Praktikum aus Softwareentwicklung 2

WS 2006/2007, LVA 365.009

Abgabetermin: Fr. 17. 11. 2005, 17:00 **Erreichbare Punkte: 24**

Übung 5 – JDK 1.5

Aufgabe 1: Mailling List Manager 14 Punkte

Entwickeln Sie einen einfachen Mailling Listen Manager. Die Email Addressen der Empfänger und die Maillinglist Gruppen sind über 2 Property-Files und/oder XML Property Files steuerbar. Der Mailling Listen Manager wird mit Angabe der beiden Property Files (eines für die Email Adressen, eines für die Gruppen) gestarted und wartet dann auf Benutzereingaben. Folgende Befehle werden unterstützt:

- quit → Beenden
- reload → lädt die Property Files neu für Änderungen der Gruppen oder Email Adressen.
- <Grp.Nr> → durch Eingabe einer Gruppennummer wird diese Gruppe als die aktive Gruppennummer gewählt.
- send → schaltet in den Textmodus und zeichnet alle nachfolgenen Eingaben bis zur Eingabe eines . (Punktes) in einer eigenen Zeile auf und versendet diesen Text dann an alle Teilnehmer der Gruppe. Das Versenden simulieren Sie einfach durch Ausgabe der Liste der Empfängeremails und den ersten Zeichen < 40 des Textes.

Beispiel:

Email.properties:

```
asommer = alexander.sommer@tiscover.com
bgates = bill.gates@microsoft.com
larry = larry@oracle.com
jamesg = james.gosling@java.com
```

Group.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="1">asommer,bgates</entry>
  <entry key="2">larry,asommer,jamesg</entry>
  <entry key="3">bgates</entry>
</properties>
```

Mailling Listen Manager:

```
java -cp . test.jdk15.MaillingMgr em Email.properties gr Group.xml
```

Startscreen:

```
Available Commands:
  quit
  reload
  <GrpNr>
  send
  .   to quit record mode
> 2
```

Eingabe: 2

```
Active Group is 2
> send
```

Eingabe: send

```
Record Mail:
> Hallo folks,
> How are …
> .
```

Eingabe: Hallo folks <ENTER> How are .. <ENTER> . <ENTER>

```
Mail send to Group 2 on 25 May 2004 at 22:38:10 with members
  alexander.sommer@tiscover.com
  larry@oracle.com
  james.gosling@java.com
> quit
```

Überlegen Sie sich eine geeignete Datenstruktur für das Speichern der Gruppen und zuuordnen der User.Eine gültige Email Adr. muss mind.1 zeichen vor dem @ und 3 Zeichen nach dem @ haben, wobei dieses Zeichen durch einen Punkt getrennt werden. Durch diese Anforderung ist eine Email Adresse nicht einfach als String zu implementieren sondern ein als ein eigenständiges Objekt.
Verwenden Sie für die Implementation das JDK 1.5 und versuchen Sie die neuen Features zu nutzen, wo es sinnvoll und möglich ist. Testen Sie ihr Programm mit einigen Userinteraktionen und geben Sie einige Testabfolgen mit dem Programm ab.

Aufgabe 2: MemoryTest 10 Punkte

Schreiben Sie unter Verwendung der neuen Mangament API ein kleines Speicher-Test Programm, welches den Speicherverbrauch beim Speichern verschiedener Objekte in Listen aufzeigt. Nehmen Sie dazu 2 verschiedene Collections und 2 Arrays, welche Sie mit einer bestimmten Anzahl von Objekten füllen und messen immer an 3 Stellen den Speicher: bei leeren Collections oder Arrays, bei halb gefülltem Zustand und am Schluß. Die Anzahl der Objekte soll beim Starten des Programms mitgegeben werden.
Beispiel:

```
  java -cp . test.jdk15.MemTest 100000

Mem at Startup:
```

```
Heap (init, used, committed, max)
2097152(2048K), 155448(151K), 2097152 (2048K), 67108864(65536K)
NonHeap
29556736(28864K), 12041968(11759K), 29818880 (29120K), 121634816(118784K)

After filling ArrayList with 50000 Strings:
Heap (init, used, committed, max)
209….
NonHeap …

After filling ArrayList with 100000 Strings:
Heap ….
NonHeap ….

After filling HashMap with 50000 Date Objects:
Heap ….
NonHeap ….

After filling HashMap with 100000 Date Objects:
Heap ….
NonHeap ….

After filling Array with 50000 Integers:
Heap ….
NonHeap ….

After filling Array with 100000 Integers:
Heap ….
NonHeap ….

ATTENTION: memory usage was reduced, perhaps GC was invoked?

After filling Array with 50000 BigDecimals:
Heap ….
NonHeap ….

After filling Array with 100000 BigDecimals:
Heap ….
NonHeap ….
```

Beobeachten Sie die Ausgaben ein wenig. Sie können den Garbage Collector GC zb. auch über `System.getRuntime().gc()` aufrufen. Probieren Sie den Speicher ihrer virtuellen Maschine (java –Xms16m –Xmx32m) zu beschränke und einen OutOfMemory zu erzeugen oder den Speicher sehr voll zu machen. Beobachten Sie ihre Tests und geben Sie ein kleines Testprotokoll mit ihren Aufrufen und ihren Beobachtungen ab. Es sollten zumindest 5 Testfälle gemacht werden. Probieren sie auch einmal einen Array oder Collection mit vielen gleichen Strings zu erzeugen: `new String(„Hallo“)` und vergleichen Sie die ergebnisse mit `new String(„Hallo“).intern();`.

Die Ausgangsklasen sind ist:

```
import java.lang.management.*;
MemoryMBean mbean = ManagementFactory.getMemoryMBean();
```

Dokumentieren Sie ihr Programm damit der Tutor ihre Tests verfolgen kann.