smartlearn
virtual learning. real success.

MODUL 226-2 OBJEKTORIENTIERT IMPLEMENTIEREN KAPITEL 5: EREIGNISSE BEHANDELN

Barbara Bielawski / Kurt Järmann / Andres Scheidegger

Kapitel 5: Ereignisse, Listeners, Adapter und Listen

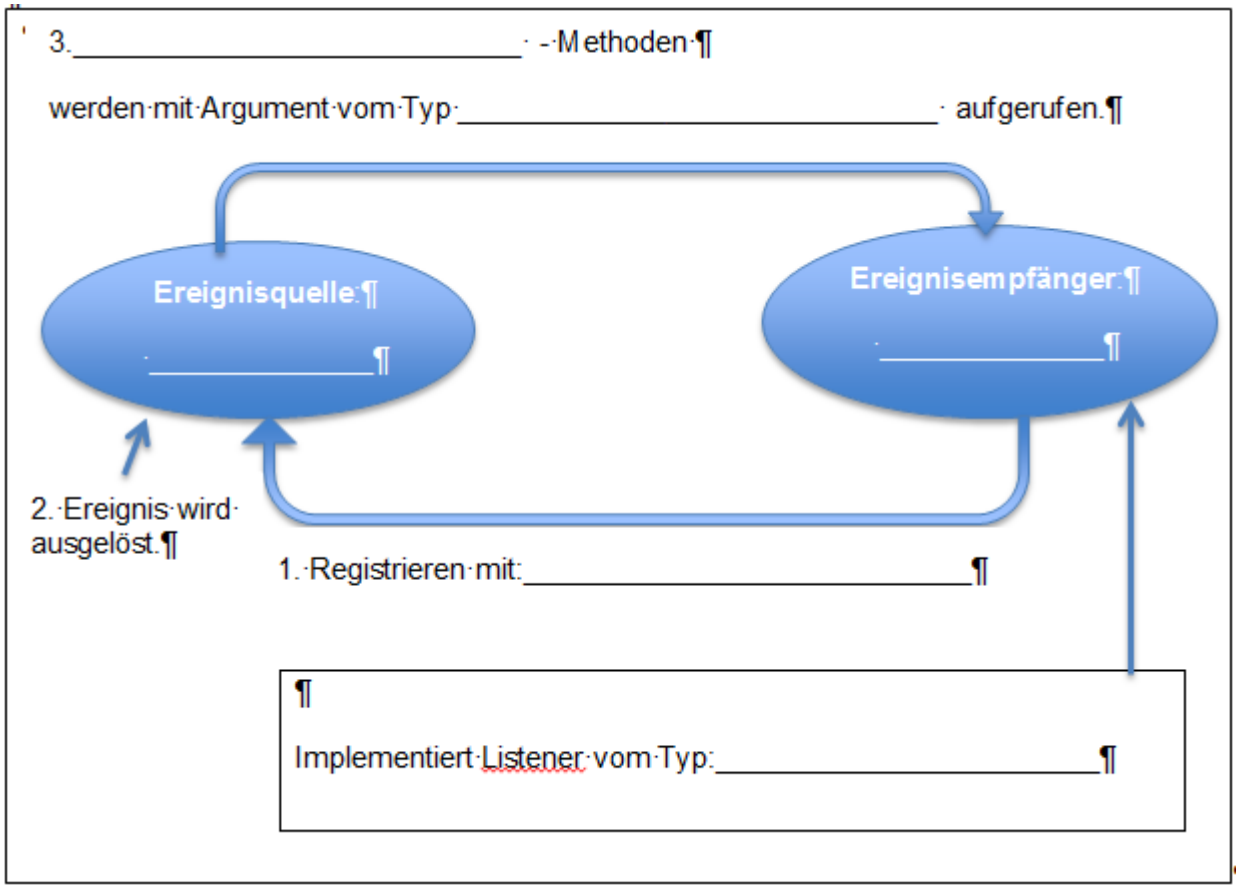
Aufgabe 1: : Ereignisbehandlung in Java

Einleitung:

Bei graphischen Benutzeroberflächen kann der Benutzer zu einem beliebigen Zeitpunkt die verschiedensten Aktionen auslösen. Zum Beispiel kann er einen Menüpunkt auswählen, er kann aber auch auf einen Knopf auf dem Bildschirm klicken, oder irgendetwas über die Tastatur eingeben. Da nicht vorhersehbar ist, was der Benutzer macht und wann er etwas macht, spricht man von *Ereignissen*, oder eben *Events*.

Für jedes Programm, welches über ein GUI verfügt, stellt sich nun das Problem, wie es mit all diesen möglichen Ereignissen umgeht. Dies soll im Folgenden zuerst auf einer allgemeinen Ebene und dann speziell für Java angeschaut werden.

Studieren Sie in der Präsentation „Ereignisse und Listen.ppt“ die 2. Seite und informieren Sie sich über den Ablauf von Events. Lesen Sie dazu die Kapitel 10.4 (Ereignisbehandlung) und 10.12 (Mausaktionen) in Ihrem Java-Buch durch.
Schauen Sie sich nun den Screencast „MouseListener.mp4“ an und ergänzen Sie folgende Grafik in Bezug auf den dort vorgestellten Ablauf:



Sie kennen nun die Möglichkeit, Event-Handling über die Implementierung von Listener-Interfaces zu realisieren. Ausser dieser Möglichkeit gibt es in Java noch andere Methoden, um Events zu behandeln.
Sehen Sie sich nun die beiden anderen Screencasts (MouseAdapter.mp4 und VerschachtelterMouseAdapter.mp4) an, um drei weitere Vorgehensweisen kennenzulernen. Lesen Sie bei Bedarf erneut im Buch die Details zu den Adapterklassen und zu Inneren Klassen (Kapitel 3.9).

Aufgabe 2: Ereignisbehandlung im Grafikeditor

Einleitung:

Der Grafikeditor ist im Moment eher ein „Grafikviewer“. Was fehlt ist die *Interaktivität* mit dem Benutzer. Das soll sich nun ändern.

Sie haben sich in Aufgabe 1 über vier verschiedene Möglichkeiten informiert, wie Event-Handling in Java implementiert werden kann. Eine dieser Möglichkeiten sollen Sie nun in Ihr Grafikeditor-Programm integrieren.

Sehen Sie sich den Screencast „Klassendiagramm Vorgabe.mp4“ an, damit Sie sich einen Überblick über die folgenden Aufträge verschaffen können.

- Integrieren Sie in Ihr bestehendes Grafikeditor-Programm die Klassen EditorFrame, EditorPanel, EditorControl und Grafikeditor. Sie finden diese im Ordner „Ressourcen“.
- Sie sollen Ihr Programm nun so ergänzen, dass die MousePressed und MouseReleased Ereignisse empfangen und bei der EditorControl-Klasse die passende Methoden aufgerufen wird. Überlegen Sie zuerst, welche der vier in den Screencasts beschriebenen Vorgehensweisen Sie umsetzen möchten, und integrieren Sie diese dann in Ihr Programm.
- Ergänzen Sie die Klasse EditorControl so, dass Rechtecke erzeugt werden können. Verwenden Sie die dazu vorbereitete Methode erzeugeFigurMitZweitemPunkt. Die erzeugten Rechtecke sollen in die Zeichnung hinzugefügt werden. Sollten Sie in der Klasse Zeichnung bisher einen Array verwendet haben, ersetzen Sie diesen durch eine Liste. Weitere Informationen zu Listen finden Sie in der Präsentation „Ereignisse und Listen.ppt“ oder im Buch im Kapitel 5.4.1.
- Implementieren Sie die Methode allesNeuZeichnen in der Klasse EditorControl, damit die Figuren gezeichnet werden. Damit das Zeichnen funktioniert, müssen Sie die repaint()-Methode des Panels aufrufen. Überlegen Sie in welchem Moment die Methode ausgeführt werden muss. Schauen Sie dazu noch einmal den Screencast zu den Vorgaben an oder orientieren Sie sich mit Hilfe des Buches (Kapitel 10.3).

Kompetenz 3.3

Aufgabe 3: Tastaturereignisse im Grafikeditor

Einleitung:

Der Grafikeditor soll nicht nur Rechtecke zeichnen können, sondern auch Linien, etc. Die Art der Figur, die gezeichnet wird, soll über die Tastatur wie folgt gesteuert werden können:

- Taste ‚r': Der Grafikeditor zeichnet ab jetzt Rechtecke
- Taste ‚l': Der Grafikeditor zeichnet ab jetzt Linien
- Taste ‚k': Der Grafikeditor zeichnet ab jetzt Kreise

- Ergänzen Sie die Klasse EditorFrame so, dass keyEvents empfangen werden können. Implementieren Sie dazu einen KeyListener oder leiten Sie von KeyAdapter ab, und übergeben Sie die gedrückte Taste an das Attribut figurTyp der Klasse EditorControl.
- Ergänzen Sie nun die Klasse EditorControl so, dass je nach Wert des Attributs figurTyp der Klasse EditorControl eine anderes Figur-Objekt erzeugt und bei der Zeichnung hinzugefügt wird.

Kompetenz 3.4

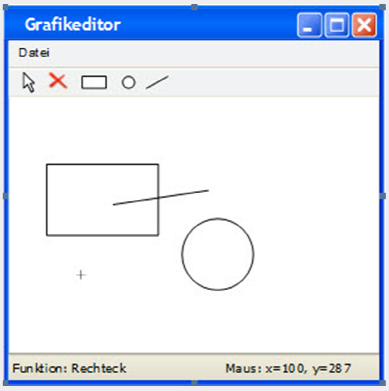
Können Sie nun mit Ihrem Grafikeditor-Programm mit Maus und Tastatur Rechtecke, Linien und Kreise gezeichnet werden, können Sie es zur Abgabe der Kompetenz 3.4 verwenden.

Vertiefungs- und Ergänzungsaufgaben

Die folgenden Aufgaben dienen Ihrer individuellen Vertiefung. Sie können diese nach Belieben bearbeiten und dadurch auch Zusatzkompetenzen erlangen. Die Aufgaben sind bewusst offen formuliert und Sie werden für die Lösung im Web recherchieren müssen.

GUI

Implementieren Sie ein komfortableres GUI für Ihren Grafikeditor. Das GUI könnte ungefähr wie folgt aussehen:



Es enthält:

- Menübalken
- Toolbar (im Bild mit weiteren Funktionen, siehe nächste Vertiefungsaufgabe)
- Statuszeile mit Angabe des aktuellen Werkzeugs
- Cursor, welcher je nach Werkzeug ändert
- Aktuelle Position der Maus
- Weitere Features nach Ihrem Geschmack

Im Buch finden Sie die nötigen Angaben, wie sie ein solches GUI mit Swing implementieren können. Stichworte sind *Layout-Manager*, *Panels*, *Buttons*, *Menü*. Weitere Hilfe finden Sie im Web.

Weitere Funktionen

Implementieren Sie folgende Funktionen:

- Selektieren und verschieben von Figuren
- Löschen von Figuren

Rubberbanding

Ergänzen Sie Ihren Grafikeditor so, dass beim Zeichnen einer Figur die Figur während dem Aufziehen mit der Maus temporär gezeichnet wird. Dasselbe soll geschehen, wenn eine Figur verschoben wird (sobald sie die Funktion Selektieren und Schieben implementiert haben). Die selektierte oder neu gezeichnete Figur soll dabei in einer anderen Farbe dargestellt werden.

Kompetenz 3.5

Alternativ zu anderen Aufgaben im Modul

Erweiterbarer Software-Design I

Vorbemerkung:

Die folgenden drei Vertiefungsaufgaben sind „Igelaufgaben“ und verlangen entsprechend Durchhaltevermögen. Als Lohn winkt Ihr erstes selbstgebautes Framework für Grafikeditoren.

Erweiterbar ist ein objektorientiertes Programm dann, wenn rein durch das Hinzufügen von einer oder mehreren neuen Klassen der Funktionsumfang des Programms zunimmt. Die bereits bestehenden Klassen sollen dabei nicht verändert, oder ergänzt werden müssen. (Man spricht auch vom Open-Closed-Prinzip: Eine Klasse ist offen für Erweiterung durch Ableitung aber geschlossen gegen Veränderung ihres eigenen Codes).

Können Sie Ihren Grafikeditor so umbauen, dass Sie mit ihm neue Figur-Typen zeichnen können, einzig indem Sie von bestehenden Klassen ableiten, ohne dass der bestehende Code angepasst werden muss?

Sie werden wahrscheinlich feststellen, dass die Klasse `EditorControl` im Moment den obigen Bedingungen nicht genügt. In der Methode `erzeugeFigurMitZweitemPunkt()` muss in einer switch-Anweisung entschieden werden, welche Art von Figur-Objekt erzeugt werden soll. Diese Anweisung muss für jeden neuen Figur-Typ um eine weitere case-Anweisung ergänzt werden. Genau solche switch-Anweisungen stehen also quer in der Landschaft, wenn es um Erweiterbarkeit geht.

Folgender Lösungsansatz kann hier helfen:
In der Methode `erzeugeFigurMitZweitemPunkt()` geht es um das Erzeugen von Figur-Objekten. Nun kann diese Aufgabe an eine Factory-Klasse, respektive ein Objekt davon, delegiert werden. Für jeden Figur-Typ gibt es eine Factory-Klasse. Alle Factory-Klassen sind von einer abstrakten Basis-Factory-Klasse abgeleitet. Diese Klasse deklariert die abstrakte Methode `createFigur()`. Deren Resultattyp ist die Klasse `Figur`. Das `EditorControl`-Objekt hat eine Referenz auf das aktuelle Factory-Objekt. Diese wird jeweils bei einem Tastendruck neu gesetzt.

Dieser Ansatz orientiert sich an den Design-Patterns *Factory-Methode* und *Abstract-Factory*.

Kompetenz 1.4 , Kompetenz 1.5

Alternativ zu anderen Aufgaben im Modul

Erweiterbarer Software-Design II

Wenn Sie die vorhergehende Vertiefungsaufgabe gelöst haben, werden Sie feststellen, dass wir das Problem mit der switch-artigen Struktur lediglich verschoben haben. Jetzt müssen wir nämlich bei der Bearbeitung des Tastatur-Events entscheiden, welches Factory-Objekt als aktuelle Factory gesetzt werden soll. Diese switch-Anweisung verarbeitet char-Werte, welche vom `KeyEvent` stammen. Diese Situation kann gelöst werden.

Der Ansatz dazu gleicht demjenigen in Vertiefungsaufgabe 2 des AB04. Jede Factory-Klasse weiss selber, welcher Buchstabe Ihr zugeordnet ist. Entsprechend wird in der Basis-Factory-Klasse eine Methode eingeführt, über welche man ein Factory-Objekt fragen kann, ob ein bestimmter char-Wert dem Buchstaben der Factory entspricht. Wenn dem so ist, wird dieses Objekt als aktuelle Factory im EditorControl-Objekt gesetzt.

Kompetenz 1.4 , Kompetenz 1.5

Alternativ zu anderen Aufgaben im Modul

Erweiterbarer Software-Design III

Um mit unserem Grafikeditor nun einen neuen Figur-Typ zeichnen zu können, müssen wir also eine neue von Figur abgeleitete Klasse und eine neue Factory-Klasse schreiben. Zudem müssen wir unserem Grafikeditor auf irgend eine Art mitteilen, dass da eine neue Factory-Klasse existiert, die zur Zeit der Programmierung der Klasse `EditorControl` noch nicht bekannt war. Wie können wir das tun, ohne die Klasse `EditorControl` zu verändern?

Dazu gibt es verschiedene Ansätze:

- Über eine Konfigurationsdatei, welche die Namen der zu verwendenden Factory-Klassen enthält.
- Über eine Konvention für die Namen der Factory-Klasen. Der Grafikeditor sucht dann zur Laufzeit in einem bestimmten Verzeichnis nach .class-Dateien, welche dieser Konvention entsprechen.

Bei beiden Ansätzen müssen anschliessend zur Laufzeit die Factory-Klassen von Hand geladen werden und anschliessend von diesen neu geladenen Klassen Objekte erzeugt werden.

Das Laden von Klassen zur Laufzeit geschieht über ein sogenanntes `ClassLoader`-Objekt. Diese erzeugt ein Objekt von der Klasse `Class`. Davon können Sie dann mit der Methode `newInstance()` ein Objekt erzeugen.

Kompetenz 3.5

Alternativ zu anderen Aufgaben im Modul