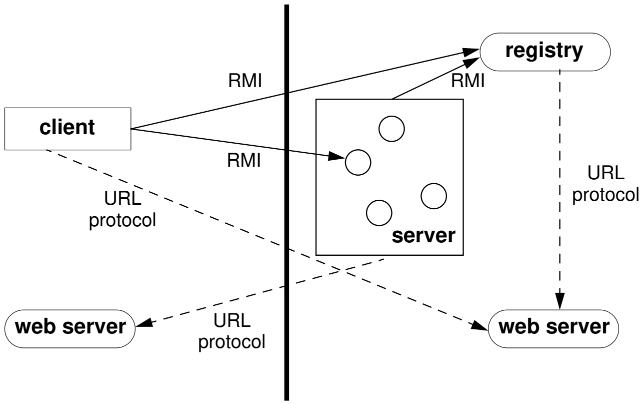
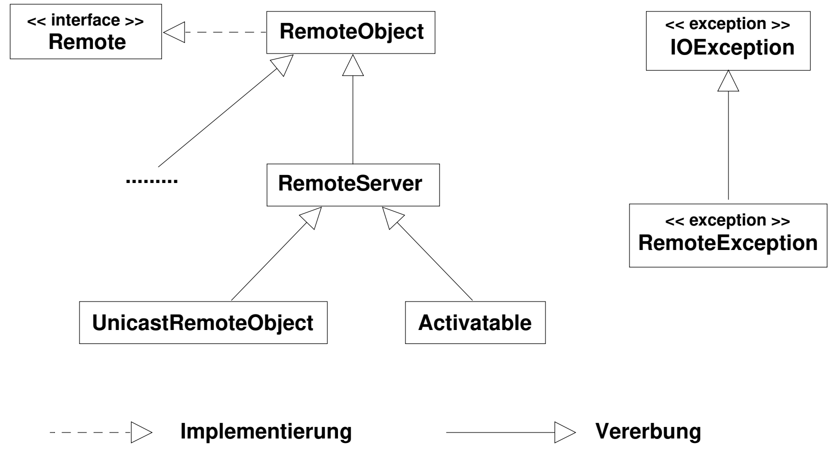
3.5 Java Remote Method Invocation (RMI)

❑ Gesamtstruktur



Klassen und Schnittstellen



Vorgehensweise

1. Definition der fernaufrufbaren Funktionen als Java-Interface
2. Schreiben der Klassen des Dienstleisters
3. Schreiben des Dienstnehmers (als Applet)

Beispiel: Hello World

- Der Dienstleister (server) `HelloImpl` gibt nach Aufruf der Methode `sayHello()` die Zeichenkette `"Hello World"` zurück.
- Der Dienstnehmer `HelloApplet` ruft mittels Fernaufruf die Methode `sayHello()` auf und gibt die zurückerhaltene Zeichenkette aus.

1. Definition der fernaufrufbaren Funktionen als Java-Interface

```
package examples.hello;

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

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

- The remote interface must be declared public. Otherwise, unless a client is in the same package as the remote interface, the client will get an error when attempting to load a remote object that implements the remote interface.
- The remote interface extends the java.rmi.Remote interface.
- Each method must declare java.rmi.RemoteException (or a superclass of RemoteException) in its throws clause, in addition to any application-specific exceptions.
- The data type of any remote object that is passed as an argument or return value (either directly or embedded within a local object) must be declared as the remote interface type (for example, Hello) not the implementation class (HelloImpl).

2. Schreiben der Klassen des Dienstleisters

```
package examples.hello;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
import java.rmi.server.UnicastRemoteObject;

public class HelloImpl extends UnicastRemoteObject
                         implements Hello {

public HelloImpl() throws RemoteException {
   super();
}

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



```
   public static void main(String args[]) {

      // Create and install a security manager
      if (System.getSecurityManager() == null) {
         System.setSecurityManager(new RMISecurityManager());
      }
      try {
         HelloImpl obj = new HelloImpl();
         // Bind this object instance to the name "HelloServer"
         Naming.rebind("///HelloServer", obj);
      } catch (Exception e) {
         System.out.println("New HelloImpl err: "
                              + e.getMessage());
         e.printStackTrace();
      }
   }
}
```


3. Schreiben des Dienstnehmers (als Applet)

```
package examples.hello;

import java.applet.Applet;
import java.awt.Graphics;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class HelloApplet extends Applet {
   String message = "blank";

   // "obj" is the identifier that we'll use to refer
   // to the remote object that implements the "Hello"
   // interface
   Hello obj = null;
```



```
   public void init() {
      try {
         obj = (Hello)Naming.lookup("//"
                              + getCodeBase().getHost()
                              + "/HelloServer");
         message = obj.sayHello();
      } catch (Exception e) {
         System.out.println("HelloApplet exception: "
                              + e.getMessage());
         e.printStackTrace();
      }
   }

   public void paint(Graphics g) {
      g.drawString(message, 25, 50);
   }
}
```

Fernaufruf: Java Remote Method Invocation

- Systemerzeugung
  - Kompilieren der drei Klassen
  - Mittels rmic Stub und Skeleton erzeugen
  - Bestandteile verteilen
  - Am dienstleistenden Rechner
    - Registratur starten
    - Dienstleister (server) starten
  - Am dienstnehmenden Rechner
    - Applet starten

Fernaufruf: Stub class generated by rmic

```
// Stub class generated by rmic, do not edit.
// Contents subject to change without notice.

package examples.hello;

public final class HelloImpl_Stub
    extends java.rmi.server.RemoteStub
    implements examples.hello.Hello, java.rmi.Remote
{
   private static final java.rmi.server.Operation[] operations = {
      new java.rmi.server.Operation("java.lang.String sayHello()")
   };

   private static final long interfaceHash = 6486744599627128933L;

   private static final long serialVersionUID = 2;

   private static boolean useNewInvoke;
   private static java.lang.reflect.Method $method_sayHello_0;
```


Fernaufruf: Stub class generated by rmic

```
static {
   try {
      java.rmi.server.RemoteRef.class.getMethod("invoke",
         new java.lang.Class[] {
            java.rmi.Remote.class,
            java.lang.reflect.Method.class,
            java.lang.Object[].class,
            long.class
         });
      useNewInvoke = true;
      $method_sayHello_0 =
         examples.hello.Hello.class.getMethod("sayHello",
            new java.lang.Class[] {});
   } catch (java.lang.NoSuchMethodException e) {
      useNewInvoke = false;
   }
}
```


Fernaufruf: Stub class generated by rmic

```
   // constructors
   public HelloImpl_Stub() {
      super();
   }

   public HelloImpl_Stub(java.rmi.server.RemoteRef ref) {
      super(ref);
   }
```

Fernaufruf: Stub class generated by rmic

```
// methods from remote interfaces

// implementation of sayHello()
public java.lang.String sayHello()
   throws java.rmi.RemoteException
{
   try {
      if (useNewInvoke) {
         Object $result
            = ref.invoke(this, $method_sayHello_0,
                         null, 6043973830760146143L);
         return ((java.lang.String) $result);
      } else {
         java.rmi.server.RemoteCall call
            = ref.newCall((java.rmi.server.RemoteObject) this,
                          operations, 0, interfaceHash);
         ref.invoke(call);
         java.lang.String $result;
         try {
            java.io.ObjectInput in = call.getInputStream();
            $result = (java.lang.String) in.readObject();
```


Fernaufruf: Stub class generated by rmic

```
            } catch (java.io.IOException e) {
               throw new java.rmi
                  .UnmarshalException
                           ("error unmarshalling return", e);
            } catch (java.lang.ClassNotFoundException e) {
               throw new java.rmi
                  .UnmarshalException
                           ("error unmarshalling return", e);
            } finally {
               ref.done(call);
            }
            return $result;
         }
      } catch (java.lang.RuntimeException e) { throw e;
      } catch (java.rmi.RemoteException e) { throw e;
      } catch (java.lang.Exception e) {
         throw new java.rmi
            .UnexpectedException("undeclared checked exception", e);
      }
   }
}
```


Fernaufruf: Skeleton class generated by rmic

```
// Skeleton class generated by rmic, do not edit.
// Contents subject to change without notice.

package examples.hello;

public final class HelloImpl_Skel
   implements java.rmi.server.Skeleton
{
   private static final java.rmi.server.Operation[] operations = {
      new java.rmi.server.Operation("java.lang.String sayHello()")
   };

   private static final long interfaceHash = 6486744599627128933L;

   public java.rmi.server.Operation[] getOperations() {
      return (java.rmi.server.Operation[]) operations.clone();
   }
```


Fernaufruf: Skeleton class generated by rmic

```
   public void dispatch(java.rmi.Remote obj,
                        java.rmi.server.RemoteCall call,
                        int opnum,
                        long hash)
      throws java.lang.Exception
   {
      if (opnum < 0) {
         if (hash == 6043973830760146143L) {
            opnum = 0;
         } else {
            throw new java.rmi
               .UnmarshalException("invalid method hash");
         }
      } else {
         if (hash != interfaceHash)
            throw new java.rmi.server
               .SkeletonMismatchException
                        ("interface hash mismatch");
      }
      examples.hello.HelloImpl server
         = (examples.hello.HelloImpl) obj;
```

```
    switch (opnum) {
      case 0: // sayHello()
        {  call.releaseInputStream();
           java.lang.String $result = server.sayHello();
           try {
              java.io.ObjectOutput out
                   = call.getResultStream(true);
              out.writeObject($result);
           } catch (java.io.IOException e) {
              throw new java.rmi
                 .MarshalException
                         ("error marshalling return", e);
           }
           break;
        }
      default:
        throw new java.rmi
           .UnmarshalException("invalid method number");
    }
  }
}
```