UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 2: Java RMI

(Zu bearbeiten bis 11.11.2019)

Client/Server-Programmierung
Wintersemester 2019/20

Aufgabe 1: Test Ihrer Kennung (zur Vorbereitung)

Testen Sie, ob die Ihnen zugeteilte Linux-Kennung ordnungsgemäß funktioniert:

- Loggen Sie sich auf dem Linux-Arbeitsplatz mit Benutzernamen und Passwort ein.
- Öffnen Sie ein Konsolen-Fenster und ändern Sie Ihr Passowrt mit Hilfe des Befehls `passwd`.
- Beenden Sie die Sitzung und wiederholen Sie die Login-Prozedur.

Machen Sie sich mit Ihrem KDE-Desktop vertraut.

- Suchen Sie sich ein Text-Editor (vi, joe, xcoral, xemacs, fte, kate, ...) und einen Datei-Manager aus, mit dem Sie am besten umgehen können.

Erstellen Sie ein Java-Programm, das „Hello World“ auf der Konsole ausgibt und speichern Sie es ab. Kompiliert wird auf den Labor-PCs mit Java 1.8 mittels `javac <Dateiname>.java` und ausgeführt wird mit dem Kommando `java -classpath . <Klassenname>`.

Aufgabe 2: Erweiterung der Börsenanwendung (für Interessierte)

Laden Sie sich ggf. zunächst den Code zum Aufgabenblatt von der CSP-Webseite herunter und entpacken Sie das Archiv. Erweitern Sie die im Verzeichnis `STOCK` vorgegebene Implementierung der Börsenanwendung um die folgenden Dienste, die in der Klasse `StockExchange` realisiert werden sollen:

1) **public WinnerInfo getWinner(int d1, int d2) throws StockException**

   Sucht die Aktie, die im Zeitraum zwischen den 2 Tagen `d1` (1...30) und `d2` (`d1+1`...31) den größten prozentualen Kursgewinn

   $$\frac{K(\texttt{d2}) - K(\texttt{d1})}{K(\texttt{d1})} \cdot 100\% \qquad \text{mit } K(\texttt{d1}) = \text{Kurs am Tag } \texttt{d1}$$

   erwirtschaftet hat.

   Ausgegeben wird ein Objekt der Klasse `WinnerInfo`, das den Namen der AG sowie den prozentualen Wert des Gewinns beinhaltet.

2) **public double tendence(int d1) throws StockException**

   Berechnet die allgemeine Wertentwicklung, indem alle Kurse an einem gegebenen Tag `d1` (1...30) aufsummiert werden. Das Gleiche geschieht für den Folgetag `d2=d1+1`. Der Trend wird als prozentuale Wertdifferenz zurückgegeben:

   $$\frac{\text{Sum}_{\texttt{d2}} - \text{Sum}_{\texttt{d1}}}{\text{Sum}_{\texttt{d1}}} \cdot 100\%$$

Die Methoden sollen dabei die selbst definierte Exception `StockException` werfen, wenn ein Fehler bei der Kommunikation mit der Datenbank auftritt oder keine passenden Daten in der Datenbank vorhanden sind.

Bearbeiten Sie die folgenden Aufgaben des Praktikums, die sich auf die Börsenanwendung beziehen, mit Ihrer erweiterten Version.

Aufgabe 3: Java-Client für die Börsenanwendung

Laden Sie sich ggf. zunächst den Code zum Aufgabenblatt von der CSP-Webseite herunter und entpacken Sie das Archiv. Sie finden im Verzeichnis `STOCK` die Serverklassen der Börsenanwendung, eine Hilfsklasse `ConnectDB` zum Zugriff auf die Datenbank, den Java-Treiber für MySQL, ein Testprogramm für den Datenbank-Zugriff sowie zwei SQL-Dumps der Datenbanktabellen (falls Sie sich die MySQL-Datenbank zuhause einrichten wollen).

Realisieren Sie dann ein einfaches User-Interface (UI) für die Börsenanwendung; entweder graphisch oder z.B. als Menü in der Textkonsole. Über das UI können Sie die Dienste auswählen sowie die notwendigen Parameter für den gewählten Dienst eingeben.

Testen Sie Ihr Programm mit einigen Testfällen, so daß jeder Dienst mindestens einmal verwendet wird. Denken Sie beim Start Ihres Programms daran, daß der MySQL-Datenbank-Treiber im Klassenpfad liegen muß. Sie sollten also beim Ausführen die Option[1]

```
-classpath /opt/dist/mysql-connector/mysql-connector-java-3.0.10-stable-b
in.jar:.
```

setzen, oder Sie definieren sich die Umgebungsvariable `CLASSPATH`:

```
export CLASSPATH=$CLASSPATH:/opt/dist/mysql-connector/mysql-connector-jav
a-3.0.10-stable-bin.jar:.
```

Mit dem Testprogramm `TestDB` können Sie sich zum Überprüfen auch die ganze Datenbank ausgeben lassen:

```
java -classpath /opt/dist/mysql-connector/mysql-connector-java-3.0.10-sta
ble-bin.jar:. TestDB | more
```

Aufgabe 4: RMI Hello World (zur Vorbereitung)

Laden Sie sich ggf. noch den Code zum Aufgabenblatt von der CSP-Webseite herunter und entpacken Sie das Archiv. Wechseln Sie ins Verzeichnis `HELLO-FIRST`. Sie finden dort zwei Unterverzeichnisse, eines für die Client-Dateien und das zweite für die Server-Dateien. Mit den zwei Verzeichnissen soll die getrennte Entwicklung von Client und Server, wie in der Vorlesung dargestellt, realisiert werden. Für den Ablauf der Entwicklung siehe Vorlesung „Verteilte Systeme“ Kapitel 3.2.

Testen Sie nun, ob der Client und der Server ordnungsgemäß funktionieren, wenn der Client und der Server getrennt auf verschiedenen Rechnern ausgeführt werden (beachten Sie, daß RMI-Registry und Server immer auf demselben Rechner laufen müssen). Dokumentieren und erklären Sie Ihre Vorgehensweise.

Hinweise:

- Starten Sie die RMI-Registry mit vorgegebener Portnummer (`rmiregistry <portnummer>`). Diese Portnummer bekommen Sie, indem Sie 500 gefolgt von den letzten beiden Ziffern ihres Logins nehmen (z.B. Login csp087 $\Rightarrow$ Portnummer: 50087). Verwenden Sie immer diese Portnummer, damit keine Kollisionen mit anderen Übungsteilnehmern auftreten.
- Mit `ssh <host>` können Sie sich auf einem anderen Rechner (`host`) einloggen. Ggf. müssen Sie sich dazu zunächst ein Schlüsselpaar erzeugen und den öffentlichen Schlüssel an `~/.ssh/authorized_keys` anfügen (siehe Aufgabenblatt 1).

Aufgabe 5: Börsenanwendung mit RMI

Erweitern Sie Ihre Implementierung der Börsenanwendung aus Aufgabe 3 (und ggf. Aufg. 2) zu einer Client/Server-Anwendung auf Basis von RMI.

Verwenden Sie die von Ihnen implementierte UI-Klasse als Basis für den RMI-Client. Die beiden Java-Klassen `StockExchange` und `Depot`, die die Dienste implementieren, sollen als serverseitige RMI-Remote-Objekte umstrukturiert werden. Testen Sie Ihre Implementierung mit mindestens zwei Instanzen des Clients.

[1] Der Zeilenumbruch dient hier nur der Formatierung und ist nicht mit einzugeben!