



TECHNOLOGISCHES GEWERBE MUSEUM

DezSys

PI Calculator

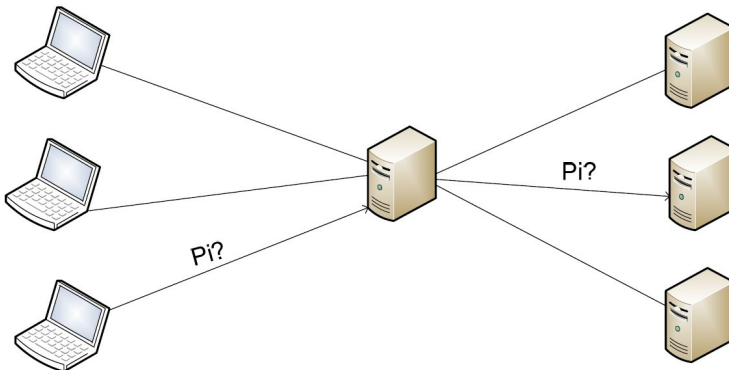
Author: Pöcher RENÉ & Steinkellner SEBASTIAN

January 8, 2015

Contents

1	Aufgabe	2
2	Working time	4
2.1	Estimated Working time	4
3	Easy Bibliography	5

1 Aufgabe



Als Dienst soll hier die beliebig genaue Bestimmung von π betrachtet werden. Der Dienst stellt folgendes Interface bereit:

```
// Calculator.java
public interface Calculator {
    public BigDecimal pi (int anzahl_nachkommastellen);
}
```

Ihre Aufgabe ist es nun, zunächst mittels Java-RMI die direkte Kommunikation zwischen Klient und Dienst zu ermöglichen und in einem zweiten Schritt den Balancier zu implementieren und zwischen Klient(en) und Dienst(e) zu schalten. Gehen Sie dazu folgendermassen vor:

Entwickeln Sie ein Serverprogramm, das eine `CalculatorImpl`-Instanz erzeugt und beim RMI-Namensdienst registriert. Entwickeln Sie ein Klientenprogramm, das eine Referenz auf das `Calculator`-Objekt beim Namensdienst erfragt und damit π bestimmt. Testen Sie die neu entwickelten Komponenten.

Implementieren Sie nun den Balancier, indem Sie eine Klasse `CalculatorBalancer` von `Calculator` ableiten und die Methode `pi()` entsprechend implementieren. Dadurch verhält sich der Balancier aus Sicht der Klienten genauso wie der Server, d.h. das Klientenprogramm muss nicht verändert werden. Entwickeln Sie ein Balancierprogramm, das eine `CalculatorBalancer`-Instanz erzeugt und unter dem vom Klienten erwarteten Namen beim Namensdienst registriert. Hier ein paar Details und Hinweise:

Da mehrere Serverprogramme gleichzeitig gestartet werden, sollten Sie das Serverprogramm so erweitern, dass man beim Start auf der Kommandozeile den Namen angeben kann, unter dem das `CalculatorImpl`-Objekt beim Load-Balancer gespeichert wird. Damit soll der Server nun seine exportierte Instanz an den Balancer übergeben, ohne es in die Registry zu schreiben. Verwenden Sie dabei ein eigenes Interface des Balancers, welches in die Registry gebündelt wird,

um den Servern das Anmelden zu ermöglichen.

Das Balancierer-Programm sollte nun den Namensdienst in festgelegten Abständen abfragen um herauszufinden, ob neue Server Implementierungen zur Verfügung stehen. Java-RMI verwendet intern mehrere Threads, um gleichzeitig eintreffende Methodenaufrufe parallel abarbeiten zu können. Das ist einerseits von Vorteil, da der Balancierer dadurch mehrere eintreffende Aufrufe parallel bearbeiten kann, andererseits müssen dadurch im Balancierer änderbare Objekte durch Verwendung von `synchronized` vor dem gleichzeitigen Zugriff in mehreren Threads geschützt werden.

Beachten Sie, dass nach dem Starten eines Servers eine gewisse Zeit vergeht, bis der Server das `CalculatorImpl`-Objekt erzeugt und beim Namensdienst registriert hat sich beim Balancer meldet. D.h. Sie müssen im Balancierer zwischen Start eines Servers und Abfragen des Namensdienstes einige Sekunden warten.

2 Working time

2.1 Estimated Working time

Estimated working time

Task	Person	Time in hours
Planung	Pöcher Steinkellner	1
UML Design	Pöcher	1/2
PI Methode desig. & Implementieren	Pöcher	2
Total	Pöcher Steinkellner	2 1/2 1
Total Team		4 hours

Table 1: Estimated working time

3 Easy Bibliography

List of Tables

1	Estimated working time	4
---	----------------------------------	---

List of Figures

References

- [1] **Who**, When
url
last used: dd.mm.yyyy, hh:mm
- [2] **Mister Super-genious**, Answer from 20.01.2015
<http://www.stackoverflow.com/question>
last used: 22.10.2014, 21:00