Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 13

Bearbeitungszeit: 07.07.2008 bis zur Klausur (Musterlösung erscheint ca. am 20.07.)

Aufgabe 1: Assembler (ausnahmsweise)

Mit dem Programmierprojekt hatten Sie sicherlich viel Spaß – aber noch nicht genug.

Die in der Vorlesung kennen gelernten **Exceptions** („Ausnahmen“) eignen sich nämlich hervorragend zur Fehlerbehandlung. In Assemblerprogrammen kann es z.B. vorkommen, dass das Register **R0** irrtümlich überschrieben wird oder durch 0 geteilt wird. Im Programmierprojekt musste sich Ihr Simulator darum noch nicht kümmern, jetzt wollen wir das ändern.

Als Vorgabe finden Sie auf unserer Webseite die Musterlösung des Programmierprojekts, bitte verwenden Sie nicht Ihr eigenes Projekt.

Bei allen aufgetretenen Fehlern wollen wir eine aussagekräftige Fehlermeldung ausgeben, die auch den gerade ausgeführten Befehl enthält, z.B. „**`Fehler bei der Ausfuehrung des Befehls MOD 1 2 3: Division durch 0!`**“.

a) Da alle möglichen Fehlermeldungen viel gemeinsam haben, schreiben Sie zuerst eine **abstrakte** Exception-Klasse (**`extends Exception`**) **`SimulatorException`**.

b) Diese soll mittels vier **`int`**-Variablen **`op`**, **`a`**, **`b`** und **`c`** den aktuellen Befehl und mittels einer **`String`**-Variablen **`error`** die spezifische Fehlermeldung speichern können. Deklarieren Sie alle fünf Attribute als **`protected`**, damit spätere Erben der Klasse darauf zugreifen können.

c) Fügen Sie einen Konstruktor hinzu, der vier **`int`**-Parameter erwartet und die vier **`int`**-Attribute damit initialisiert. Die Variable **`error`** müssen Sie noch nicht initialisieren.

d) Schreiben Sie nun die Exception-typische Methode **`public String getMessage()`**, die eine aussagekräftige Fehlermeldung zurückgeben soll (siehe oben). Den Befehlsnamen bekommen Sie aus dem OpCode mittels **`Parser.opcodes[op]`**, der Teil nach dem Doppelpunkt ist einfach die Variable **`error`**.

e) Erstellen Sie nun eine Klasse **`DivByZeroException`**, die von **`SimulatorException`** erbt (und damit auch eine Exception-Klasse ist). Der Konstruktor soll auch wieder vier **`int`**-Parameter erwarten und diese mittels super(…) an den Konstruktor der Überklasse geben. Die Variable **`error`** soll nun mit einer sinnvollen Fehlermeldung initialisiert werden, z.B. „Division durch 0!“.

f) Die Klasse **`RegisterZeroException`** soll analog zur Klasse **`DivByZeroException`** erstellt werden, nur die Fehlermeldung unterscheidet sich (z.B. „Register 0 wird ueberschrieben!“).

Nach diesen Vorarbeiten müssen wir uns noch um die eigentliche Fehlerbehandlung kümmern. Dazu sind drei Schritte nötig:

g) In der **`main`**-Methode der Klasse **`Simulator`** müssen Sie dafür sorgen, dass das Programm abgebrochen und die richtige Fehlermeldung (**`getMessage()`**) ausgegeben wird, wenn ein Fehler der Art **`SimulatorException`** auftritt – Sie brauchen um den **`execute()`**-Aufruf also einen **`try`**-**`catch`**-Block.

h) Dem Compiler müssen Sie mitteilen, dass in der Methode **`execute()`** eine **`SimulatorException`** auftreten kann – dazu müssen Sie nicht die eigentliche Methode ändern, sondern nur zwei Wörter zum Methodenkopf hinzufügen.

i) Außerdem müssen Sie sich natürlich darum kümmern, dass die Exceptions auch geworfen werden, wenn die zugehörigen Fehler auftreten. bei den vier Divisionsbefehlen können beide Fehler auftreten – dabei soll die Division durch 0 „Vorrang“ haben. Die **`RegisterZeroException`** kann in allen Fällen auftreten, wo ein Register beschrieben wird. Vermutlich sparen Sie sich Tipparbeit, wenn Sie die komplette Fehlerbehandlung vor der eigentlichen Ausführung der Befehle vornehmen.

Die in der Vorgabe enthaltenen Testprogramme **`test2.txt`** bzw. **`test3.txt`** testen die Division durch 0 (0 eingeben!) bzw. das Überschreiben des Registers **`R0`**.

Wenn Sie wollen, können Sie auch noch eine **`RegistersException`** schreiben und verwenden, die den Fehler abfängt, dass auf ein Register >31 zugegriffen wird.

Viel Spaß – und viel Erfolg in der Abschlussklausur!

Wenn Sie Fragen zu dieser oder anderen Übungen haben, steht Ihnen das Forum natürlich auch während der vorlesungsfreien Zeit zur Verfügung!