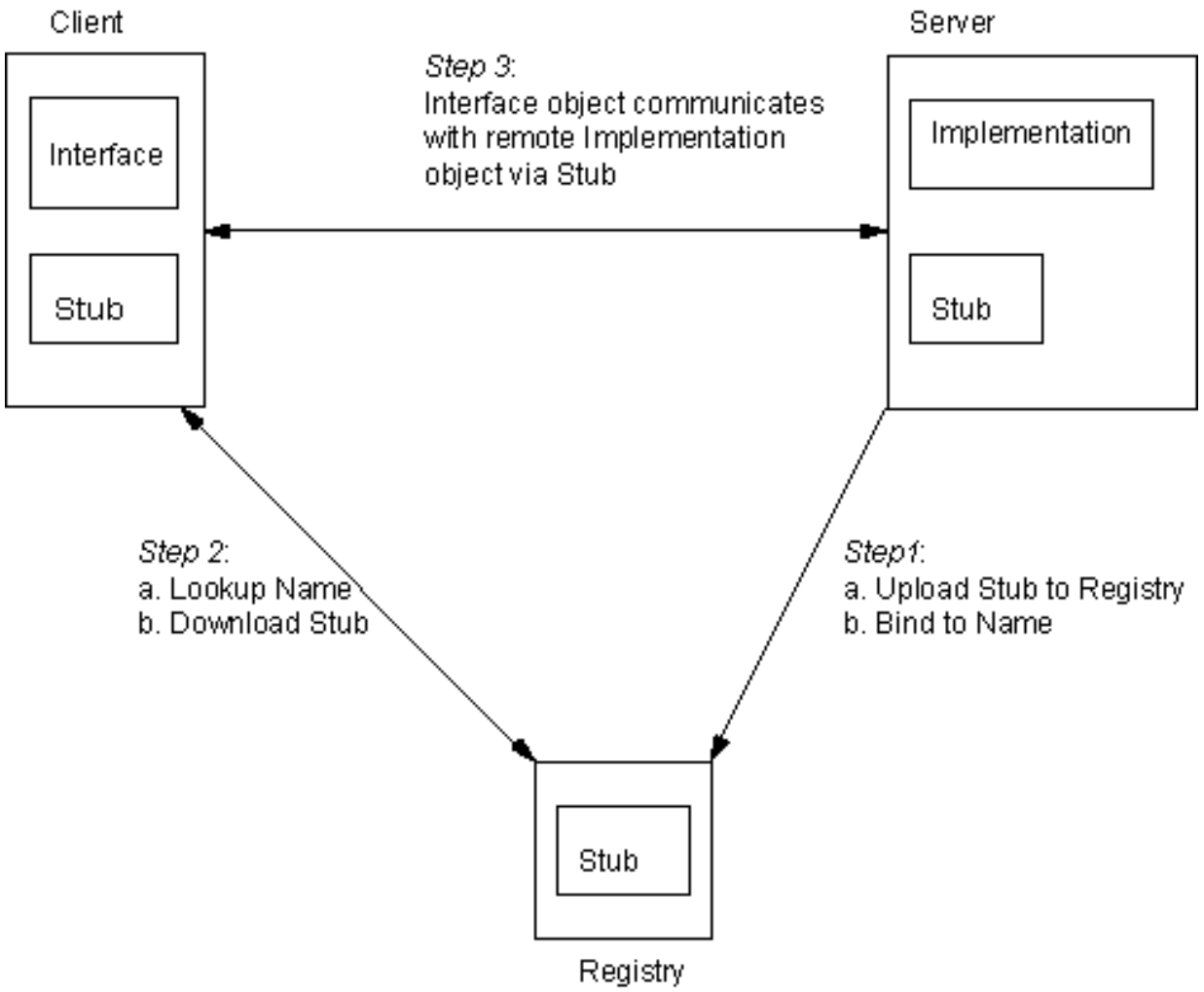
Life is great
VIA University College

Software Development with UML and Java 2

RMI


Client
Server
Interface
Stub
Implementation
Stub
Step 3:
Interface object communicates
with remote Implementation
object via Stub
Step 2:
a. Lookup Name
b. Download Stub
Step1:
a. Upload Stub to Registry
b. Bind to Name
Stub
Registry

RmiServer (in parts) – Registry

```
public class RmiServer extends UnicastRemoteObject
                       implements ServerInterface
{
   public static void main(String[] args) throws RemoteException
   {
      Registry reg = LocateRegistry.createRegistry(1099);
      ServerInterface rmiServer = new RmiServer();

      Naming.rebind("toUpperCase", rmiServer);
      System.out.println("Starting server...");
   }
   public RmiServer() throws RemoteException
   {
      super();
   }
```

RmiServer (in parts) – Registry

```
public class RmiServer extends UnicastRemoteObject
                       implements ServerInterface
{
   public static void main(String[] args) throws RemoteException
   {
      Registry reg = LocateRegistry.createRegistry(1099);
      ServerInterface rmiServer = new RmiServer();

      reg.rebind("toUpperCase", rmiServer);
      System.out.println("Starting server...");
   }
   public RmiServer() throws RemoteException
   {
      super();
   }
```

Create and publish the remote object

1) Create:
   a) Create an object of the class that implements the remote interface (in a main method or in another class), or
   b) Use `this` if publishing is done in the class implementing the remote interface
2) Publish:
   a) The runtime need to create a TCP server socket and start waiting for connecting clients. `UnicastRemoteObject` is used for this purpose
      I. Either extending `UnicastRemoteObject`, or
      II. Calling static method `exportObject` in `UnicastRemoteObject`
   b) Upload the stub to the registry and bind it to a name/string

RmiServer (in parts) - main method

```
public class RmiServer extends UnicastRemoteObject
                       implements ServerInterface
{
   public static void main(String[] args) throws RemoteException
   {
      Registry reg = LocateRegistry.createRegistry(1099);
      ServerInterface rmiServer = new RmiServer();

      Naming.rebind("toUpperCase", rmiServer);
      System.out.println("Starting server...");
   }
   public RmiServer() throws RemoteException
   {
      super();
   }
```

Upload the stub to registry and bind it to a name/string

Publish the object (start listening for clients)

Create an object of the class implementing the remote interface

RmiServer (in parts) - main method

```
public class RmiServer implements ServerInterface
{
    public static void main(String[] args) throws RemoteException
    {
        Registry reg = LocateRegistry.createRegistry(1099);
        ServerInterface rmiServer = new RmiServer();

        UnicastRemoteObject.exportObject(rmiServer, 0);

        Naming.rebind("toUpperCase", rmiServer);
        System.out.println("Starting server...");
    }
    public RmiServer()
    {
    }
```

Upload the stub to registry and bind it to a name/string

Publish the object (start listening for clients)

Create an object of the class implementing the remote interface

RmiServer (in parts) - main method

```
public class RmiServer implements ServerInterface
{
    public static void main(String[] args) throws RemoteException
    {
        Registry reg = LocateRegistry.createRegistry(1099);
        ServerInterface rmiServer = new RmiServer();

        ServerInterface stub = (ServerInterface)
                UnicastRemoteObject.exportObject(rmiServer, 0);

        Naming.rebind("toUpperCase", stub);
        System.out.println("Starting server...");
    }
    public RmiServer()
    {
    }
```

Upload the stub to registry and bind it to a name/string

Publish the object (start listening for clients)

Create an object of the class implementing the remote interface

RmiServer (in parts) – constructor

```
public class RmiServer implements ServerInterface
{
   public static void main(String[] args) throws RemoteException
   {
      Registry reg = LocateRegistry.createRegistry(1099);
      RmiServer server = new RmiServer();
   }
   public RmiServer() throws RemoteException
   {
      ServerInterface stub = (ServerInterface)UnicastRemoteObject
            .exportObject(this, 0);

      Naming.rebind("toUpperCase", this);
      System.out.println("Starting server...");   }
   }
```

Upload the stub to registry and bind it to a name/string

Publish the object (start listening for clients)

An object of the class implementing the remote interface

RmiServer (in parts) – constructor

```
public class RmiServer implements ServerInterface
{
   public static void main(String[] args) throws RemoteException
   {
      Registry reg = LocateRegistry.createRegistry(1099);
      RmiServer server = new RmiServer();
   }
   public RmiServer() throws RemoteException
   {
      ServerInterface stub = (ServerInterface)UnicastRemoteObject
            .exportObject(this, 0);

      Naming.rebind("toUpperCase", stub);
      System.out.println("Starting server...");   }
   }
```

Upload the stub to registry and bind it to a name/string

Publish the object (start listening for clients)

A stub to the object of the class implementing the remote interface

Create and publish the remote object

- Method 1
  - The "server" `extends UnicastRemoteObject`
- Method 2
  - The "server" calls method

```
UnicastRemoteObject.exportObject(server, 0);
```

Publish remote object – extending

```
public class RmiTaskServer extends UnicastRemoteObject
                           implements RemoteTaskList
{
   // ...

   public RmiTaskServer() throws RemoteException
   {
      // ...
      super();
      Naming.rebind("tasks", this);
   }

   // ...
```

Publish remote object – without extending

```
public class RmiTaskServer implements RemoteTaskList
{
   // ...

   public RmiTaskServer() throws RemoteException, ...
   {
      // ...
      RemoteTaskList stub = (RemoteTaskList)
            UnicastRemoteObject.exportObject(this, 0);
      Naming.rebind("tasks", stub);
   }

   // ...
```

Publish remote object – without extending

```
public class RmiTaskServer implements RemoteTaskList
{
   // ...

   public RmiTaskServer() throws RemoteException, ...
   {
      // ...
      UnicastRemoteObject.exportObject(this, 0);
      Naming.rebind("tasks", this);
   }

   // ...
```

Start the Registry (in its own try-catch block)

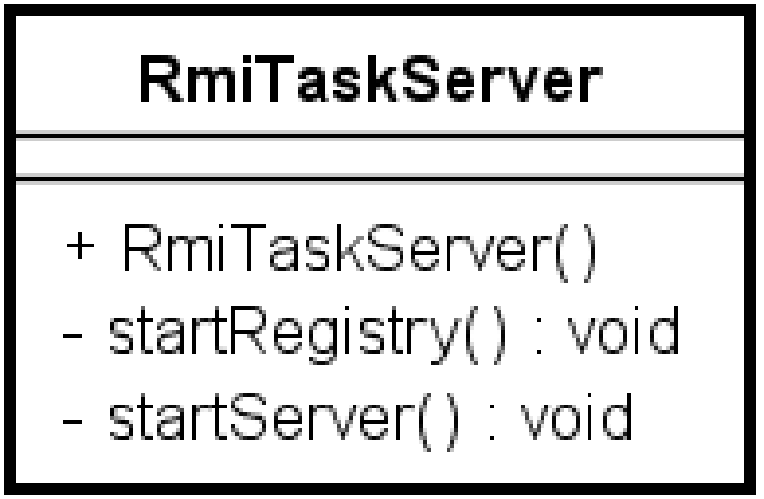
- If registry is already started:

```
java.rmi.server.ExportException: Port already in use: 1099;
nested exception is:
    java.net.BindException: Address already in use: JVM_Bind
    ...
```

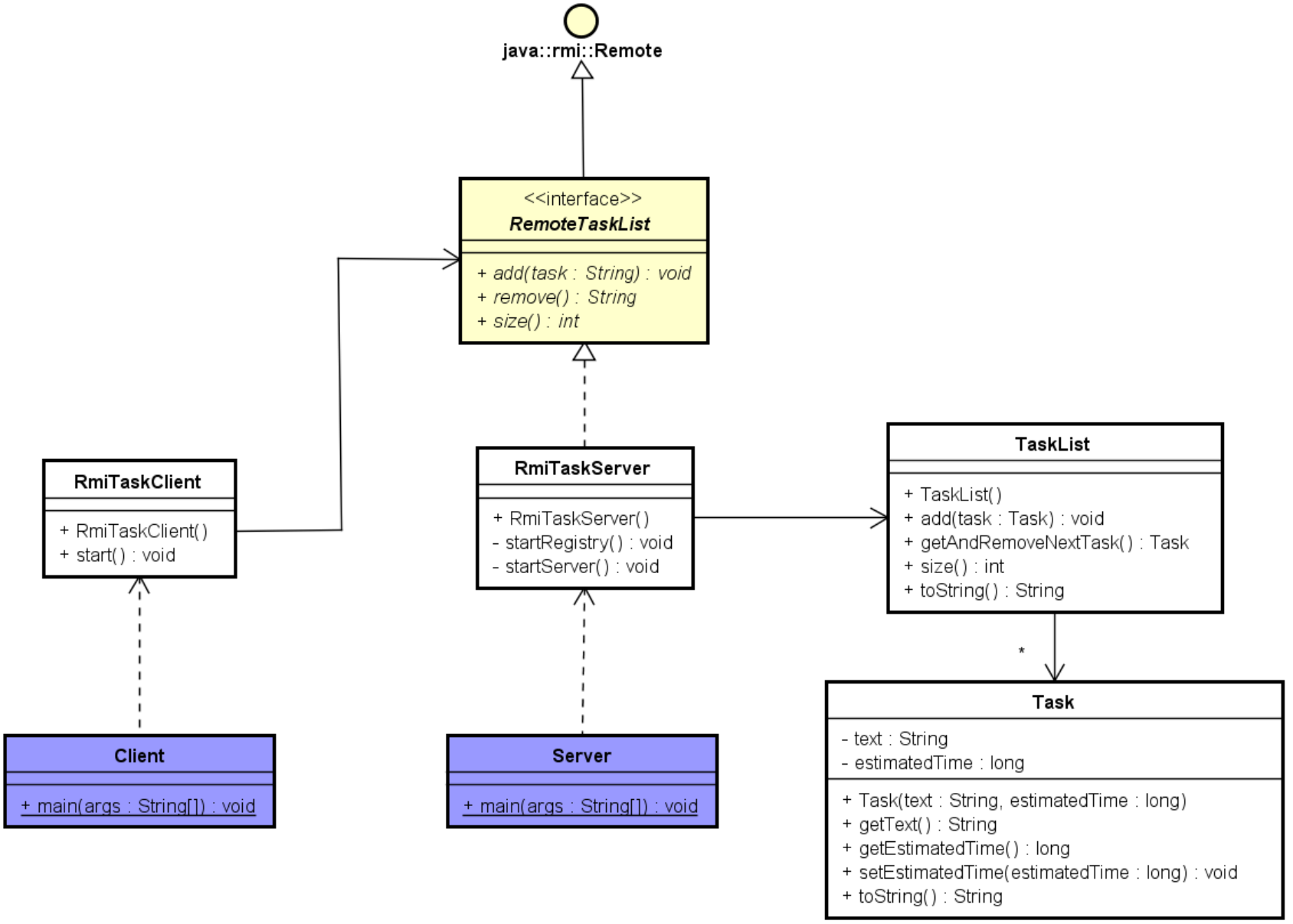
```
try
{
   Registry reg = LocateRegistry.createRegistry(1099);
   System.out.println("Registry started...");
}
catch (java.rmi.server.ExportException ex)
{
  // already started
  System.out.println("Registry already started?"
                     + " Error: " + ex.getMessage());
}
```

RmiTaskServer with private methods

```
public RmiTaskServer()
{
   //...
   startRegistry();
   startServer();
}
```

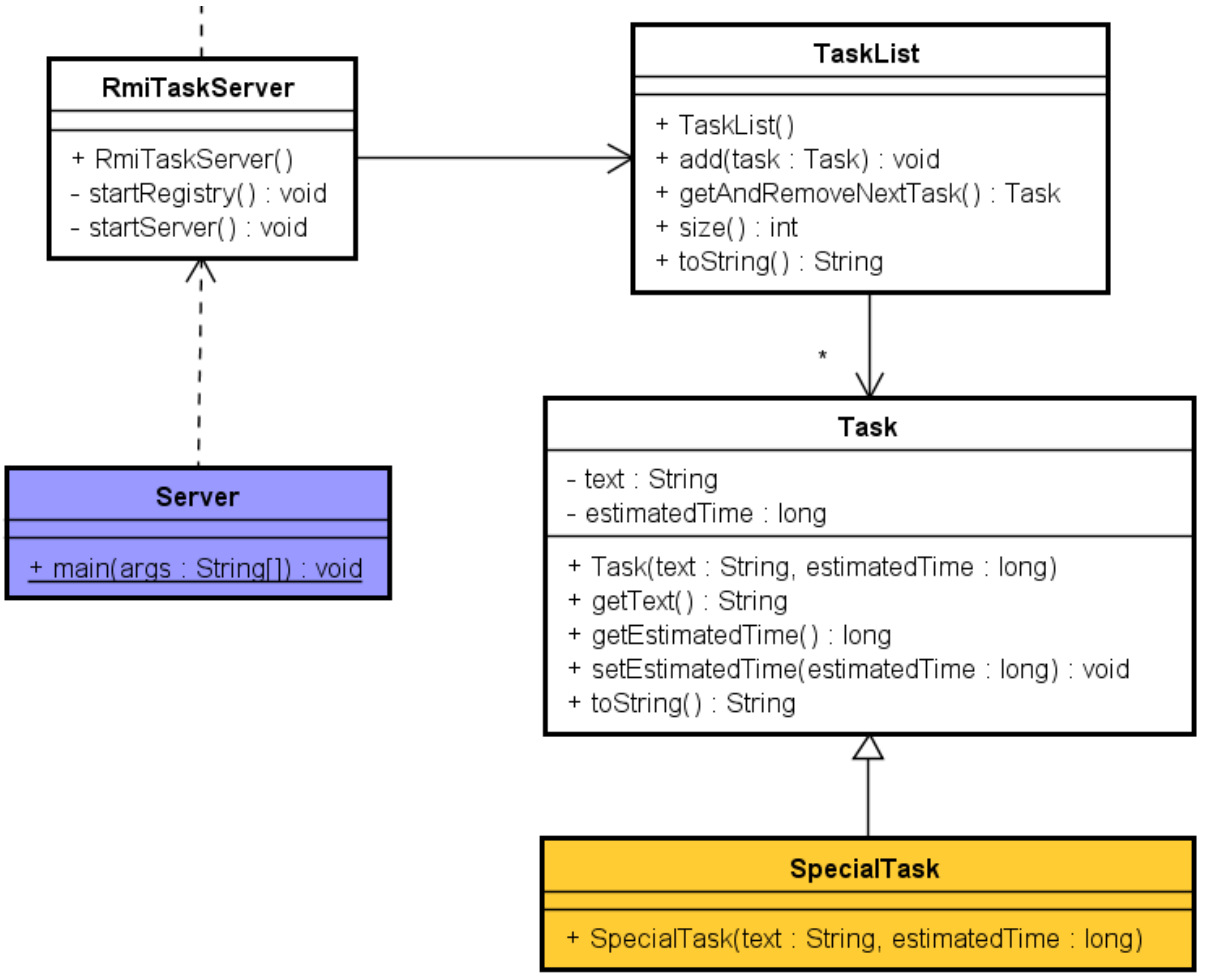


RMI version of a remote Task list



Version 2 (Server may create SpecialTask's)

- Client don't have the class file for SpecialTask



Security – main method

```
public class Client
{
  public static void main(String[] args) throws RemoteException
  {
     if (System.getSecurityManager() == null)
     {
        System.setSecurityManager(new SecurityManager());
     }
     RmiTaskClient client = new RmiTaskClient();
     client.start();
  }
}
```

```
public class Server
{
  public static void main(String[] args) throws Exception
  {
      if (System.getSecurityManager() == null)
      {
         System.setSecurityManager(new SecurityManager());
      }
      RemoteTaskList server = new RmiTaskServer();
   }
}
```

Run the Server (outside Eclipse)

server.bat

```
java -Djava.security.policy=all.policy Server
pause
```

all.policy

```
grant {
   permission java.security.AllPermission;
};
```

Run the Client (outside Eclipse)

client.bat

```
java -Djava.rmi.server.codebase=http://ict-engineering.dk/class/
     -Djava.security.policy=all.policy Client
pause
```

all.policy

```
grant {
  permission java.security.AllPermission;
};
```