1 Übungsaufgabe #6: Asynchrone Fernaufrufe

1.1 Allgemeines

In dieser Aufgabe soll das Fernaufruf-System um die Unterstützung von asynchronen Fernaufrufen (mit und ohne Rückgabewert) erweitert werden. Die Grundidee dieses Konzepts besteht darin, einem Aufrufer die Möglichkeit einzuräumen, selbst zu entscheiden, wie er die Zeit zwischen dem Absetzen eines Fernaufrufs und dem Eintreffen der korrespondierenden Antwort nutzt. Sollte ein Resultat nicht unmittelbar im nächsten Programmschritt benötigt werden, lässt sich somit im Vergleich zum synchronen Fall die mit einem Fernaufruf verbundene erhöhte Antwortzeit eines Methodenaufrufs teilweise maskieren.

1.2 Futures

1.2.1 Schnittstelle (für alle)

Wie in der Tafelübung erläutert, zeichnen sich asynchrone (Fern-)Aufrufe üblicherweise dadurch aus, dass sie anstatt des eigentlichen Ergebnisses ein *Future*-Objekt zurückliefern. Dieses stellt im Allgemeinen folgende Schnittstelle zur Verfügung (vgl. *java.util.concurrent.Future*):

```
public interface VSFuture {
    public Object get() throws InterruptedException, ExecutionException;
    public boolean isDone();
}
```

Ein Aufruf von *get()* am Future-Objekt blockiert solange, bis der Rückgabewert (bzw. die Bestätigung, dass der Methodenaufruf beendet ist) lokal vorliegt. Anschließend wird dieser zurückgegeben. Sollte bei der Ausführung der Methode ein Fehler auftreten, wird dies über eine *ExecutionException* signalisiert. Um nichtblockierend festzustellen, ob ein Methodenaufruf beendet ist, kann *isDone()* verwendet werden.

Aufgaben:

→ Realisierung einer Klasse *VSFutureImpl*, die das Interface *VSFuture* implementiert.

Hinweise:

- *isDone()* liefert auch dann *true*, wenn der Methodenaufruf ausnahmebedingt abgebrochen wurde.

1.2.2 Methodenaufruf (für alle)

Die Frage, ob es sich bei einem Fernaufruf um einen synchronen oder asynchronen Aufruf handelt, spielt für die Server-Seite keine Rolle: In beiden Fällen ist die identische Operation auszuführen. Auf Client-Seite dagegen muss entschieden werden, ob ein Aufruf entweder blockiert oder stattdessen sofort zurückkehrt. Zu diesem Zweck soll jeder erzeugte Stub (→ Proxy) pro Remote-Interface eine erweiterte Schnittstelle zur Durchführung asynchroner Fernaufrufe anbieten. Diese umfasst zu jeder verfügbaren Methode eine asynchrone Variante, die durch das Postfix „Async" gekennzeichnet ist, z.B.:

```
public interface RemoteInterface extends VSRemote {
    public void sayHello();
    public int multiply(int a, int b);
}

public interface RemoteInterfaceAsync extends RemoteInterface {
    public VSFuture sayHelloAsync();
    public VSFuture multiplyAsync(int a, int b);
}
```

Während das eigentliche Objekt (auf Server-Seite) nur die Schnittstelle *RemoteInterface* implementiert, kann auf Client-Seite zusätzlich über *RemoteInterfaceAsync* darauf zugegriffen werden. Bei asynchronen Aufrufen ist es also Aufgabe der Client-Seite des Fernaufrufsystems diese intern in synchrone Fernaufrufe umzuwandeln.

Aufgaben:

→ Integration asynchroner Fernaufrufe in das bestehende System

Hinweise:

- Auf der Serverseite sind keinerlei Änderungen an der bestehenden Implementierung notwendig!
- Futures kommen auch bei Methoden ohne Rückgabewert zum Einsatz. Sie repräsentieren in diesem Fall das Ende der Ausführung.

1.3 Testen der Implementierung (für alle)

Im nächsten Schritt soll die Behandlung von asynchronen Aufrufen durch das Fernaufrufsystems (mit *JUnit*) getestet werden. Hierzu sind Testfälle zu implementieren mit deren Hilfe sich das Verhalten der Futures überprüfen lässt. Aufgrund der Tatsache, dass asynchrone Fernaufrufe ihre Vorteile vor allem bei Methoden mit langer Bearbeitungszeit ausspielen können, sind diese bevorzugt zu betrachten.

Aufgaben:

→ Implementierung geeigneter Testfälle für asynchrone Fernaufrufe

1.4 Evaluation des Fernaufrufsystems (optional für Nebenfächler)

Abschließend soll mittels geeigneter Szenarien das im Rahmen der Übung implementierte Fernaufrufsystem evaluiert werden. Dabei ist für einen Fernaufruf im VS-System anhand von Messergebnissen ein Vergleich zu folgenden „Konkurrenten" zu ziehen:

1. Lokaler Methodenaufruf
2. Fernaufruf mit *Java-RMI*

Als Ausgangspunkt kann hierbei der im Verzeichnis */proj/i4vs/pub/a6* abgelegte *JUnit*-Test *VSEvaluation* dienen. Dieser stellt ein rudimentäres Grundgerüst zur Verfügung, das sich zur Bestimmung relevanter Vergleichswerte erweitern lässt. Folgende Größen stehen dabei im Mittelpunkt des Interesses:

1. Dauer eines Lookups
2. Antwortzeit eines Methodenaufrufs
3. Erreichbarer Durchsatz von Methodenaufrufen

Um sicherzustellen, dass die ermittelten Werte ein möglichst präzises Abbild der Realität darstellen, sind die Messungen mehrfach durchzuführen und Durchschnittswerte zu bilden. Die auf diese Weise erhaltenen Ergebnisse jeder Variante sollen in einer (kurzen) Auswertung (Diagramme!) präsentiert und einander gegenüber gestellt werden.

Aufgaben:

→ Durchführung, Auswertung und Vergleich von Messungen der drei Aufrufvarianten

Hinweise:

- Jede Bildschirmausgabe (z.B. zu Debug-Zwecken) kostet Zeit und verfälscht damit die Messungen unnötig.
- Um eine gemeinsame Vergleichsbasis zu schaffen, sind jegliche Mechanismen zur Fehlererzeugung (aus Aufgabe 5) abzuschalten und nur der fehlerfreie Fall zu betrachten.
- Minimale Antwortzeit und maximaler Durchsatz sind in den meisten Fällen sich gegenseitig ausschließende Ziele. Daher sind zu ihrer Bestimmung üblicherweise unterschiedliche Evaluationsszenarien notwendig.
- Sollte sich herausstellen, dass das eigene System in sämtlichen Bereichen grandios unterlegen ist, ist dies mit Fassung zu ertragen.

1.5 Abgabe: bis 15.7.2008, 12:00 Uhr

Die für diese Teilaufgabe erstellten Testfälle sind in *vsue.tests* abzulegen und alle weiteren Dateien in einem Subpackage *vsue.future* zusammenzufassen.