Aufgabe 5

RPC lokal (Teilaufgabe a)

Der lokale RPC-Provider ist eine simple Anwendung von Reflection: Er sucht sich die Klasse, deren Methode und ruft diese (statisch = mit Objektreferenz null) auf. Wurde das entsprechende Flag gesetzt, werden dabei die Parameter-Klassen mit wrapToPrimitiveClass verarbeitet.

```
paramClasses[i] = RPCSecrets.warpToPrimitiveClass(params[i].getClass());
```

Alle dabei auftretenden Exceptions werden als RPCException gekapselt.

```
catch(ClassNotFoundException e)
{
	throw new RPCException(e);
}
// ...
```

RPC über das Netzwerk (Teilaufgaben b und c)

Der RPCRemoteServiceProvider baut für die Anfrage ein Protobuf-Paket und serialisiert dafür die Parameter:

```
for(Serializable p: params)
{
	call.addParameters(RPCSecrets.serialize(p));
}
```

Anschließend sendet er das Paket per UDP an den vorher festgelegten Server. Dieser entpackt das Paket, deserialisiert die Parameter und führt die Methode über den lokalen RPC-Provider aus.

```
Serializable[] params = (Serializable[])
RPCSecrets.deserialize(call.getParametersList());

result = RPCSecrets.serialize(serviceProvider.callexplicit(call.getClassname(),
call.getMethodname(), params));
```

In der Variable result wird im Erfolgsfall der Rückgabewert gespeichert, oder, wenn eine Exception auftritt, die Exception selbst. Anschließend wird das Antwortpaket gepackt, entweder mit einem Rückgabewert oder einer Exception, und per UDP an den gleichen Port zurückgesendet.

```
if(isException) {
	response.setException(result);
} else {
	response.setResult(result);
}
```

Der Client, der auf die Antwort gewartet hat, entpackt das Paket und reicht entweder die Exception oder den Rückgabewert an den Aufrufer weiter.

```
if(result.hasException())
{
	throw new RPCException("Server exception",
RPCSecrets.deserialize(result.getException()));
}
else if(result.hasResult())
{
	return RPCSecrets.deserialize(result.getResult());
}
```