Anleitung zum Programm „Taschenrechner“.

Vorwort

Zunächst einmal: Seien Sie nicht frustriert, wenn Sie plötzlich das Gefühl haben, nichts mehr zu verstehen. Die Programmierung graphischer Benutzeroberflächen mit der „Swing“-API von Oracle ist nicht einfach: Es werden viele Konzepte genutzt, die zunächst einmal verstanden und ausprobiert werden müssen, und es werden viele Klassen in großen, unübersichtlichen Vererbungshierarchien kombiniert. Wenn Sie sich daher in Swing „verirren“, dann ist das völlig normal!

Um es Ihnen jedoch wieder etwas leichter zu machen, sind die Swing-Methoden und -Klassen in diesem Programm relativ gut gekapselt, so dass Sie sich damit nicht auseinander setzen müssen, aber gleichzeitig etwas damit „spielen“ können (s.u.). Außerdem sind alle Klassen und Methoden ausführlich dokumentiert, so dass Sie mit der „eigentlichen“ Hausaufgabe keine Schwierigkeiten haben sollten. Pro Klasse, die Sie implementieren sollen, müssen Sie weniger als 10 Zeilen Code schreiben, und den können Sie zu 90% per „Copy & Paste“ aus bereits existierenden Klassen übernehmen.

Um das Programm auszuführen, müssen Sie natürlich wissen, wo die main-Methode definiert ist – dies ist die Klasse „Application.java“.

Ihre Aufgabe besteht darin, den Taschenrechner um einige nützliche Operationen zu erweitern. Bis jetzt kann er ausschließlich Addieren und Multiplizieren sowie die Potenz einer Zahl berechnen, aber es wäre doch sehr schön, wenn er außerdem noch **Subtrahieren** und **Dividieren** könnte! Dies sollen Sie übernehmen, wobei es natürlich völlig freigestellt ist, weitere oder andere Operationen zu implementieren (Allerdings sind nur zweistellige Operationen möglich, einstellige Operationen wie bspw. die Quadratwurzel lassen sich nicht berechnen).

Die *JavaDoc* zu diesem Programm finden Sie im Verzeichnis „doc“. Öffnen Sie die Datei „index.html“ in einem Browser, damit Sie die

komplette Dokumentation browsen können (und nicht jede Datei einzeln öffnen müssen).

Noch ein wichtiger Hinweis: Die Hausarbeit, die Sie am Ende des Semesters schreiben werden, wird vom Prinzip her ähnlich aufgebaut sein wie dieses Programm – Auch dort werden Sie Klassen in verschiedenen Packages ablegen, mit JavaDoc umgehen müssen, abstrakte Klassen konstruieren oder als Basisklasse verwenden, Interfaces implementieren etc. Allerdings wird die Hausarbeit natürlich umfangreicher werden als diese Hausaufgabe. Daher sollten Sie diese Aufgabe auch als Vorbereitung für die Hausarbeit betrachten, und hin und wieder mal ansehen (insbesondere die „fakultativen Hausaufgaben" weiter unten).

Erste Hausaufgabe (obligatorischer Teil):

Um eine neue Taste zu implementieren, müssen Sie zwei Dinge tun:

(1) Die „Logik" der Taste implementieren
(2) Die neue Funktion am Taschenrechner „anmelden".

Wie Schritt (1) bewältigt werden kann, können Sie an bereits existierenden Tasten sehen:

Im Package `de.uni_koeln.spinfo.calculator.operationbuttons` finden Sie die Tasten für +, * usw. Diese sind alle von der Klasse `AbstractButtonAction` im gleichen Package abgeleitet.

Wie Sie sehen werden, müssen Sie lediglich einen passenden Konstruktor erstellen sowie die Methode `executeOperation(long parameter1, long parameter2)`, die in der Superklasse abstrakt ist, implementieren. Damit haben Sie gleichzeitig ein sehr gutes Beispiel für abstrakte Klassen, denn der Implementationsaufwand für eine neue Operations-Klasse ist minimal, weil sämtliche Funktionalität, die die Operationen teilen, in der Basisklasse enthalten ist.

Das folgende *UML-Klassendiagramm* zeigt die Ableitungshierarchie für

die bereits existierenden Operator-Klassen, das UML-Sequenzdiagramm veranschaulicht, was eigentlich passiert, wenn eine Operator-Taste gedrückt wird.

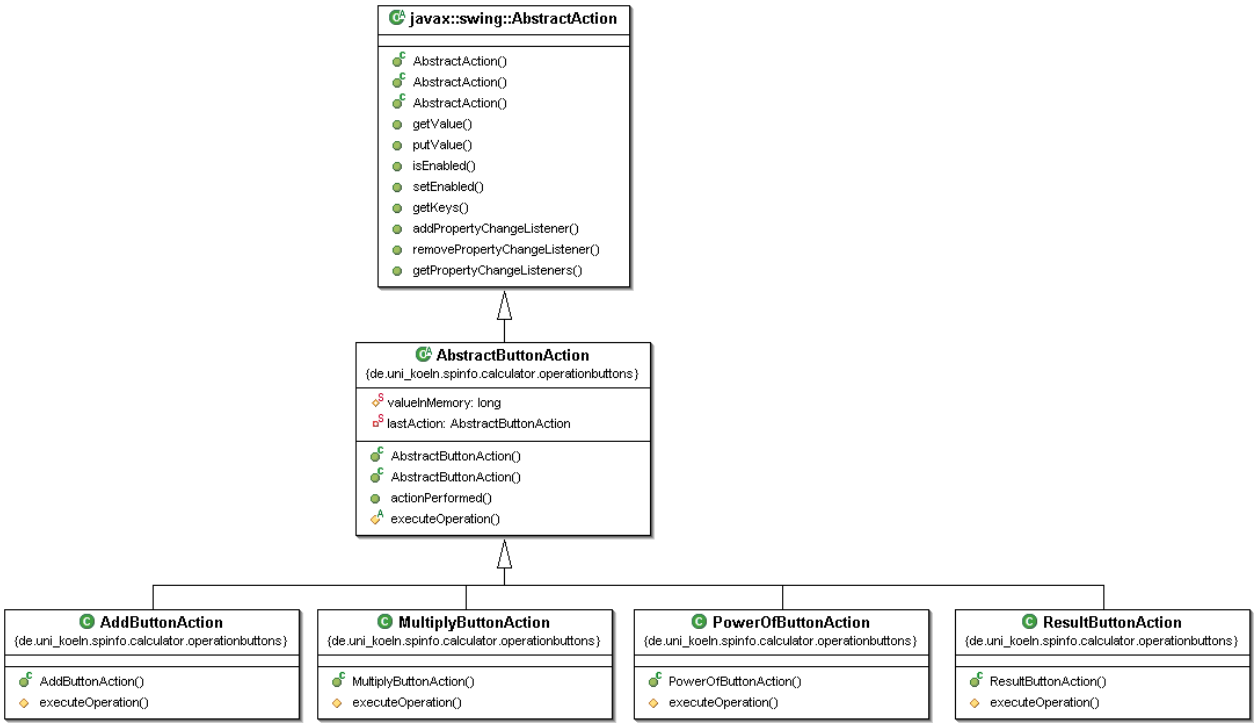

Abbildung 1 - Die Klassenhierarchie von Operator-Klassen (im Projekt enthalten)

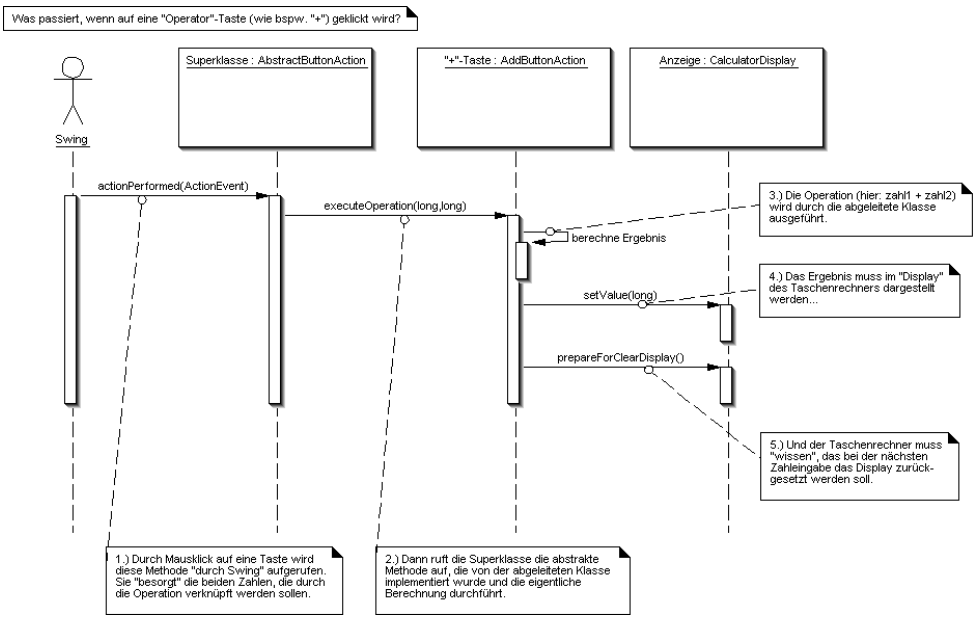

Abbildung 2 - Sequenzdiagramm zur Veranschaulichung des Programmablaufs bei Betätigung einer Operator-Taste (ebenfalls im Projekt enthalten)

Um Schritt (2) zu realisieren, müssen Sie eine Instanz Ihrer Klasse in der Methode `initialize()` in der Klasse `OperatorsPanel` im Package `de.uni_koeln.spinfo.calculator.panels` hinzufügen. Dazu genügt eine Zeile Code, auch hier können Sie sich an den bereits existierenden Operatoren orientieren.

Damit ist ihr Taschenrechner schon fast perfekt, allerdings sollten Sie die abstrakte Methode noch etwas verbessern: Wenn Sie sich den Methodenkörper der `executeOperation()`-Methoden in den Operationen-Klassen ansehen, werden Sie feststellen, dass in jeder Klasse einige Anweisungen ausgeführt werden, die auch in jeder anderen Klasse ausgeführt werden. Damit bietet es sich an, diese Anweisungen in die Superklasse auszulagern, so dass Sie schließlich in jeder `executeOperation()`-Methode nur noch eine einzelne Anweisung ausführen müssen.

Die Methode `actionPerformed()` in der Klasse `AbstractButtonAction` ruft `executeOperation()` auf, an dieser Stelle müssen Sie eingreifen... und eigentlich nichts weiter tun, als zwei Zeilen einzufügen! (Das Sequenzdiagramm zeigt schon recht deutlich, welche Anweisungen gemeint sind).

Damit sind Sie dann auch schon fertig! Allerdings sollten bzw. können Sie sich noch einige Stellen im Code ansehen, die Konzepte, die Sie bereits kennen, an einem konkreten, weniger weltfremden Beispiel veranschaulichen...

Zweite Hausaufgabe (fakultativer Teil)

Sehen Sie sich den Code und die JavaDoc an!

So finden Sie beispielsweise in der Klasse `CalculatorDisplay` einige interessante Auffälligkeiten, wie einen als `private` markierten Konstruktor – wie kann das sein?

Im Package `de.uni_koeln.spinfo.menu` können Sie in der Klasse `AboutAction` den Text, der bei Aufruf des Menüs „Über diesen Taschenrechner“ dargestellt wird, modifizieren – plötzlich erscheint. **Swing ganz einfach!**

In der Klasse `QuitAction` finden im selben Package finden Sie einen sinnvollen Einsatz der Methode `System.exit()`.

In der Klasse `AbstractButtonAction` finden Sie zwei statische Klassenvariablen, die sinnvoll eingesetzt werden – versuchen Sie, ggf. mit dem Debugger, den Programmablauf bei Aufruf der Methode `executeOperation` nachzuvollziehen. Was würde passieren, wenn diese Variablen nicht statisch wären?

Neben jeder Klasse ist auch jedes Package kommentiert – wenn Sie die JavaDoc ansehen, und im Bereich oben rechts ein Package auswählen, können Sie im Bereich darunter noch einmal auf den Packagenamen klicken. Dann erscheint im Fenster rechts die Package-Dokumentation (ist leider etwas umständlich...) und dort finden Sie eine Beschreibung des Packages, sowie (meistens) einen Verweis auf ein Tutorial von Oracle, dass das entsprechende Swing-Konzept und den Umgang damit erläutert (wie bspw. Buttons, Menüs oder Panels). Wenn Sie möchten, können Sie es sich einmal ansehen – oder auch einfach nur merken, damit Sie später, wenn Sie etwas mehr Routine im Programmieren haben und die Syntax der Sprache kein Problem mehr für Sie ist, einen Anlaufpunkt kennen, um in die GUI-Programmierung einzusteigen.

Klassen, die Sie sich nicht ansehen sollten...

1) Alle Klassen, die aus dem Package `javax.swing.*` stammen, müssen Sie sich nicht ansehen, denn diese werden Sie vermutlich etwas erschlagen (siehe Bemerkung oben zu Swing). Meistens beginnen die Namen dieser Klassen mit „J“, wie etwa `JButton`, `JPanel` usw. Für die „Logik“ des Taschenrechners sind sie nicht relevant und im Kurs werden wir nicht weiter mit ihnen arbeiten, daher müssen Sie sie auch nicht kennen lernen (außer natürlich aus

eigenem Interesse...).

2) Aus ähnlichen Gründen sollten Sie auch die Klassen `CalculatorFrame` und `CalculatorWindowListener` ignorieren bzw. nur oberflächlich betrachten.

Modifizieren Sie den Code und erweitern Sie ihn, wie Sie möchten

Ein anderer Vorteil von Objektorientierung ist die Tatsache, dass Sie viele Instanzen einer Klasse anlegen können. So ist beispielsweise jede Zahlentaste des Taschenrechners eine Instanz der Klasse `NumberButton` im Package `de.uni_koeln.spinfo.numberbuttons` und in der Methode `initialize()` der Klasse `NumberButtonsPanel` im Package `de.uni_koeln.spinfo.panels` sehen Sie, wie einfach die Tasten erzeugt werden können. Dort könnten Sie – da in der Oberfläche des Taschenrechners ja noch Platz ist – noch zwei weitere Tasten hinzufügen, und sich einen einzigartigen Taschenrechner bauen, der neben den „langweiligen" Tasten von 0 bis 9 auch noch die Taste „-13" besitzt...

Wenn Sie sich den Konstruktor der Klasse `PowerOfButtonAction` ansehen, werden Sie feststellen, dass ein anderer Konstruktor benutzt wird als bei den anderen Operatoren. Hier wird ein Icon übergeben, das auf der Taste statt eines Strings dargestellt wird. Wenn Sie möchten, können Sie auch für andere Tasten ein Icon statt eines Texts benutzen...

Sehr fakultativ und schwierig:

Wie Sie sicher bereits festgestellt haben, fehlt diesem Taschenrechner eine wichtige Eigenschaft, denn er kann nicht mit `float-` oder `double`-Werten umgehen. Der Grund dafür ist der, dass ich keine Lust hatte, mich um die „Komma"-Taste zu kümmern, denn natürlich darf nur dann ein Komma hinzugefügt werden, wenn noch keins vorhanden ist... Aber vielleicht haben Sie ja eine Idee, wie sich dies realisieren ließe?

Natürlich müssten die Variablen der beiden Parameter speichern statt `long` vom Typ `double` sein. Ebenso müssten die entsprechenden Methoden angepasst werden... Insgesamt müssten Sie an einigen Stellen des Programms Änderungen durchführen, aber dann haben Sie auch eine Menge gelernt!

Nachwort:

Wenn Sie diesen Text bis zu dieser Stelle gelesen haben, haben Sie bereits die meiste Arbeit hinter sich, zumindest, wenn Sie sich auf den ersten Teil der Aufgabe beschränken. Alles weitere bleibt letztlich Ihnen überlassen – mit diesem Programm besitzen Sie ein relativ einfaches, aber überdurchschnittlich gut dokumentiertes, „echtes“ Programm, das Sie als Ausgangspunkt für eigene Programme nehmen können, das hoffentlich auch das eine oder andere Konzept veranschaulicht, und das – ebenfalls hoffentlich – zum Experimentieren einlädt