Karlsruhe Institute of Technology

Institut für Angewandte Informatik und Formale Beschreibungsverfahren

Prof. Dr. D. Ratz

Philipp Stegmaier, Nikolai Polley
E-Mail: philipp.stegmaier@kit.edu

Programmierung kommerzieller Systeme – Anwendungen in Netzen mit Java

Sommersemester 2022

Lösungshinweise zu Übungsblatt P04

(Bearbeitung im Rechnerpraktikum in der Zeit vom 30.05.2022 bis zum 03.06.2022)

Lernziele:

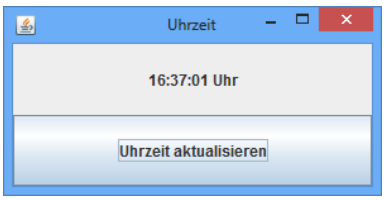
- Einführung und Vertiefung der Ereignisverarbeitung
- Kennenlernen und üben des MVC-Ansatzes

Theorie zum MVC-Ansatz: Bei graphischen Oberflächen kann es schnell unübersichtlich werden, wenn Sie die Programmlogik (zum Beispiel Reaktionen auf Benutzereingaben) und die eigentlich Darstellung (GUI) in einer einzelnen Klasse implementieren. Für umfangreichere Programmierprojekte hat sich deshalb das *MVC-Pattern* durchgesetzt.
Dabei werden Klassen nach einem bestimmten Muster in drei verschiedene Pakete unterteilt: *Model*, *View* und *Controller*. Klassen, die der Darstellung des User-Interfaces und der Interaktion mit dem Nutzer dienen, sind im Paket *View* zusammengefasst. Klassen, die das Datenmodell repräsentieren liegen im Paket *Model*. Im Paket *Controller* ist die Programmlogik implementiert. Da die `main`-Methode keinem der drei Bereiche zuzuordnen ist, liegt sie in einem eigenen Paket. Um den Ansatz zu üben, implementieren Sie die Aufgaben 3.3 bis 3.7 dieses Übungsblattes nach dem MVC-Ansatz.

Aufgabe P 4.1 * (Listener in vier Varianten)

In der Vorlesung haben Sie gelernt, dass es verschiedene Möglichkeiten gibt, Listener zu implementieren. In dieser Aufgabe sollen Sie nun diese verschiedenen Möglichkeiten ausprobieren und jeweils ein Beispiel implementieren.

Schauen Sie sich dazu zunächst folgende grafische Oberfläche an:



Im oberen Bereich des Fensters wird eine Uhrzeit angezeigt und im unteren Bereich befindet sich ein Button. Bei Betätigen des Buttons soll die Uhrzeit aktualisiert werden, sodass die aktuelle Zeit in dem im Bild dargestellten Format angezeigt wird.

Erstellen Sie das User Interface und implementieren Sie die Ereignisbehandlung

a) als innere Klasse,

b) als anonyme Klasse,

c) als separate Klasse und

d) innerhalb der Hauptklasse, die das Frame darstellt.

Aufgabe P 4.2 * (Ereignisverarbeitung)

a) Schreiben Sie ein Programm *ButtonFrame*, dass ein Fenster mit einem `JButton` erstellt und mit einer zufällig ausgewählten Hintergrundfarbe versehen wird. Die RGB-Werte der Farben sollen als Tooltips erscheinen, wenn man mit dem Mauszeiger über den Button fährt. Verwenden Sie dazu kein normales `JButton`-Objekt, sondern ein Objekt vom Typ `ColorButton`, einer selbst zu implementierenden Klasse, deren Konstruktor eine Methode `changeColor()`aufruft, die dafür sorgt, dass das`ColorButton`-Objekt mit Zufalls-Farbe und Tooltips ausgestattet ist.

b) Falls die zufällig gewählten Farben nun zu unpassend erscheinen (z.B. rosa oder mint-grün) sollen sie mit einem Klick auf den Knopf geändert werden können.

   Damit der Knopf eine entsprechende Funktionalität hat, benötigen sie einen `ActionListener`, also eine Klasse, die das entsprechende Interface implementiert. Schreiben sie diese so, dass für den Knopf die `changeColor()`-Methode aufgerufen wird und registrieren sie eine solche Listener-Klasse für den Knopf.

Aufgabe P 4.3 ** (Swing-Komponenten, Ereignisverarbeitung)

Erstellen Sie eine Java-Programm mit grafischer Oberfläche, das jeweils das aktuelle Datum in drei unterschiedlichen Formaten anzeigen kann und nachfolgende dargestellte Oberfläche und Funktionalität haben soll.

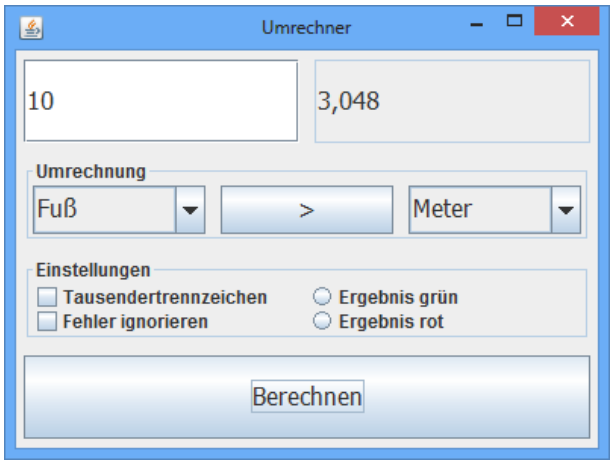
Der Frame soll also beim Start die im linken Bild dargestellte Form haben und der Benutzer soll durch Auswahl in der Klapptafel eine andere Darstellungsform für das Datum wählen können (z. B. eine Anzeige ohne die Jahreszahl).

a) Verwenden Sie in Ihrem Programm die privaten Variablen `beschriftung` und `datumsAnzeige` vom Typ `JLabel`, `formatAuswahl` vom Typ `JComboBox`, für die benötigten Swing-Komponenten und erstellen Sie das Layout des Fensters nach der gezeigten Abbildung.

b) Verwenden Sie in Ihrem Programm die privaten Variablen `datum` vom Typ `Date` sowie `kurz`, `mittel` und `lang` vom Typ `SimpleDateFormat` für die Darstellung des Datums. Ergänzen Sie den in a) programmierten Konstruktor so, dass das Datum erzeugt und für die Beschriftung des `datumsAnzeige`-Labels verwendet wird. Außerdem soll bei der Auswahl-Klapptafel ein Event-Listener registriert werden, den Sie als Objekt des Controllers `AnzeigeController` erzeugen können.
   Diese Klasse muss das Interface `ItemListener` implementieren und die Methode `itemStateChanged` so überschreiben, dass bei Änderung der Auswahl der entsprechende Auswahl-Index des `JComboBox`-Objekts bestimmt und (abhängig von dessen Wert) das entsprechende Darstellungsformat für das aktuelle Datum gewählt und zur Beschriftung des `datumsAnzeige`-Labels verwendet wird.

Aufgabe P 4.4 *** (Swing-Komponenten, Ereignisverarbeitung)

Trotz Ihres harten Studierendenalltags verreisen Sie sehr gerne. Dabei kommen Sie jedoch immer wieder mit den Längeneinheiten verschiedener Reiseziele (Fuß, Yard, Zoll etc.) und Reisemittel (Seemeile) durcheinander. Um dieser Problematik Herr bzw. Frau zu werden, möchten Sie sich ein kleines Tool programmieren, mit dem Sie die verschiedenen Einheiten ineinander umrechnen können. Nach reiflicher Überlegung haben Sie sich folgenden grafischen Aufbau des User Interface überlegt:



a) Erstellen Sie eine graphische Oberfläche gemäß dem oberen Bild. Wählen Sie dabei für den oberen Bereich des Fensters zwei Textfelder, um später Eingaben machen zu können. Nur eines davon soll editierbar sein (wählen sie als default-Einstellung das linke Textfeld als das editierbare aus).

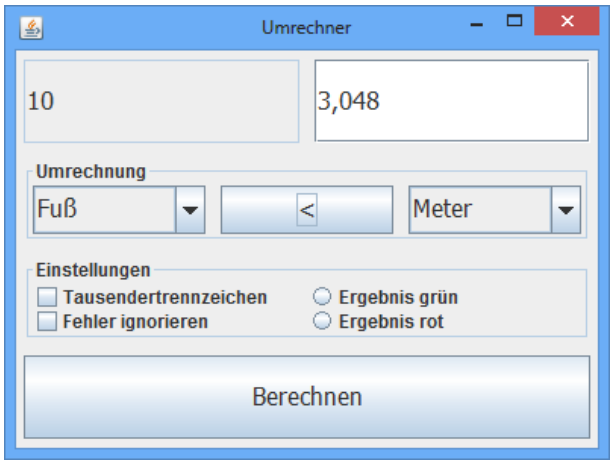
Um einem Panel einen Rahmen mit Titel hinzuzufügen (z.B. Panel „Umrechnung“) stellen Objekte der Klasse `JPanel` die Instanzmethode `setBorder()` bereit:

```
setBorder(new TitledBorder(null, "TITEL", TitledBorder.LEADING,
    TitledBorder.TOP, null, null))
```

Die Zeichenkette „TITEL“kann dabei durch einen beliebig zu wählenden Titel ersetzt werden.

b) Ergänzen sie die Ereignisverarbeitung zu der in Teilaufgabe a) erstellten graphischen Oberfläche. Im oberen Bereich des Fensters sind zwei Textfelder, wobei nur eins editierbar ist. Im Bereich direkt darunter (Umrechnung) finden Sie zwei Dropdown-Listen, in denen die verschiedenen Längeneinheiten aufgelistet werden. Der Button in der Mitte zeigt an, in welche Richtung die Umrechnung erfolgt. Zeigt er als Beschriftung „>“an, wird von der linken in die rechte Einheit umgerechnet. Durch einen Klick auf den Button verändert sich die Beschriftung zu „<“und es wird von der rechten in die linke Einheit umgerechnet. Die beiden Textfelder aus dem oberen Bereich müssen immer entsprechend eingestellt werden, sodass das Feld mit dem umzurechnenden Wert editierbar und das Feld des Zielwertes nicht editierbar (und somit grau dargestellt) ist.

Nach einem Klick auf den Richtungsbutton in der Mitte, sieht das Fenster beispielsweise so aus:

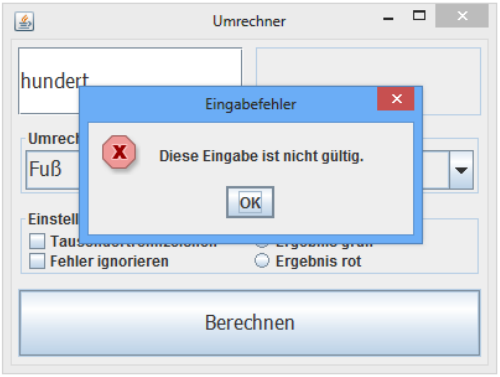


Für die Ein- und Ausgabe in den Textfeldern soll das Komma als Dezimaltrennzeichen verwendet werden. Um einen String mit diesem Format einlesen zu können, verwenden Sie beispielsweise folgendes Code-Fragment:

```
NumberFormat format = NumberFormat.getInstance(Locale.GERMANY);
Number number = format.parse("1,234");
double d = number.doubleValue();
```

Die entsprechend formatierte Ausgabe erzeugen Sie mit dem Ihnen bekannten `DecimalFormat`-Objekt. Verwenden Sie für alle Ausgaben eine Dezimalzahl mit genau drei Nachkommastellen.

Der Bereich „Einstellungen“ besteht aus zwei Checkbox-Elementen und zwei Radiobuttons, deren Funktionalität jedoch **nicht** implementiert werden muss. Lediglich die „Fehler ignorieren“-Checkbox soll folgende Funktion bieten: Bei der Eingabe von Zahlen über Textboxen stehen Sie häufig vor dem Problem, dass fehlerhafte Benutzereingaben (z.B. nicht numerische Zeichenketten) zu Exceptions während der Programmausführung führen können. Bei aktivierter Checkbox soll das Programm solche Fehler ignorieren und einfach nichts tun. Ist die Checkbox dagegen nicht aktiviert, soll der Benutzer gegebenenfalls beim Klicken des „Berechnen“-Buttons über ein Dialogfeld gewarnt werden (abgesehen von dieser Warnung soll aber nichts weiter unternommen werden):



Hinweis: Sie können diesen Dialog selbst implementieren oder die Funktion `JOptionPane.showMessageDialog(...)` verwenden, über die Sie sich in der Dokumentation informieren können.

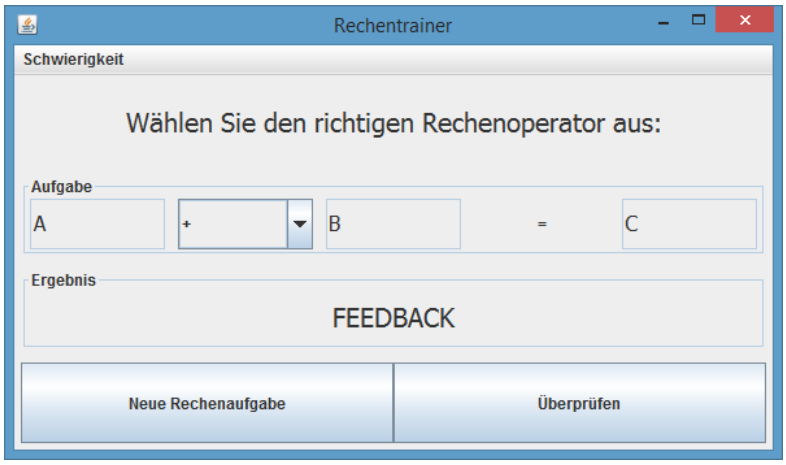
Die Berechnung selber startet erst nach Drücken des „Berechnen“-Buttons.

Laden Sie sich aus ILIAS die Vorgaben zu dieser Aufgabe herunter, betrachten Sie genau die bereits implementierten Klassen und vervollständigen Sie die Klasse `UmrechnerView` an den gekennzeichneten Stellen.

Aufgabe P 4.5 *** (Ereignisverarbeitung)

Als begeisterter Wiwi haben Sie sich ein aufregendes und prestigeträchtiges Hobby zugelegt: Kopfrechnen. Um Ihre außerordentlichen Fähigkeiten weiter auszubauen, möchten Sie sich ein kleines Trainingsprogramm schreiben, das Iher Skills im Addieren und Subtrahieren mit ganzen Zahlen perfektioniert.

Laden Sie sich zunächst die Vorgabe `Rechenspiel.zip` aus ILIAS herunter. Wenn Sie das Programm starten, erscheint folgendes Fenster:



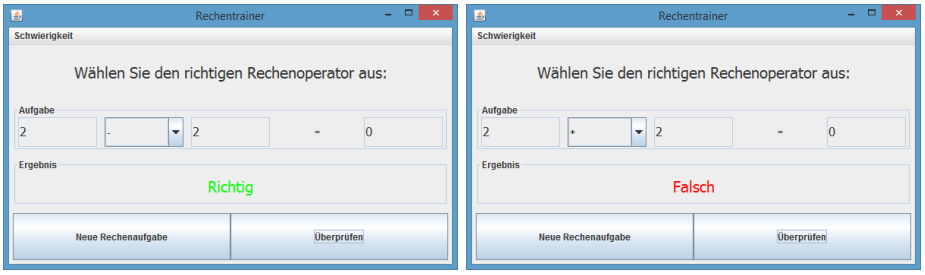
Bisher weisen sowohl die Buttons „Neue Rechenaufgabe“ und „Überprüfen“ als auch die `ComboBox` mit den Rechenoperatoren + bzw. - keinerlei Funktion auf. Außerdem lässt sich das Fenster nicht schließen. Ihre Aufgabe ist es nun, mittels Ereignisbehandlung folgende Funktionen umzusetzen:

a) Implementieren Sie die Methode `neueRechenaufgabe()` in Controller. Achten Sie darauf sie mit getter- und setter-Methoden mit den graphischen Elementen des Views zu verbinden.

Je nach im Menü „Schwierigkeit“ ausgewähltem Schwierigkeitsgrad sollen im Zahlenraum 1 bis 10 (Schwierigkeit „Einfach“) bzw. im Zahlenraum von 1 bis 100 (Schwierigkeit „Schwer“) zwei ganzzahlige Zufallszahlen `a` und `b` erzeugt werden. Diese werden dann zufällig entweder addiert oder subtrahiert. Die Zufallszahlen werden im Rechentrainer anstelle der Strings „A“ bzw. „B“ angezeigt, das Ergebnis der Rechenoperation soll anstelle des Strings „C“ angezeigt werden.

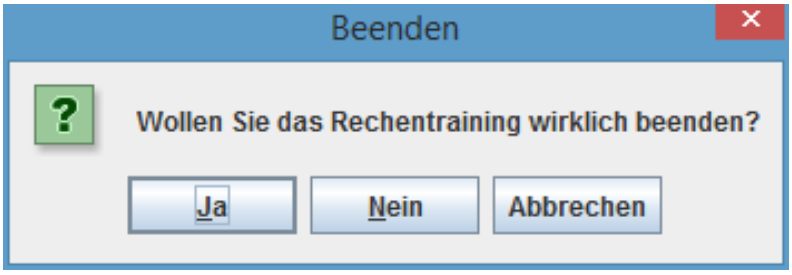
b) Beim jedem Klick auf den Button „Neue Rechenaufgabe“ soll die Methode `neueRechenaufgabe` aufgerufen werden. Setzen Sie die Listener-Klasse dazu als innere Klasse innerhalb des Controllers um.

c) Beim Klick auf den Button „Überprüfen“ soll abhängig vom in der ComboBox ausgewählten Rechenoperator anstelle des Strings „FEEDBACK“ entweder „Richtig“ (in der Farbe grün) oder „Falsch“ (in der Farbe rot) erscheinen. Implementieren Sie dazu die Methode `ueberpruefen()`, die wiederum von einem als innere Klasse umgesetzten Listener aufgerufen wird (in der Klasse Controller).



d) Beim Klick auf „Schließen“ (kleines Kreuz (Windows) bzw. roter Kreis (Mac) am oberen Fensterrand) soll sich vor dem Beenden des Rechentrainers eine Sicherheitsabfrage öffnen, die den Benutzer fragt, ob er den Rechentrainer wirklich schließen möchte. Nur beim Klick auf „Ja“ soll sich der Rechentrainer wirklich schließen. Beim Klick auf

„Nein“ oder „Abbrechen“ bleibt er weiterhin geöffnet.



Erstellen Sie dazu eine neue innere Klasse `BeendenListener` im Controller, die das `WindowListener`-Interface implementiert. Besonderes Augenmerk sollten Sie auf die Methode `public void windowClosing(WindowEvent e)` werfen, die auf das Schließen des Fensters reagiert. Informationen zur Verwendung des `WindowListener` finden Sie in der API-Dokumentation. Vergessen Sie nicht, eine Instanz des `BeendenListener` als `WindowListener` beim `RechentrainerFrame` zu registrieren.

Hinweis: Die Klasse `JOptionPane` bietet Methoden, um Dialogfenster mit Hinweistexten (Informationen, Warnungen oder Fragen) anzuzeigen. Dabei können die Dialogfenster auch verschiedene Buttons (z.B. „Ja“, „Nein“ oder „Abbruch“) anbieten, mit denen der Dialog beendet werden kann. für diese Aufgabe bietet sich die Klassenmethode `JOptionPane.showConfirmDialog` an, über deren Verwendung Sie Informationen in der API-Dokumentation finden können.

Viel Erfolg!