2 Programmentwicklung mit Java

Die Programmentwicklung mit Java wird in drei Schritten eingeführt:

- Zuerst wird die prinzipielle Arbeitsweise geschildert,
- danach wird das Zusammenspiel der erzeugten Module beschrieben, insbesondere die zeitliche Reihenfolge der Aktionen,
- und zuletzt werden einige Beispielprogramme gebracht, noch ohne genau auf formale Fragen einzugehen.

2.1 Die Arbeit mit Entwicklungsumgebungen

Java ist eine objektorientierte Programmiersprache, die von der Firma *Sun* entwickelt wurde. Sie ist nicht an ein Betriebssystem gebunden, sondern erzeugt einen Programmcode (*Bytecode*) für virtuelle Java Maschinen (*JVM: **J**ava **V**irtual **M**achine*), die für praktisch alle Betriebssysteme als *Laufzeitumgebung* zur Verfügung stehen. Um Java-Programme zu entwickeln, benötigt man eine entsprechende *Entwicklungsumgebung*, die mindestens den Java-Compiler (*javac*) enthält, der Java-Quelltexte in Bytecode übersetzen kann. Dieser ist im ***J**ava **D**evelopment **K**it* (*JDK*) neben der Laufzeitumgebung und anderen Werkzeugen enthalten.

Im Prinzip ist es möglich, Java-Programme mit einem normalen Texteditor zu erstellen und dann mithilfe von *javac* zu übersetzen. Wir wollen allerdings diesen Weg nicht gehen, weil wir dann auch wirklich alles im Editor selbst eingeben müssen – und zu „allem“ gehören standardisierte Programmteile, die genauso gut automatisch erzeugt werden können. Würden wir diese jeweils selbst schreiben, dann wären wir zu einem wesentlichen Teil der Arbeit mit Dingen beschäftigt, die mit dem eigentlich gestellten Problem nur wenig zu tun haben; und weil unsere Arbeitszeit beschränkt ist, würde dadurch natürlich auch der Umfang (und meist auch die Relevanz) der Problemstellung arg eingeschränkt. Vor allem aber würden wir auf die Hilfs- und Testmöglichkeiten verzichten, die moderne Computer leicht zur Verfügung stellen können.

Wir werden in diesem Buch grafische Entwicklungsumgebungen benutzen, die kostenlos im Internet erhältlich sind und sich in ihrer Funktionsweise stark ähneln. Dazu gehören derzeit die *NetBeans* der Firma *Sun*, die Personal-Version des *JBuilders* der Firma *Borland* und die Open Source Umgebung *Eclipse*. Alle diese Entwicklungsumgebungen werden schnell weiterentwickelt und ändern sich entsprechend in Umfang und Aussehen. Es kann deshalb nicht Aufgabe dieses Buches sein, ihre Installation und Arbeitsweise im Detail zu erklären. (Hierfür wird auf die aktuellen Online-Hilfen verwiesen.)

Stattdessen werden wir meist Javaklassen und -methoden schreiben, die weitgehend unabhängig von der Oberfläche, also den sichtbaren Teilen des entwickelten Programms – dem ***G****raphical* ***U****ser* ***I****nterface* (*GUI*) – sind. Das GUI gestalten wir mithilfe der benutzten Entwicklungsumgebung. In dieser legen wir die *Eigenschaften* der *Steuerelemente* (Buttons, Textfelder, …) fest und bestimmen, auf welche *Ereignisse* (Mausklicks, Tastatureingaben, …) sie reagieren sollen. Die hierfür erforderlichen Code-Teile werden von der Entwicklungsumgebung automatisch erzeugt. Erst danach beschreiben wir, was genau als *Reaktion* auf die Ereignisse geschehen soll. Meist wird das ein Methodenaufruf eines Java-Objekts sein, also eine Nachricht an ein solches – und für eben diese Methoden sind wir selbst zuständig, sie können nicht automatisch erzeugt werden.

Eine Java-Entwicklungsumgebung ist ein Werkzeug im Sinne von Kapitel 1.1.2 zur Entwicklung von Programmen. Die Grundidee des Systems besteht darin, den Programmierern fertige Strukturen und darin lauffähige Komponentenklassen für die Standardaufgaben eines Programms bereitzustellen. Die Programmierer erzeugen – auch hier von der Entwicklungsumgebung unterstützt – Instanzen der vorgegebenen sichtbaren Komponenten (Buttons, …), deren Datenfelder sie mit konkreten Werten füllen. Das geschieht weitgehend interaktiv am Bildschirm, indem die Instanzen entweder mit der Maus bearbeitet werden, um z. B. die Größe und Position zu bestimmen, oder durch das Setzen bestimmter Eigenschaften (*Properties*) wie Farbe oder Schriftart mit Hilfe eines *Objektinspektors*. Als Besonderheit bietet Java die Möglichkeit, nicht nur Programme zu schreiben, die als *Anwendungen* (*Applications*) unter einem Betriebssystem arbeiten, sondern auch *Applets*, die in einem Browser wie z. B. dem *Internet-Explorer*, *Mozilla* oder *Opera* laufen. Die Entwicklung von Anwendungen und Applets unterscheidet sich mit Entwicklungsumgebungen kaum, weil entsprechende Vorlagen zur Verfügung stehen. Wir werden deshalb zuerst meist Applets entwickeln, um diese z. B. auf Homepages etc. zu veröffentlichen.

Die sichtbaren Komponenten agieren (meist) nicht selbst, sondern reagieren auf Ereignisse (*Events*), die z. B. durch einen Mausklick oder einen Tastendruck auf der Tastatur ausgelöst werden. Das Betriebssystem empfängt solche Ereignisse und reicht sie an die Elemente auf dem Bildschirm weiter. Ein typisches *Mausereignis* ist ein Doppelklick auf das Kreuz-Symbol in der oberen rechten Ecke eines Fensters, das z. B. eine Ereignismeldung *<Doppelklick an der Position (400,120)>* erzeugt, die dann der Reihe nach an die Fenster des *Desktops* durchgereicht wird. Stellt das angeklickte Fenster fest, dass sein Kreuzchen getroffen wurde, dann reagiert es darauf, indem es sich selbst schließt.

Die Objekte müssen somit

- der grafischen Oberfläche des Betriebssystems bekannt gemacht werden, um die Botschaften des Systems zu empfangen,
- und über Methoden verfügen, um auf die Standardereignisse des Systems zu regieren.

Beides wird erreicht, indem alle Objekte von einer Mutterklasse *Object* (bzw. deren Tochterklasse *Component*) abgeleitet werden, die über solche Eigenschaften verfügt und sie an die Tochterklassen vererbt. Die Standardreaktion auf die Standardereignisse beruht typischerweise darin, nichts zu tun.

Weil die sichtbaren Komponenten eines Programms ebenso wie die Standardereignisse festgelegt sind, kann das Entwicklungssystem fertige Programmschablonen (*Templates*) erstellen, die die vom Programmierer am Entwicklungsbildschirm zusammengestellte Oberfläche aus Fenstern, Beschriftungen, Knöpfen, Ein- und Ausgabefeldern, ... erzeugen, ohne dass eine einzige Zeile Java-Quelltext selbst geschrieben werden muss. Damit ist nur festgelegt, wie die Oberfläche aussieht, aber nicht, was sie tut. Bis auf wenige Standardreaktionen (z. B. das Fenster zu schließen) reagiert das Programm gar nicht!

Das Programmieren unter Java besteht deshalb weitgehend daraus, die anfangs leeren Ereignisbehandlungsmethoden (*Event-Handler*) auszufüllen, indem Java-Quelltext eingegeben wird, der beschreibt, wie das gerade bearbeitete Objekt auf einzelne Ereignisse zu reagieren hat:

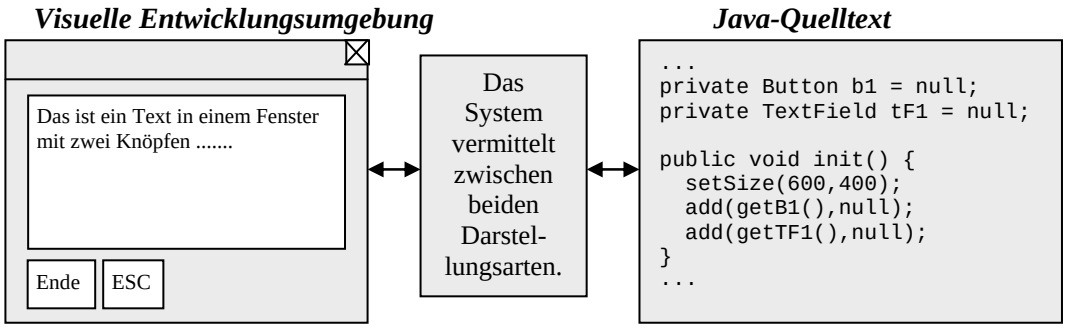
- Was passiert, wenn ein bestimmter Knopf angeklickt wird? *(Das Programm endet.)*
- Was passiert, wenn ein anderer Knopf angeklickt wird? *(Daten werden gespeichert.)*
- Was passiert, wenn eine Taste gedrückt wird? *(Je nach gedrückter Taste wird anders reagiert.)*
- ...

Das Schreiben von Javaprogrammen geschieht im Allgemeinen in drei Phasen:

1. **Zusammenstellung der Programmoberfläche in der Entwicklungsumgebung.**
   Dazu werden Komponenten erzeugt, die in einer Komponentenpalette am Bildschirm angeboten werden, z. B. Fenster, Knöpfe, Textfelder, ... Die Eigenschaften dieser Objekte werden mit Hilfe des Objektinspektors geeignet gesetzt. Die dazu gehörenden Java-Programmteile werden automatisch erzeugt.
2. **Ausfüllen der benötigten Ereignisbehandlungsmethoden mit Java-Quelltext.**
   Nach Auswahl des gewünschten Ereignisses (z. B. *mouseClicked*) erzeugt die Entwicklungsumgebung eine entsprechende Methode (sowie die zusätzlich erforderlichen Mechanismen) und springt an die richtige Stelle im Programm. Der „wohlüberlegte“ Text wird hier eingegeben.
3. **Übersetzen und Testen des erzeugten Programms**
   Der Compiler übersetzt das Programm in Bytecode und erzeugt eine von der Laufzeitumgebung ausführbare Datei, die sofort gestartet wird. Das so erzeugte Programm wird normalerweise fehlerhaft sein, zumindest aber unvollständig. Deshalb sucht man die Fehler und ergänzt die Oberfläche bzw. ändert den Programmcode.

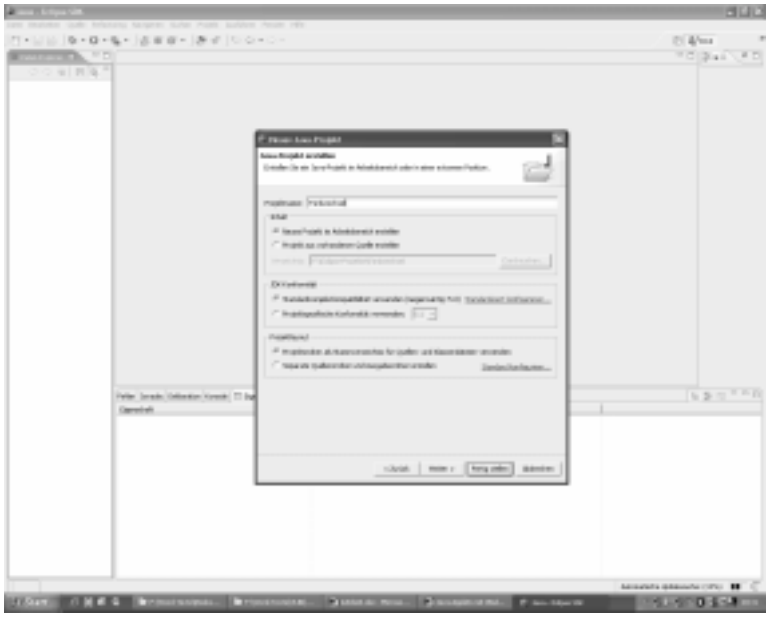
Insgesamt wird mit einem Entwicklungssystem auf zwei Ebenen gearbeitet:

- In der visuellen Entwicklungsumgebung werden Komponenten erzeugt und mit Eigenschaften ausgestattet, die die spätere Programmoberfläche bilden.
- Parallel dazu wird von der Entwicklungsumgebung automatisch Programm-Quelltext erzeugt, der zur Laufzeit genau die in der Entwicklungsumgebung definierte Oberfläche nachbildet. Dieser wird ggf. per Hand verändert oder ergänzt.



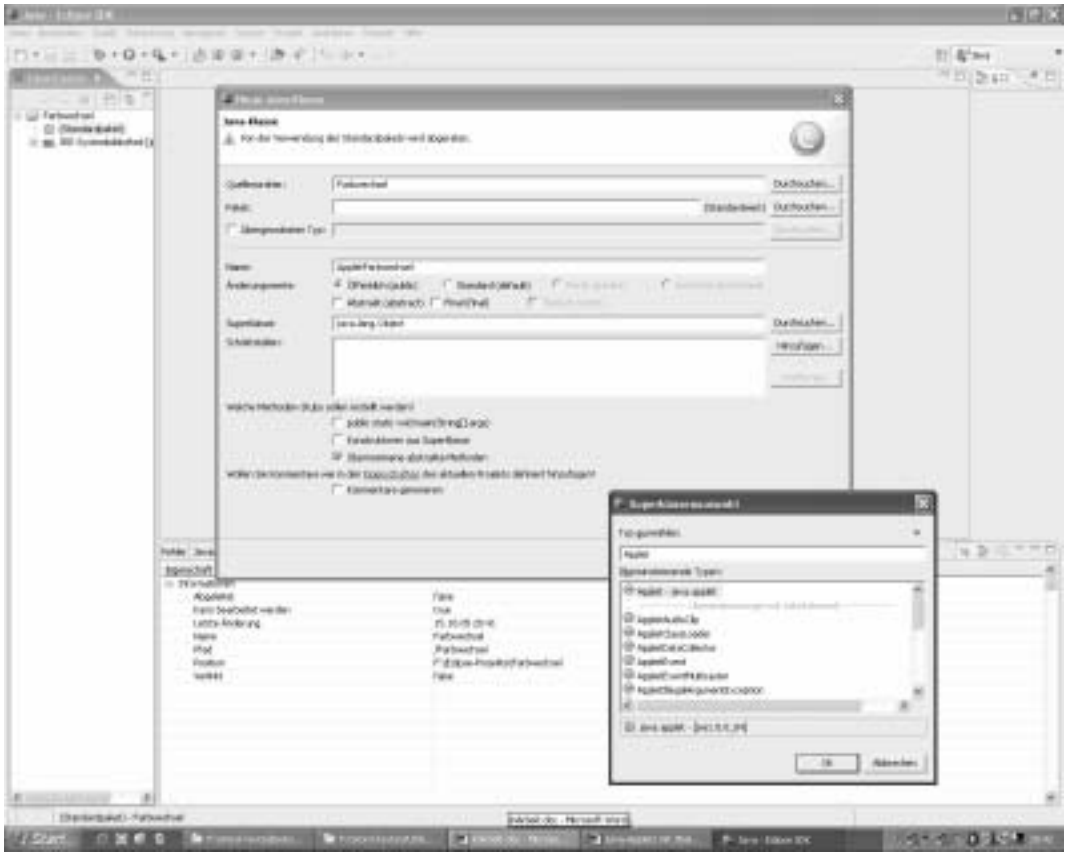
Java-Entwicklungsumgebungen arbeiten mit *Projekten*, die aus allen Bestandteilen bestehen, die erforderlich sind, um vom Compiler zu einem lauffähigen Bytecode-Programm zusammengesetzt zu werden. Dazu gehören die Quelltextdateien und andere, z. B. die unterschiedlichen Entwicklungsstufen des Projekts. Üblicherweise speichert man diese Bestandteile zusammen in einem eigenen Unterverzeichnis.

Startet man ein neues Projekt, wird meist zuerst dessen Name und Speicherort erfragt. Anschließend erhält man einen weitgehend leeren Bildschirm – denn viel mehr als ein neues Unterverzeichnis auf der Festplatte wurde noch nicht erzeugt.



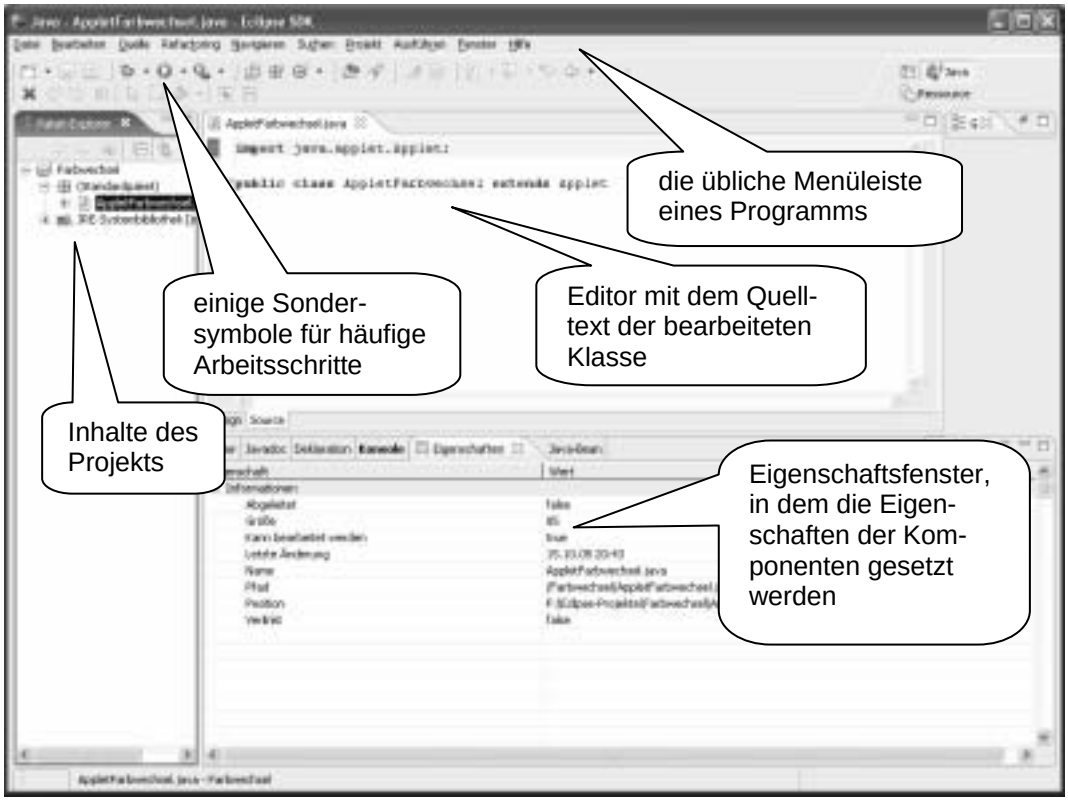
Wir wollen das weitere Vorgehen anhand zweier Entwicklungsumgebungen erläutern. Zuerst mit *Eclipse*, dessen Bildschirm „neues Projekt“ wir auf der vorigen Seite gesehen haben.

Wir fügen dem Projekt eine Javaklasse hinzu (Menü: Datei → Neu → Klasse) und legen deren Eigenschaften fest. In diesem Fall wählen wir ein Applet mit dem schönen Namen *AppletFarbwechsel*, das zum Projekt *Farbwechsel* gehören soll.



Dieses Applet speichern wir, wobei es automatisch den Suffix „*.java*“ zur Kennzeichnung einer Java-Quelltextdatei erhält, und öffnen es wieder mit dem *Visual Editor*, der dafür natürlich installiert sein muss. Diesen sollten wir aus Platzgründen so einstellen, dass der Designer und der Quelltexteditor „übereinander“ liegen, also nicht gleichzeitig angezeigt werden. (Menü: Fenster → Benutzervorgaben → Java → Visual Editor, Einstellung „*On separate notebook tabs*“ wählen.)

Danach sieht der Bildschirm schon etwas gefüllter aus.



Schon dieses Rahmenprogramm kann man erfolgreich übersetzten (grüner Pfeil oben-links), und das erzeugte leere Fenster verfügt über einige nicht ganz triviale Eigenschaften: es kann z. B. in der Größe verändert, verschoben oder geschlossen werden.

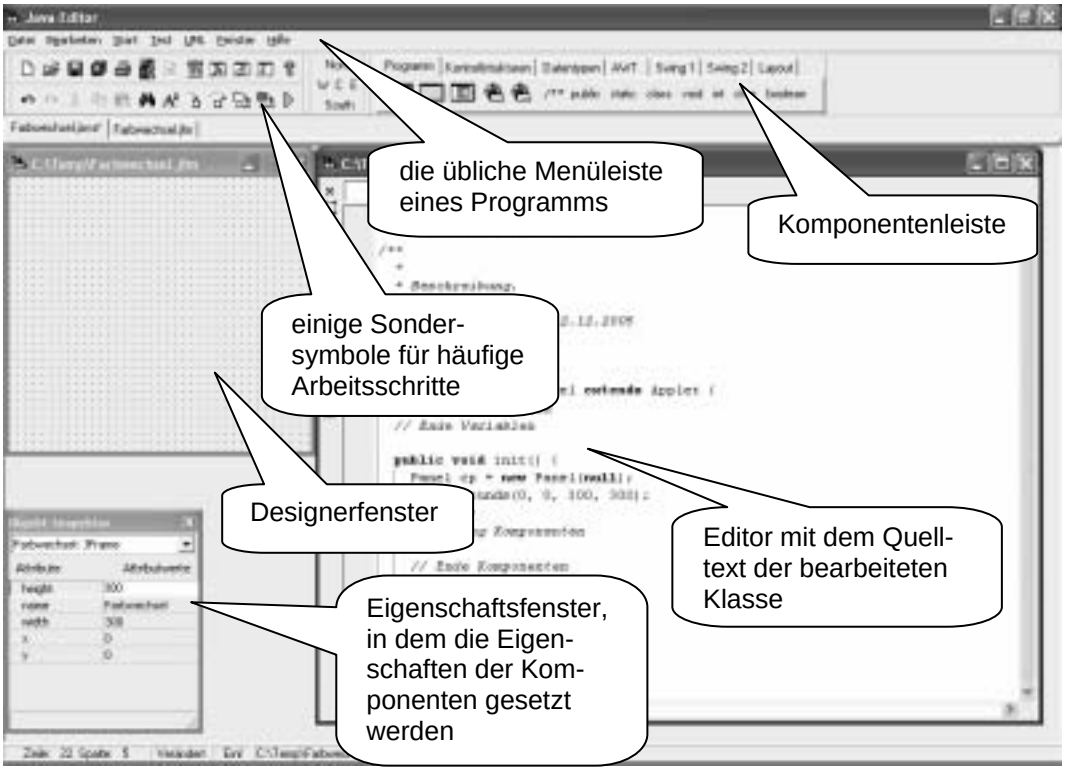
Der Bildschirm der Entwicklungsumgebung enthält zur Bearbeitung eines Projekts eine Reihe von Werkzeugen und Anzeigehilfen. Ihre Art und Anzahl hängt von der benutzten Version ab. Hier stehen Designer, Eigenschaftsfenster, Projekt-Explorer, Komponentenpalette, Quelltexteditor und Debugger sowie andere Hilfsmittel zur Verfügung. Sie können jederzeit zwischen dem visuellen Entwerfen des Objekts (*Design*), dem Bearbeiten seiner Eigenschaften und dem Erstellen der Ausführungslogik (*Source*) wechseln.

Wenn man im Objektinspektor eine quelltextbezogene Eigenschaft ändert (z. B. den Namen einer Ereignisbehandlungsroutine), wird die entsprechende Stelle im Quelltext automatisch aktualisiert. Außerdem werden Änderungen im Quelltext (z. B. Umbenennen einer Methode in einer Klassendeklaration) sofort in den Objektinspektor übernommen. Viele Möglichkeiten sind nicht nur über das Hauptmenü, sondern schneller über Kontextmenüs erreichbar, die über die rechte Maustaste gestartet werden.

Die Elemente der Komponentenpalette entsprechen Klassen, die über vorgegebene Eigenschaften und Methoden verfügen. Diese können (teilweise) im Eigenschaftsfenster angezeigt und verändert werden, aber auch zur Laufzeit des Programms.

- Die wichtigste Eigenschaft ist der *Name* der Komponente, der standardmäßig aus dem Typ und einer laufenden Nummer gebildet wird. Der erste Knopf hat damit den Namen *button1*, der zweite den Namen *button2* usw. Diese Vorgaben sollte man so ändern, dass sie der Funktion des Objekts entsprechen, z. B. in *bEingabe*.
- Beschriftete Komponenten enthalten eine Aufschrift *label*, die zuerst leer ist (bei anderen Umgebungen aber auch dem Namen der Komponenten entsprechen kann). Das Label kann verändert werden, so dass Name und Aufschrift nicht übereinstimmen müssen – z. B. wenn die Aufschrift aus mehreren Worten besteht oder Sonderzeichen enthält, die in Namen nicht erlaubt sind.
- Die Farbe der Objekte wird in der Eigenschaft *background* gespeichert. Die Werte können z. B. mit einem Farbregler eingestellt werden, der erscheint, wenn man im Eigenschaftsfenster den Wert bearbeiten will. Entsprechendes gilt für die Größe.
- Weitere Eigenschaften hängen sehr von der Funktion der Komponenten ab: Position und Größe werden durch Anordnen der Symbole mit der Maus gesetzt (und ggf. im Objektinspektor korrigiert), Schriftart und –größe (Eigenschaft *font*) werden aus einem Auswahlmenü gebildet. Details hierzu folgen in den späteren Kapiteln.
- Ereignisse, auf die eine Komponente reagieren kann, werden über das Kontextmenü (rechte Maustaste) erreicht. Details dazu folgen später. Am häufigsten wird die Reaktion auf einen Mausklick benutzt, aber auch eine Größenänderung löst Ereignisse aus.

Eine zweite, in Schulen viel verwendete Entwicklungsumgebung ist der *Java-Editor* von Gerhard Röhner, der sich auf einen für Lehrzwecke gut gewählten Funktionsumfang beschränkt und deshalb sehr viel einfacher zu handhaben ist. Startet man diesen und klickt mit der Maus auf das Applet-Symbol, dann erhält man (nach der Auswahl eines Dateinamens, hier: *Farbwechsel*) das folgende Bild:



Nach Klicken des grünen dreieckigen Startbuttons in der Menüleiste erhalten wir das gleiche leere Applet, das wir schon von Eclipse her kennen – kein Wunder, denn die Entwicklungsumgebungen benutzen das gleiche Javasystem des Rechners.

Man sieht also, dass sich die Arbeit mit diesen – und anderen – Entwicklungsumgebungen stark ähnelt. Wir werden uns folglich in diesem Buch nicht sehr stark mit deren speziellen Eigenheiten beschäftigen. Die findet man jeweils im zugehörigen Hilfesystem.