17

Verschönern Sie Ihre Java-GUI mit dynamischen Menüs und einer Statusleiste

ALEX GARRISON

Ich vermute, Sie haben sich nun durch einige der früheren Java-Tutorials in diesem Buch gearbeitet und können bereits einfache Java-Applets und Anwendungen erstellen. Nun ist es an der Zeit, sich einige der Features anzusehen, die die Anwender normalerweise von allen GUI-Programmen erwarten. Zu den häufigsten Features gehört eine Menüleiste ganz oben und eine Statusleiste ganz unten in der GUI-Anwendung. Die Menüleiste sollte den aktuellen Status der Anwendung reflektieren bzw. Aktionen, die aktuell nicht möglich sind, sollten visuelles Feedback bieten, d.h.: Sie sollten ausgeblendet oder deaktiviert werden. Statusleisten sind allgemeine Medien, die für die Kommunikation von nicht kritischen Informationen an den Anwender verwendet werden, ohne den Arbeitsfluss zu unterbrechen.

In diesem Kapitel werde ich Ihnen erklären, wie Sie Menü- und Statusleisten erstellen können, indem Sie ein kurzes Vorlagenprogramm mit der Bezeichnung **DynamicMenu** erstellen. Sie erfahren, wie sich eine Anwendung mit der AS/400 verbinden und die Verbindung wieder beenden kann. **DynamicMenu** kann problemlos erweitert werden, um neue Anwendungen zu entwickeln. Das Programm benötigt das Java Development Kit (JDK) 1.2 oder später (wenn Sie nicht JDK 1.1 und das Swing-Paket von Sun verwenden).

DYNAMICMENU

DynamicMenu, wie in Abbildung 17.1 gezeigt, ermöglicht es einem Benutzer, sich mit einer AS/400 zu verbinden und die Verbindung wieder zu beenden, indem er eine Option aus dem Menü CONNECTION auswählt. Beachten Sie, dass das Programm eine Nachricht in der Statusleiste anzeigt, sobald Sie die Option Connect ausgewählt haben.

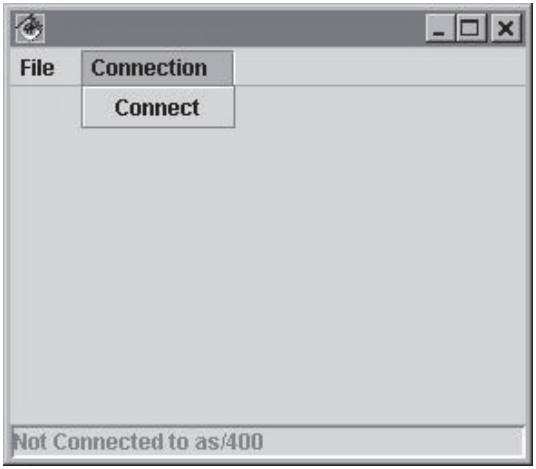

Abbildung 17.1: Die Menüs der GUI-Anwendungen sollten den aktuellen Status der Anwendung durch Aktivierung, Deaktivierung, Hinzufügen oder Entfernen von Menüoptionen anzeigen.

Wenn Sie erneut auf das Menü CONNECTION klicken, wird Ihnen auffallen, dass die Menüoption jetzt Disconnect heißt. Die Änderung der Menüoption von Connect zu Disconnect ist ein Beispiel für die Verwendung eines dynamischen Menüs, um sicherzustellen, dass alle Optionen, die einem Benutzer angezeigt werden, den aktuellen Status der Anwendung reflektieren. Abbildung 17.2 zeigt den kompletten Sourcecode für die Anwendung **DynamicMenu**.

```
package DynamicMenu;

// Demonstrate use of a dynamic menu
// and status bar

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class DynamicMenu extends JFrame
       implements ActionListener
{
```

Abbildung 17.2: Die Java-Anwendung DynamicMenu kann als Grundlage für eigene dynamische Menüs verwendet werden (Teil 1 von 3).

```
  //status bar for the bottom of the app
  private StatusBar stsbar = null;
public DynamicMenu()
{
  super();

  //allow user to exit the program by clicking
  //on the 'X' in the top right of the JFrame.
  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent evt)
    {try{System.exit(0);}catch(Throwable e){};
  }});

  //Exit menu option shows how you can use an
  //inner class to respond to the menu option.
  //Use this approach when the response is
  //simple (as in calling System.exit)
  JMenuBar mb = new JMenuBar();
  JMenu fileMenu = new JMenu("File");
  JMenuItem quitItem = new JMenuItem("Exit");
  fileMenu.add(quitItem);
  quitItem.addActionListener(
   new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {System.exit(0);
  }});
  mb.add(fileMenu);

  //the Connect menu option shows how to set the
  //application JFrame as the listener to
  //respond to the menu option.
  //This approach requires the JFrame to
  //implement ActionListener and to have an
  //actionPerformed(ActionEvent) method.
  //Use this approach when the response is
  //involved or you have many menu options
  //and want to consolidate where they
  //are handled into a single
  //actionPerformed method.
  JMenu conMenu = new JMenu("Connection");
  JMenuItem connectItem =
    new JMenuItem("Connect");
  conMenu.add(connectItem);
  connectItem.addActionListener(this);
```

Abbildung 17.2: Die Java-Anwendung DynamicMenu kann als Grundlage für eigene dynamische Menüs verwendet werden (Teil 2 von 3).

```
    mb.add(conMenu);

    setJMenuBar(mb);

    //status message at the bottom of the frame.
    //JFrame uses Border layout by default, so
    //just put the frame in the southern
    //location and let the layout manager handle
    //the status bar resizing.
    stsbar = new StatusBar();
    stsbar.setMsg("Not Connected to as/400");
    getContentPane().add(stsbar, "South");

    //manually set the size of the JFrame since
    //this application shell is basically empty
    pack();
    setSize(200,150);
  }
  public void actionPerformed(ActionEvent evt) {
    JMenuItem item = (JMenuItem)evt.getSource();
    String lbl = item.getText();

    if(lbl.equals("Connect")) {
      //!!! add code here to connect to the as/400
      //now we are connected to an as/400.  change menu option from
"Connect" to "Disconnect"
      item.setText("Disconnect");
      //set the message on the status bar
      stsbar.setMsg("Connected to as/400");
    } else if(lbl.equals("Disconnect")) {
      //!!! add code here to disconnect from the as/400
      //now we are disconnected.  change menu option from "Disconnect"
to "Connect"
      item.setText("Connect");
      //set the message on the status bar
      stsbar.setMsg("Not Connected to as/400");
    }
  }
  public static void main(String args[]) {
    DynamicMenu f = new DynamicMenu();
    f.show();
  }
  } // end DynamicMenu class
```

Abbildung 17.2: Die Java-Anwendung DynamicMenu kann als Grundlage für eigene dynamische Menüs verwendet werden (Teil 3 von 3).

Die Objekte **JMenu** und **JMenuItem** werden im Konstruktor für **DynamicMenu** erstellt. Die Erstellung eines brauchbaren Menüs besteht aus vier Schritten: Erstellung eines Objekts **Jmenu,** Hinzufügen von **JMenuItem-**Objekten zu **Jmenu**, Hinzufügen von **JMenu** zum Inhaltsfenster von **Jframe**, und – zuletzt – Hinzufügen von **ActionListener-**Objekten zu den **JMenuItem-**Objekten. Die ersten drei Schritte befassen sich mit den visuellen Aspekten des Menüs (d.h.: wie es aussieht und wo es positioniert wird). Der letzte Schritt bietet eine Methode für eine Aktion, die ausgeführt werden kann, wenn die Menüoption ausgewählt wird.

Zwei verschiedene Methoden werden normalerweise verwendet, um **ActionListener-**Objekte zuzuweisen: anonyme innere Klassen und Zuweisung zum äußeren Container. Die Option Exit im Menü FILE verwendet die Methode mit der inneren Klasse. Beachten Sie, dass der Parameter für **quitItem.addActionListener()** ein neues **ActionListener**-Objekt ist, das über nur eine Methode verfügt: **actionPerformed**. Sie müssen keinen Code in **actionPerformed** ausführen, um feststellen zu können, welche Menüoption ausgeführt wurde. Dieses innere Klassenobjekt reagiert nur auf die Exit-Menüoption. Diese Lösung ist geeignet, wenn die Reaktion auf ein Menü einfach ist (wie z.B. **System.exit(0))**.

Die zweite Methode für die Verarbeitung von Menüaktionen besteht darin, den äußeren Container als **ActionListener** hinzuzufügen. Das Objekt **connectItem** fügt **JFrame** mit folgendem Code als Listener hinzu:

```
connectItem.addActionListener(this);
```

JFrame muss nun die Schnittstelle **ActionListener** interpretieren und eine eigene **actionPerformed**-Methode bereitstellen. Diese Technik ist geeignet, wenn die Reaktionen auf eine Menüoption komplex sind oder wenn Sie viele Menüoptionen verwenden und den gesamten Verarbeitungscode in eine einzelne **actionPerformed**-Methode packen möchten. Wenn Sie z.B. mehrere Menüelemente **JFrame** als Listener verwenden, benötigt die Methode **JFrame actionPerformed** speziellen Code, um zu erkennen, welches Menüelement ausgewählt wurde.

Die Methode **actionPerformed** der Klasse **DynamicMenu** bietet ein Beispiel dafür, wie Sie erkennen können, welche Menüoption ausgewählt wurde. Die beste Methode dafür ist **getText**. Einige Programmierer verwenden **getActionCommand()**; diese Methode gibt jedoch Null zurück, wenn das Ereignis tatsächlich von einer Tastatur-Tastenkombination für das Menüobjekt ausgelöst wurde (in JDK 1.1*n*). Die Methode **getText** funktioniert richtig für beide Fälle.

Nun befinden Sie sich im Kern der dynamischen Menüelemente. Für die Änderung eines Menüelements muss lediglich die Methode **setText** von **JMenuItem** aufgerufen werden. Die Methode **actionPerformed** z.B. führt **item.setText("Disconnect")** aus, wenn der Benutzer auf das Menüelement Connect klickt. Beachten Sie hierzu, dass Sie ebenso gut die Menüoption hätten deaktivieren können, indem Sie **item.setEnabled(false)** aufrufen, aber so würden Sie keine Möglichkeit erhalten, um es dem Benutzer zu ermöglichen, die VVerbindung mit der AS/400 zu trennen.

Sobald Sie das richtige Gefühl für Änderungen an Menüobjekten bekommen haben, können Sie damit beginnen, weitere fortgeschrittene Techniken einzusetzen: z.B. das Hinzufügen neuer Menüoptionen oder das Hinzufügen oder Ändern von Untermenüs. Der Schlüssel hierzu ist, das vorrangigste Ziel nicht aus den Augen zu lassen: Der Benutzer sollte nur die Optionen auswählen lassen, die für den aktuellen Status der Anwendung geeignet sind.

Statusleiste

Im vorherigen Abschnitt haben Sie festgestellt, dass ein Mausklick auf die Menüoption Connect dazu führte, dass eine Nachricht in der Statusleiste ganz unten in der Anwendung angezeigt wird. Die Statusleiste wird mit Hilfe der Klasse **StatusBar** erstellt (siehe Abbildung 17.3), bei der es sich einfach um ein **JLabel** handelt, das in ein **JPanel** mit einem sauber gezogenen Rand eingefügt wurde. Um die Statusleiste im **DynamicMenu**-Konstruktor in **JFrame** zu positionieren (gezeigt in Abbildung 17.2), gehen Sie wie folgt vor:

```
getContentPane().add(stsbar, "South");
```

Da **JFrame** standardmäßig den **BorderLayout** Layout-Manager verwendet, wird der Layout-Manager sicherstellen, dass die Statusleiste immer ganz unten in **JFrame** positioniert wird und sich um die Größenänderung der Statusleiste kümmern, wenn sich die Maße von **JFrame** ändern.

```
package DynamicMenu;

import java.awt.*; //used by BorderLayout
import javax.swing.*;
```

Abbildung 17.3: Die Klasse StatusBar sorgt für die Darstellung des Status einer Anwendung an der südlichen Grenze des Fensterrahmens (Teil 1 von 2).

```
import javax.swing.border.*;

public class StatusBar extends JPanel
{
private JLabel stsMsg = null;
public StatusBar()
{
  super();
  setLayout(new BorderLayout());
  stsMsg = new JLabel("");
  add(stsMsg,"Center");
  stsMsg.setBorder(
    BorderFactory.createBevelBorder(
       BevelBorder.LOWERED));
}
public void setMsg(String newMsg)
{stsMsg.setText(newMsg);}
} //end StatusBar class
```

Abbildung 17.3: Die Klasse StatusBar sorgt für die Darstellung des Status einer Anwendung an der südlichen Grenze des Fensterrahmens (Teil 2 von 2).

Wenn der Benutzer z.B. auf die Menüoption Connect klickt, wird ein **ActionEvent** in die Abstract Window Toolkit- (AWT-) Ereigniswarteschlange für jeden registrierten Listener (**JFrame** in diesem Fall) eingefügt. Der AWT-Ereignis-Thread nimmt anschließend das Ereignis aus der Warteschlange und ruft die Methode **actionPerformed** des **JFrame** auf. Die Methode **actionPerformed** modifiziert anschließend die Statusleiste, indem folgender Befehl ausgeführt wird:

```
stsbar.setMsg("Connected to as/400");
```

Sie können die Statusleiste an einer beliebigen Stelle in Ihrer Anwendung festlegen, indem Sie einfach die Methode **StatusBar.setMsg(String)** aufrufen.

Die Klasse **StatusBar** ist trügerisch simpel. Durch Vererbung von **JPanel** hat die Klasse viel Funktionalität in nur wenigen Codezeilen verkapselt. In der besten Tradition der Objektorientierung und Codewiederverwendung können Sie nun die Klasse **StatusBar** in anderen Anwendungen einsetzen, die eine Statusleiste benötigen.

Menüerstellung und Status-Aspekte

Dynamische Menüs sind ein nützliches Tool im GUI-Design. Wie jedes Tool können sie jedoch falsch eingesetzt werden. Die übermäßige Verwendung von dynamischen Menüs kann die Benutzer desorientieren, indem ihr Bezugsrahmen drastisch verändert wird. Der übermäßige Einsatz kann auch die Komplexität eines ansonsten einfachen Programms dramatisch erhöhen.

Statusleisten sollten bei jeder Gelegenheit verwendet werden, um den Benutzer zu informieren und mit ihm zu interagieren. Lange Prozesse können dazu führen, dass der Benutzer glaubt, das Programm sei abgestürzt, weil er kein visuelles Feedback wie z.B. eine Statusleiste erhält, mit der der Verlauf des Programms angezeigt wird. Achten Sie darauf, dass die Klasse **StatusBar** nicht Thread-sicher ist (wie alle Swing-Klassen). Wenn die Statusleisten-Meldung von einem anderen Thread als dem AWT-Ereignis-Thread festgelegt wird, sollten Sie die Methoden **invokeAndWait** oder **invokeLater** entsprechend den Beschreibungen in der Dokumentation von Sun Microsystems einsetzen.

Die Erstellung eines nützlichen Programms erfordert in jeder Programmiersprache eine gründliche Prüfung der Benutzerschnittstellen-Aspekte – und Java bildet keine Ausnahme. Wenn Sie die einfachen Techniken verwenden, die erforderlich sind, um dynamische Menüs und eine Statusleiste darzustellen, können Sie eine Anwendung verschönern, indem Sie Features hinzufügen, die der Benutzer erwartet und die die Produktivität erhöhen.

AWT-Anwendungen mit Menüs und Beschleuniger versehen

Die Menüklassen, die mit den Java Foundation Classes (JFC) ausgeliefert werden, sind ein Fortschritt gegenüber den Menüklassen von AWT. Aber einige IT-Abteilungen sind eventuell nicht in der Lage, die JFC-Komponenten zu verwenden. Wenn Sie z.B. Java mit webbasierten Applets einsetzen, bestehen Einschränkungen aufgrund dem von den Webbrowsern Ihres Clients unterstützten JDK. Viele Browser unterstützen nur JDK 1.1, das die JFC-Komponenten nicht bietet. Sie können dem aus dem Wege gehen, indem Sie die Datei **Swingall.jar**, die JFC für JDK 1.1 bietet, entweder als Applet-Archiv oder über das JDK 1.2 Plugin von Sun bereitstellen; diese Optionen verlängern jedoch entweder die Applet-Startzeit oder sie setzen voraus, dass Ihre Kunden ein Plugin akzeptieren.

Auf jeden Fall ist es kein Problem, wenn Sie nur JDK 1.1 verwenden können, weil das Hinzufügen von Menüs und Beschleunigungstasten für Menüoptionen zu Ihren AWT-Anwendungen ganz einfach ist. In dem in Abbildung 17.4 gezeigten Beispiel ruft z.B. die Tastenfolge „Ctrl-A“ die Menüoption Attach auf. Die Javaklasse **AWTMenu** in Abbildung 17.5 zeigt z.B., wie einfach das Hinzufügen von Menüs und Beschleunigungstasten sein kann, wenn Sie die Java AWT-Klassen verwenden, die mit Ihrem JDK 1.1 oder mit Ihrer bevorzugten integrierten Java-Entwicklungsumgebung ausgeliefert wird.

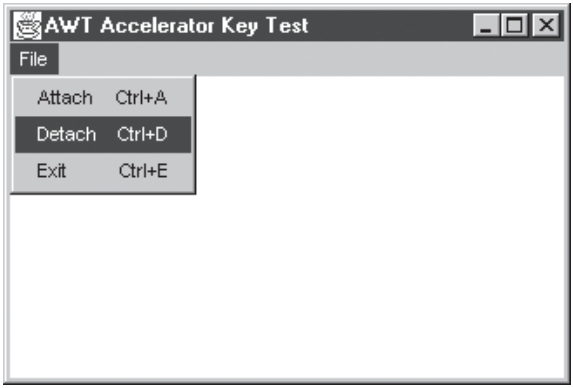

Abbildung 17.4: Beschleunigertasten sparen Zeit bei der Auswahl von Objekten aus Menüs.

```
import java.awt.*;
import java.awt.event.*;
public class AWTMenu implements ActionListener {
  public static void main(String args[]) {

    // bootstrap myself
    AWTMenu awtMenu = new AWTMenu();

    // create application Frame and Menu
    Frame awtFrame =
       new Frame("AWT Accelerator Key Test");
    Menu menu = new Menu("File");

    // Attach menu option
    MenuShortcut hotAttach =
      new MenuShortcut(KeyEvent.VK_A);
    MenuItem menuItem1 =
      new MenuItem("Attach", hotAttach);
    menuItem1.addActionListener(awtMenu);
    menu.add(menuItem1);
```

Abbildung 17.5: Die Java-Source für AWTMenu zeigt, wie einfach es ist, Menüoptionen mit Beschleunigertasten zu ergänzen (Teil 1 von 2).

```
    // Detach menu option
    MenuShortcut hotDetach =
      new MenuShortcut(KeyEvent.VK_D);
    MenuItem menuItem2 =
      new MenuItem("Detach", hotDetach);
    menuItem2.addActionListener(awtMenu);
    menu.add(menuItem2);

    // Exit menu option
    MenuShortcut hotExit =
      new MenuShortcut(KeyEvent.VK_E);
    MenuItem menuItem3 =
      new MenuItem("Exit", hotExit);
    menuItem3.addActionListener(awtMenu);
    menu.add(menuItem3);

    // create menu bar
    MenuBar menuBar = new MenuBar();
    menuBar.add(menu);
    awtFrame.setMenuBar(menuBar);
    awtFrame.setSize(200,200);
    awtFrame.setVisible(true);

    return;
  }
  public void actionPerformed(ActionEvent e) {
    String label = ((MenuItem)e.getSource()).
        getLabel();
    if (label.equals("Attach"))
       System.out.println(
          "Connect to the AS/400 code here");
    else if (label.equals("Detach"))
       System.out.println(
          "Disconnect to the AS/400 code here");
    else if (label.equals("Exit"))
       System.exit(0);
  }
}
```

Abbildung 17.5: Die Java-Source für AWTMenu zeigt, wie einfach es ist, Menüoptionen mit Beschleunigertasten zu ergänzen (Teil 2 von 2).

Die Funktion **main** von **AWTMenu** erstellt als Erstes eine Objektinstanz mit seinem eigenen Typ „Klasse“ mit der Bezeichnung **awtMenu**. Die Funktion **main** erstellt anschließend den **Frame** und ein **Menu** namens FILE. Dann wird **MenuShortcut** erstellt, das als Beschleuniger für die Menüoption Attach dienen wird. Der Parameter für den Konstruktor **MenuShortcut** ist **KeyEvent.VK_A** (**VK_A** ist eine Konstante, die bedeutet, dass die virtuelle Taste der Buchstabe *A* ist).

Als Nächstes erstellt die Funktion **main** für die Option Attach ein **MenuItem**. Der Konstruktor **MenuItem** kann entweder das Menü-Label als einzelnen Parameter erhalten oder ein Menü-Label und einen **MenuShortcut** als mehrere Parameter. Es wurde ein Menüobjekt erstellt, indem der Konstruktur **MenuItem** das Label „Attach“ erhielt und einen **MenuShortcut** mit der Bezeichnung **hotAttach** (wobei es sich um die Beschleunigertaste „Ctrl-A“ handelt). An das **MenuItem** wird anschließend ein Listener angehängt, der die Benutzerauswahl dieser Menüoption verarbeitet (ich werde diesen Punkt im letzten Absatz genauer erklären). Die Funktion **main** fügt anschließend das neu erstellte Menüobjekt dem Menü hinzu.

Die Funktion **main** der Klasse **AWTMenu** verwendet den gleichen Prozess, um die Menüoptionen Detach und Exit zu ergänzen, die beide zugeordnete Beschleuniger entsprechend ihrem letzten Buchstaben sind. **main** erstellt zuletzt eine **MenuBar** für die Anzeige ganz oben im Programm-Fensterrahmen. Die **MenuBar** wird das Menü aufnehmen, das Menü der Menüleiste hinzufügen, das Menü in den Rahmen der Anwendung einfügen und dann die Größe des Rahmens bestimmen.

Die Ereignisverarbeitung für die Auswahl einer Menüoption erfolgt in der Funktion **actionPerformed**. Nachdem die einzelnen Menüoptionen erstellt wurden, wurde das Objekt **awtMenu** (die Anwendungsklasse selbst) diesem Menüelement als ActionListener mit einem Aufruf der Funktion **addActionListener** hinzufügt. Wenn ein Benutzer eine Menüoption aktiviert, wird automatisch die Funktion **actionPerformed** des Objekts aufgerufen, das in der Funktion c**addActionListener** angegeben wird. In diesem Fall handelt es sich bei der Klasse, die die Funktion **actionPerformed** enthält, um die gleiche Klasse, die auch die Objekte **Frame**, **Menu** und alle **MenuItem-**Objekte enthält. Auf jeden Fall verwendet die Funktion **actionPerformed** beim Aufruf die Funktion **getSource**, um das Label des Menüs abzurufen, das vom Benutzer ausgeführt wurde, und führt dann den entsprechenden Code aus.