RMI – PI CALCULATOR

SYT / DEZSYS07

14. JANUAR 2015

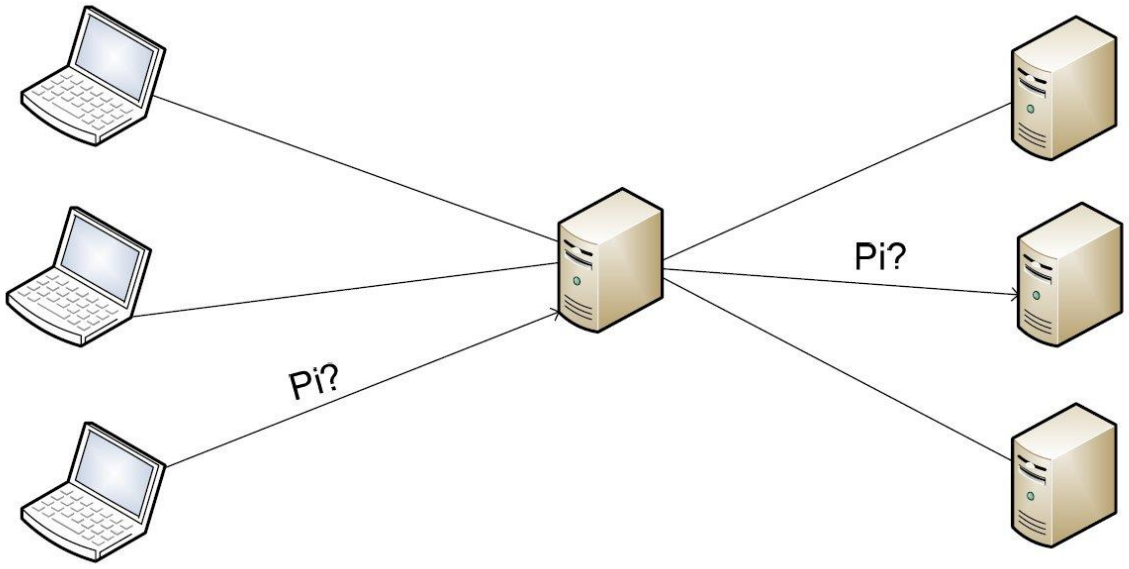
ERCEG, KRITZL
4AHITT

Inhalt

Github-Link: https://github.com/mkritzl-tgm/DezSys07-PI_Calculator

Github-Tag: erceg_kritzl_dezsys07_v1

1. Aufgabenstellung



Als Dienst soll hier die beliebig genaue Bestimmung von pi 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 Balancierer zu implementieren und zwischen Klient(en) und Dienst(e) zu schalten. Gehen Sie dazu folgendermassen vor:

1. Entwicklen Sie ein Serverprogramm, das eine CalculatorImpl-Instanz erzeugt und beim RMI-Namensdienst registriert. Entwicklen Sie ein Klientenprogramm, das eine Referenz auf das Calculator-Objekt beim Namensdienst erfragt und damit pi bestimmt. Testen Sie die neu entwickelten Komponenten.
2. Implementieren Sie nun den Balancierer, indem Sie eine Klasse CalculatorBalancer von Calculator ableiten und die Methode pi() entsprechend implementieren. Dadurch verhält sich der Balancierer aus Sicht der Klienten genauso wie der Server, d.h. das Klientenprogramm muss nicht verändert werden. Entwickeln Sie ein Balanciererprogramm, 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 ksnn, unter dem das CalculatorImpl-Objekt beim Namensdienst registriert wird. dieses nun seine exportierte Instanz an den Balancierer übergibt, ohne es in die Registry zu schreiben. Verwenden Sie

dabei ein eigenes Interface des Balancers, welches in die Registry gebinded 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.

Testen Sie das entwickelte System, indem Sie den Balancierer mit verschiedenen Serverpoolgrössen starten und mehrere Klienten gleichzeitig Anfragen stellen lassen. Wählen Sie die Anzahl der Iterationen bei der Berechnung von pi entsprechend groß, sodass eine Anfrage lang genug dauert um feststellen zu können, dass der Balancierer tatsächlich mehrere Anfragen parallel bearbeitet.

2. Requirementanalyse

Arbeitspaket	Zuständige Person	Geschätzte Zeit	Erledigt
Client	Erceg	60 min	x
Server	Erceg	60 min	x
Balancer		450 min	x
Balancier-Algorithmus	Erceg	210 min	x
Service (Registry)	Kritzl	240 min	x
CLI	Kritzl	120 min	x
Calculator-Algorithmus	Erceg	30 min	x
Gesamt		720 min	x

3. detaillierte Arbeitsaufteilung mit Aufwandsabschätzung

3.1 Aufwandabschätzung

Teilaufgabe	benötigte Gesamtzeit
UML-Diagramm erstellen	240 Minuten (4 Stunden)
Implementierung des Programms inkl. JavaDoc	720 Minuten (12 Stunden)
Testen des Programms	300 Minuten (5 Stunden)
Protokoll schreiben	180 Minuten (3 Stunden)
Gesamt	**1440 Minuten (24 Stunden)**

2.2 Arbeitsaufteilung für die Implementierung des Programms

2.2.1 Package „control“

Klassen/Interfaces	Erceg	Kritzl
CLI		x
Input		x
Main		x

2.2.2 Package „algorithm“

Klassen/Interfaces	Erceg	Kritzl
BalancerAlgorithm	x	
CalculatorAlgorithm	x	
SequenceAlgorithm	x	

2.2.3 Package „components“

Klassen/Interfaces	Erceg	Kritzl
Balancer		x
Calculator	x	
Client	x	
Server	x	

2.2.4 Package „service“

Klassen/Interfaces	Erceg	Kritzl
CalcService		x
Service		x

4. anschließende Endzeitaufteilung

4.1 Erceg

Arbeit	Datum	Zeit in Minuten
UML	12.12.2014	180 Minuten
UML fertiggestellt	06.01.2015	30 Minuten
Protokoll	07.01.2015	70 Minuten
Implementierung	07.01.2015	120 Minuten
Implementierung	08.01.2015	420 Minuten
Implementierung	11.01.2015	180 Minuten
Programm testen	14.01.2015	240 Minuten
Protokoll	14.01.2015	100 Minuten
Gesamt	*14.01.2015*	**1340 Minuten (22 h 20 min)**

4.2 Kritzl

Arbeit	Datum	Zeit in Minuten
UML	12.12.2014	180 Minuten
UML	04.01.2015	30 Minuten
UML fertiggestellt	06.01.2015	30 Minuten
Protokoll	07.01.2015	70 Minuten
Implementierung	07.01.2015	120 Minuten
Implementierung	08.01.2015	420 Minuten
Implementierung	09.01.2015	60 Minuten
Implementierung	10.01.2015	120 Minuten
Implementierung	11.01.2015	240 Minuten
Implementierung	12.01.2015	120 Minuten
Implementierung	13.01.2015	120 Minuten
Programm testen	14.01.2015	240 Minuten
Gesamt	*14.01.2015*	**1750 Minuten (29 h 10 min)**

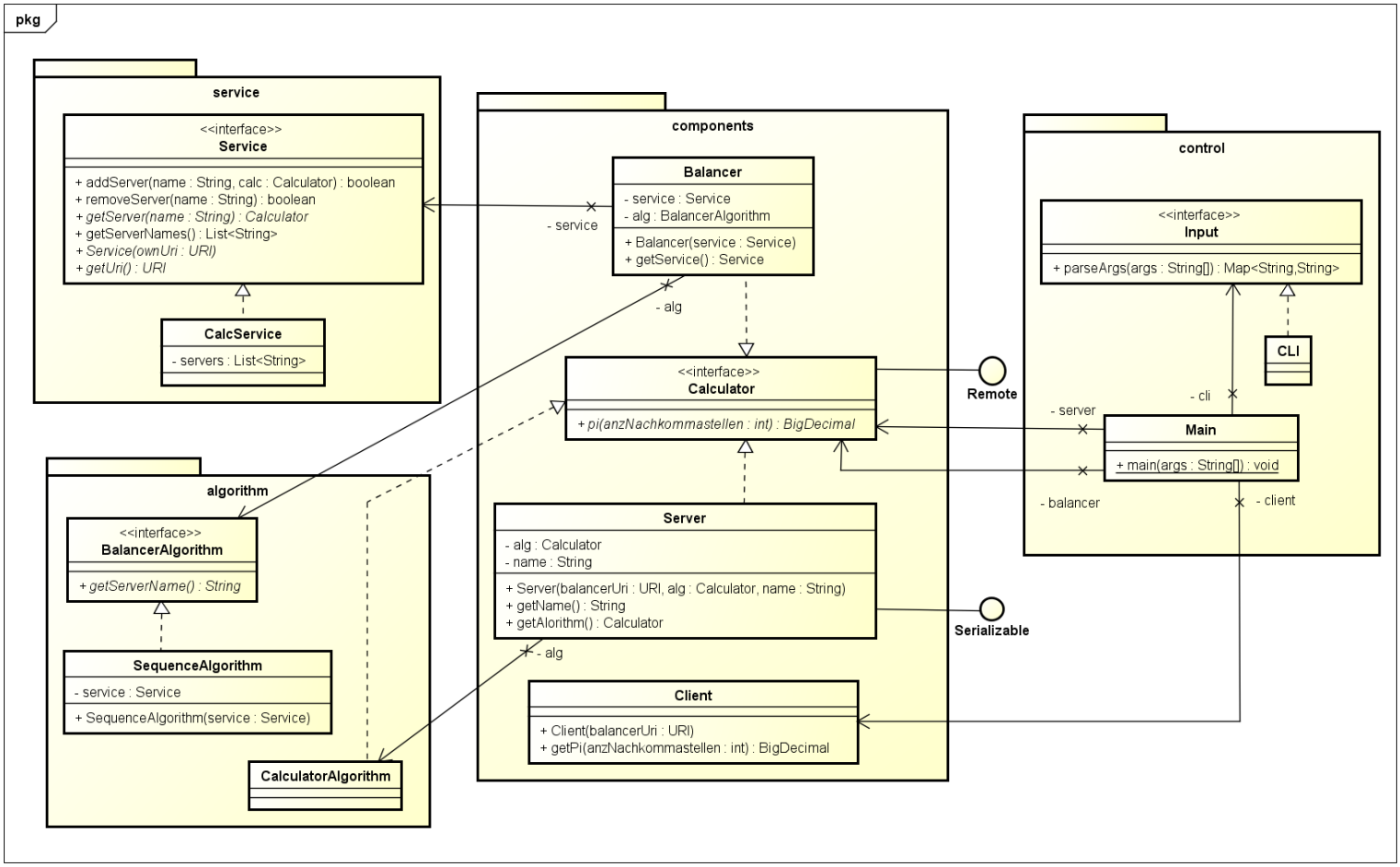
4.3 Gesamtsumme

Insgesamt haben wir für diese Übung **51 Stunden und 30 Minuten** benötigt. Geschätzt wurden 24 Stunden, daher lag unsere Einschätzung ziemlich daneben.

5. Designüberlegung

5.1 Abbildung

Das UML-Diagramm wurde mit dem Programm „Astah“ erstellt.



5.2 Überlegungen zur Struktur

Wir haben uns überlegt, unser Programm in 4 Packages unterzuordnen:

1.) control

In diesem Package befinden sich die Klassen, die mit der Eingabe und der Verwaltung dieser Eingaben beschäftigt sind.

2.) algorithm

Dazu zählen die zwei Algorithmen zur Bestimmung des verarbeitenden Servers und der Berechnung von PI.

3.) components

Hier wird mit den 3 Klassen „Balancer“, „Client“ und „Server“ die Verbindung zwischen den jeweiligen Komponenten definiert.

4.) service

Die Klasse, die sich in diesem Package befindet, kümmert sich um die Verwaltung der verfügbaren Server. Die Server können dabei zu einer Registry hinzugefügt oder entfernt werden.

5.2.1 Package „control“

Klassen, die im Package enthalten sind:

1.) Input (Interface)

- beinhaltet die Methode „parseArgs“, mit der die jeweiligen vom Benutzer eingegebenen Argumente getrennt werden

2.) CLI

- implementiert das Interface Input

3.) Main

- initialisiert die benötigten Objekte abhängig durch die Eingabe des Benutzers
- betreibt das Exception-Handling

5.2.2 Package „algorithm“

Klassen, die im Package enthalten sind:

1.) BalancerAlgorithm (Interface)

- schreibt die Methode „getServerName“ vor, die den nächsten zur Verfügung stehenden Server ermittelt

2.) SequenceAlgorithm

- implementiert das Interface BalancerAlgorithm

3.) CalculatorAlgorithm

- implementiert das Interface Calculator (siehe 5.2.3 – Punkt 4)
- dient zur Berechnung von PI

5.2.3 Package „components“

Klassen, die im Package enthalten sind:

1.) Client

- bittet den Balancer für die Berechnung von PI mit den angegebenen Nachkommastellen

2.) Server

- berechnet für den Client PI und gibt das Ergebnis dem Balancer zurück

3.) Balancer

- stellt die Verbindung zwischen Client und Server dar
- verwendet den Balancer-Algorithmus und den Service

4.) Calculator (Interface)

- schreibt die Methode „pi“ vor, welcher als Parameter die Anzahl der Nachkommastellen übergeben wird und als Ergebnis ein BigDecimal zurückgibt

5.2.4 Package „service“

Klassen, die im Package enthalten sind:

1.) Service (Interface)

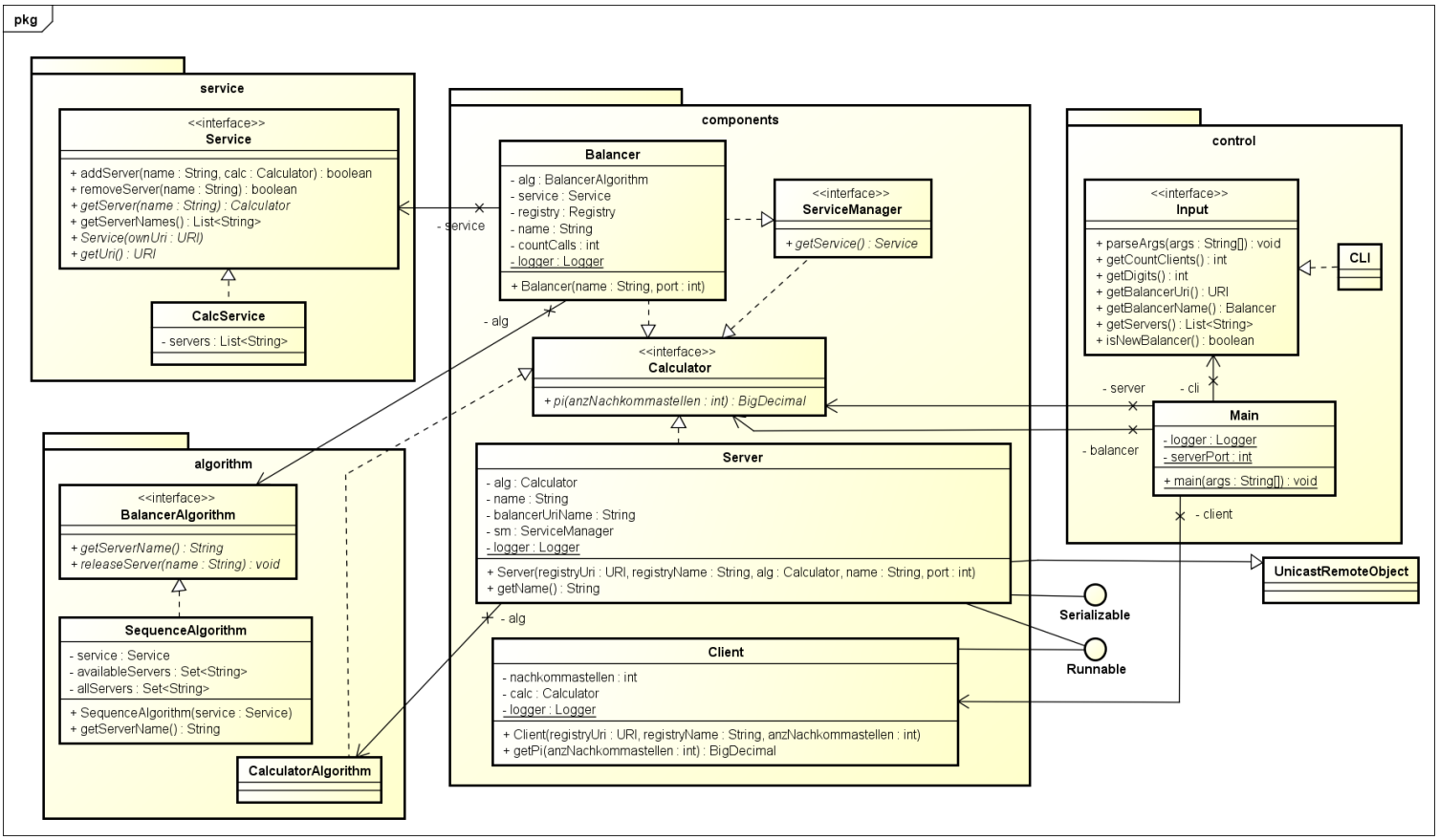
- bietet Methoden zum Hinzufügen und Entfernen von bestimmten Servern an
- das Erhalten aller Servernamen in einer Liste geschieht durch die Methode „getServerNames“
- mittels der Methode „getServer“ kann ein bestimmter Server, dessen Name im Parameter angegeben wird, zurückgegeben werden
- mittels der Methode „getURI“ kann die URI des Balancers ausgelesen werden

2.) CalcService

- implementiert das Interface Service

6. Arbeitsdurchführung

Da wir während der Implementierung auf einige Verbesserungen bezüglich der Struktur gekommen sind, sieht unser finales UML-Diagramm folgendermaßen aus:



- Erstellen einer Requirement-Analyse(Aufwandabschätung, Arbeitsaufteilung)
- Überlegen eines Designs in Form eines UML Diagramms
- Generieren dieses Quellcodes über die Exportfunktion von Astah
- Erstellen eines Build-Files für ein einfacheres Testen des Programms
- Algorithmus für das Berechnen von Pi von einer Oracle Tutorial Seite heruntergeladen [1]
- Implementierung der Klassen in ca. folgender Reihenfolge:
  - Main
  - CLI
  - CalcService
  - CalculatorAlgorithm
  - Balancer
  - Server
  - Client
  - SequenceAlgorithm
- Testen der einzelnen Funktionen
- Testen des Gesamtsystems

7. Testbericht

Für die einzelnen funktionalen Anforderungen, wie z.B. die CLI-Eingaben oder der Service für das Hinzufügen bzw. Entfernen von Server, wurden Unit Tests erstellt. Diese können sie im Source-Ordner im Package „at.erceg_kritzl.pi_calculator.tests“ einsehen.

Für die Kommunikation zwischen Client, Balancer und Server haben wir folgenden Testbericht erstellt:

Durchführung	Anzahl d. Clients	Anzahl d. Server	Erwartetes Ergebnis	Tatsächliches Ergebnis
In der CMD notwendige Befehle eingeben	100	2	PI mit 50 Nachkommastellen; Balancer-Verarbeitung	Wie erwartetes Ergebnis (siehe Abb 1 unten)
In der CMD notwendige Befehle eingeben	6	3	PI mit 10 000 Nachkommastellen; Balancer-Verarbeitung	Wie erwartetes Ergebnis (siehe Abb 2 unten)
In der CMD notwendige Befehle eingeben	50	2	PI mit 1000 Nachkommastellen; Balancer-Verarbeitung	Wie erwartetes Ergebnis (siehe Abb 3 unten)

Abb 1:

Balancer:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-n true -b rmi://127.0.0.1:60000/MeinBalancer
2015-01-14 20:30:42,221 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- MeinBalancer hat sich unter 10.0.0.1 angemeldet.
```

```
2015-01-14 20:37:44,083 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 50 Stellen berechnet.(Aufruf nr. 98)
2015-01-14 20:37:44,085 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 50 Stellen berechnet.(Aufruf nr. 99)
2015-01-14 20:37:46,622 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 50 Stellen berechnet.(Aufruf nr. 100)
```

Server:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-s server1,server2 -b rmi://127.0.0.1:60000/MeinBalancer
2015-01-14 20:32:18,133 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server1 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
2015-01-14 20:32:18,157 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server2 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
```

Client:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-c 100 -d 50 -b rmi://127.0.0.1:60000/MeinBalancer
```

```
3.14159265358979323846264338327950288419716939937511
3.14159265358979323846264338327950288419716939937511
3.14159265358979323846264338327950288419716939937511
3.14159265358979323846264338327950288419716939937511
```

Abb 2:

Balancer:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-n true -b rmi://127.0.0.1:60000/MeinBalancer
2015-01-14 20:44:19,955 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- MeinBalancer hat sich unter 10.0.0.1 angemeldet.
2015-01-14 20:44:23,866 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server3 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 1)
2015-01-14 20:44:23,879 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 2)
2015-01-14 20:44:23,881 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server1 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 3)
2015-01-14 20:44:26,793 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server1 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 4)
2015-01-14 20:44:26,793 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 5)
2015-01-14 20:44:26,815 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server3 hat pi fuer 10000 Stellen berechnet.(Aufruf nr. 6)
```

Server:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-s server1,server2,server3 -b rmi://127.0.0.1:60000/MeinBalancer
2015-01-14 20:43:31,290 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server1 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
2015-01-14 20:43:31,316 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server2 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
2015-01-14 20:43:31,323 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server3 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
```

Client:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-c 6 -d 10000 -b rmi://127.0.0.1:60000/MeinBalancer
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778057321712268066130019278766111959092164201989380952572010654858632788659361533818279682303019520353018529689957736225994138912497217752834791315155748572424541506959508295331168617278558890750983817546374649393192550604009277016711390098488240128583616035637076601047101819429555961989467678374494482553797747268471040475346462080466842590694912933136770289891521047521620569660240580381501935112533824300355876402474964732639141992726042699227967823547816360093417216412199245863150302861829745557067498385054945885869269956909272107975093029553211653449872027559602364806654991198818347977535663698074265425278625518184175746728909777727938000816470600161452491921732172147723501414419735685481613611573525521334757418494684385233239073941433345477624168625189835694855620992192221842725502542568876717904946016534668049886272327917860857843838279679766814541009538837863609506800642251252051173929848960841284886269456042419652850222106611863067442786220391949450471237137869609563643719172874677646575739624138908658326459958133904780275909099465764078951269468398352595709825822620522489407726719478268482601476990902640136394437455305068203496252451749399651431429809190659250937221696461515709858 3
```

Abb 3:

Balancer:

```
2015-01-14 20:47:55,754 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server1 hat pi fuer 1000 Stellen berechnet.(Aufruf nr. 48)
2015-01-14 20:47:58,548 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 1000 Stellen berechnet.(Aufruf nr. 49)
2015-01-14 20:47:58,738 INFO  at.erceg_kritzl.pi_calculator.components.Balancer
- server2 hat pi fuer 1000 Stellen berechnet.(Aufruf nr. 50)
```

Server:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-s server1,server2 -b rmi://127.0.0.1:60000/MeinBalancer
2015-01-14 20:32:18,133 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server1 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
2015-01-14 20:32:18,157 INFO  at.erceg_kritzl.pi_calculator.components.Server -
server2 hat sich bei MeinBalancer unter rmi://127.0.0.1:60000/MeinBalancer angem
eldet.
```

Client:

```
C:\Users\Martin Kritzl>java -jar "C:\Users\Martin Kritzl\Schule\4AHITT\SYT\Dezen
trale Systeme\Progamme\DezSys07-PI_Calculator\build3\Erceg_Kritzl_DezSys07.jar"
-c 50 -d 1000 -b rmi://127.0.0.1:60000/MeinBalancer
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019 89
```

8. Lessons learned

- Die Verwendung eines Policy-Files ist unbedingt notwendig
- Jedes Objekt, das sich zur Laufzeit nicht auf derselben Maschine befindet muss von UnicastRemoteObjekt erben oder durch die statische Methode exportObject exportiert werden
- Gleichzeitiger Zugriff auf Attribute muss verhindert werden
- Die Verwendung von Build-Files erweist sich bei der mehrfacher gleichzeitigen Verwendung des Programms als sehr zeitsparend
- `new Thread(Runnable).setDaemon(true)` fährt die Threads beim Schließen des Programms herunter
- Die Interfaces mit Methoden, die von einem anderen Rechner aus aufgerufen werden sollen, müssen Remote extenden
- Mit Naming.lookup(URI/NameObjekt) wird das gewünschte Objekt aus der Registry geholt
- Mit Registry.bind(NameObjekt, Objekt) wird das gewünschte Objekt in der Registry eingetragen

9. Quellenangaben

[1] Oracle (1995, 2008). Java Tutorials Code Sample – Pi.java [Online]. Available at: http://docs.oracle.com/javase/tutorial/displayCode.html?code=http://docs.oracle.com/javase/tutorial/rmi/examples/client/Pi.java [zuletzt abgerufen am 14.01.2015]