Systemisches Software-Engineering

Christian Silberbauer

Übungsblatt CounterService

Folgende Aufgaben sind aufeinander aufbauend bis Aufgabe 6. Sehen Sie bitte für jede Aufgabe ein eigenes Projekt vor. Beurteilen Sie umfassend bei jeder Aufgabe schriftlich die Gesamtarchitektur.

Aufgabe 1

Reproduzieren Sie das Counter-Beispiel (ursprünglich; ohne RMI) aus den Folien. Gruppieren Sie die Klassen je nach Zugehörigkeit in die Packages „client“, „server“ und „common“.

Aufgabe 2

Integrieren Sie einen `AddService`, der es erlaubt, zwei Zahlen zu addieren und das Ergebnis zurückzugeben. Dieser soll folgendes Interface implementieren:

```
public interface Add {
   double add(double summand1, double summand2);
}
```

Verwenden Sie zum Testen folgende Client-Klasse:

```
public class Client {
   public static void main(String[] args) {
      Counter counter =
         ServiceProvider.instance().createService(Counter.class);
      counter.setValue(1000);
      for (int i = 0; i < 3; i++) {
         counter.increment();
      }

      int r = counter.getValue();
      System.out.println(r);

      Add adder = ServiceProvider.instance().createService(Add.class);
      System.out.println(adder.add(3, 7));
   }
}
```

Aufgabe 3

Schaffen Sie eine generische Lösung mit Hilfe der Java-Reflection.
Service-spezifische Stubs und Skeletons sowie Service- und Method-IDs werden dadurch obsolet.

Werfen Sie eine `UnknownServiceException` bei unbekanntem Service und eine `UnknownMethodException` bei unbekannter Methode. Definieren Sie beide Exceptions als `RuntimeException`s.

Entfernen Sie zudem die `throws`-Klauseln der Interface-Methoden und des `ServiceProvider`s des `Client`s.

Umwickeln Sie etwaige `IOException`s mit `RuntimeException`.

Aufgabe 4

Stellen Sie sicher, dass ein `Client` nur eine `Socket`-Verbindung zum `Server` hat, statt eine Verbindung pro Service.

Aufgabe 5

Erlauben Sie, dass der Client zunächst seinen Server auf Port `8011` sucht, falls dieser nicht vorhanden ist, soll ein Server auf Port `8012` verwendet werden. Falls dieser nicht vorhanden ist, soll der `Counter`-Service lokal angeboten werden.

Stellen Sie sicher, dass der Code redundanzfrei bleibt.

Aufgabe 6

Schaffen Sie eine generische (generierte) Lösung mit Hilfe von Java-RMI auf Basis von Aufgabe 2.