[Die Benutzeroberfläche erstellen – „Geldkarte“]

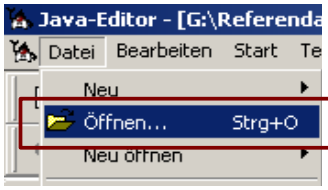
Eine Testoberfläche „Mockup“ simuliert die Vorgänge *„Betrag aufladen“, „Betrag bezahlen“* und *„Guthaben abfragen“* mit einer Geldkarte.

Schritt 1: Die *Entwicklungsumgebung*, den *Java-Editor* öffnen.

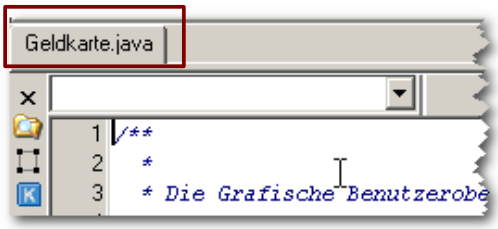
Kopieren Sie das bestehende Verzeichnis Geldkarte02. Fügen Sie das Verzeichnis ein und benennen es in Geldkarte03 um! Das ist ihr heutiges Arbeitsverzeichnis...

Öffnen Sie ihre Entwicklungsumgebung, den Java-Editor: *Start >> Programme >> Java-Editor*

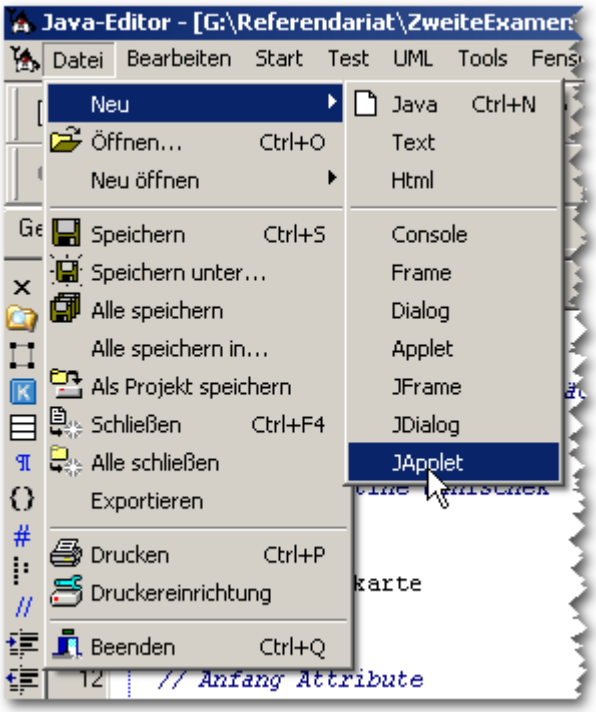
Schritt 2: die *Fachklasse* „Geldkarte.java“ öffnen.



Öffnen Sie die Datei `Geldkarte.java` aus dem gerade erstellten (kopierten) Ordner Geldkarte03 über die *Menü-Leiste >> Datei >> Öffnen.*



Schritt 3[1]: *Hauptfenster*-Klasse (grafische Benutzeroberfläche, GUI) erstellen.



Die Vorgänge nennen sich *„Betrag aufladen“, „Betrag bezahlen“ und „Guthaben abfragen“*.

Dazu benötigen wir die GUI. Der Benutzer des Geldkarten-Systems kann einen *Betrag angeben* und anschließend mit einem entsprechenden Klick auf die entsprechende Schaltfläche den Vorgang auslösen.

Heute erstellen wir diese Benutzerfläche, nicht das auslösen der Vorgänge! (Kommt später...)

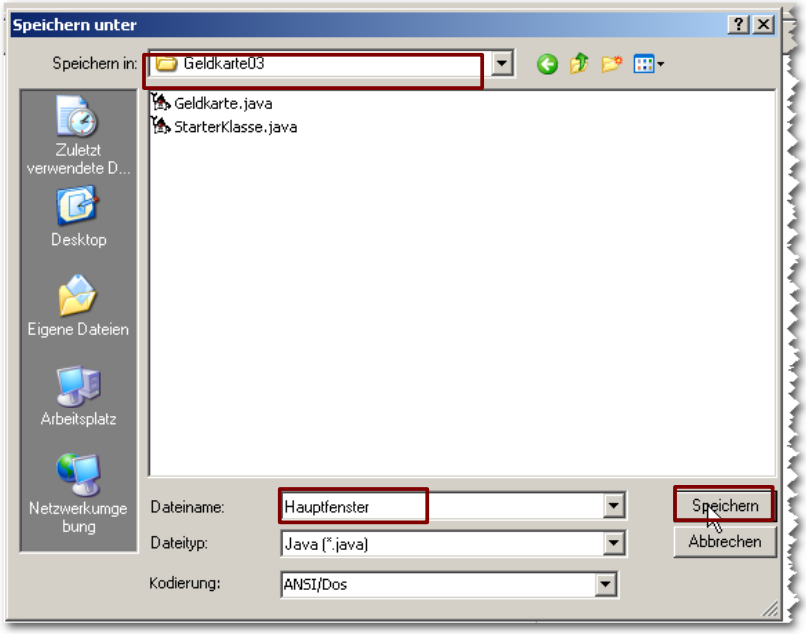
Für die Web-Anwendungen wählen wir die Klasse *JApplet*.

Wählen Sie dazu in der Menü-Leiste die Option: *Datei >> Neu >> JApplet*

Erklärung:
Ein *JApplet* ist das Gerüst für das *Hauptfenster (die Benutzeroberfläche, GUI)* einer *Web-Anwendung.* Für Anwendungen anderer Art, wie Beispielsweise Software oder Mobile Anwendungen nutzen wir andere Gerüste *(JFrame oder eine XML-Datei)* für die GUI.

1 *JApplet* ist eine Klasse der Swing-API (Applikation Programming Interface), eine der mächtigsten Java-Bibliotheken für grafische Benutzeroberflächen.

Schritt 4: Eine *Hauptfenster*-Klasse (grafische Benutzeroberfläche, GUI) speichern.



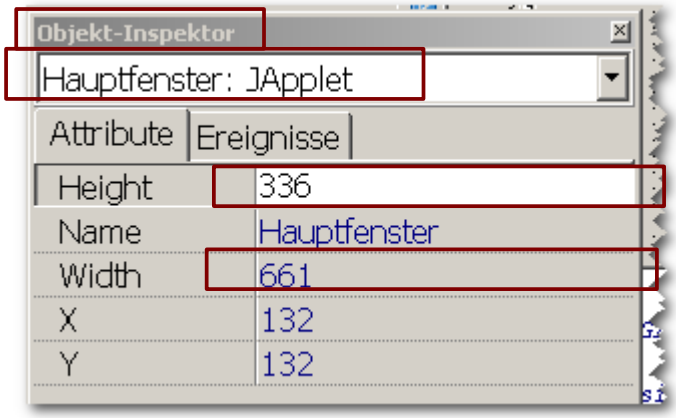
Nennen Sie die neue Klasse für die Benutzeroberfläche `Hauptfenster.java` und bestätigen Sie Ihre Eingabe mit einem Klick auf *Speichern*.

Mit dem Klick auf *Speichern* erstellt der Java-Editor **zwei** Dateien.

Eine mit der Endung *.jfm* und eine mit der Endung *.java.*

Erläuterung:
Die Datei `Hauptfenster.jfm` ist die grafische Benutzeroberfläche des Java-Editor zum erstellen von grafischen Benutzeroberflächen, sie kann ausschließlich mit dem Java-Editor geöffnet werden. Die generierte Datei `Hauptfenser.java` enthält den dafür übersetzten (generierten) Java-Quellcode (kann von jeder Entwicklungsumgebung geöffnet werden). Da Wir im Java-Editor entwickeln sollten wir Veränderungen der grafischen Benutzeroberfläche stets in der *.jfm-Datei* durchführen, der Quellcode-Generator des Java-Editors passt den Quellcode der *.java-Datei* dann automatisch an.

Schritt 5: *Hauptfenster*-Größe einstellen.

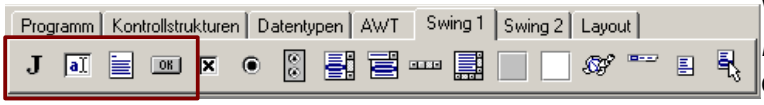


Im Vordergrund erscheint ein Fenster, der Objekt-Inspektor. Das Fenster zeigt im Moment die Maße (Höhe und Breite) und die Position (X und Y) des Hauptfensters an.

Verändern Sie die Höhe (*Height*) auf *336* (Pixel) und die Breite (*Width*) auf *661* (Pixel), wie im Bild angezeigt. ***Achtung:*** Bestätigen Sie Veränderungen immer mit der *Enter-Taste* auf Ihrer Tastatur, damit die Veränderungen in Java-Quellcode übersetzt werden können.

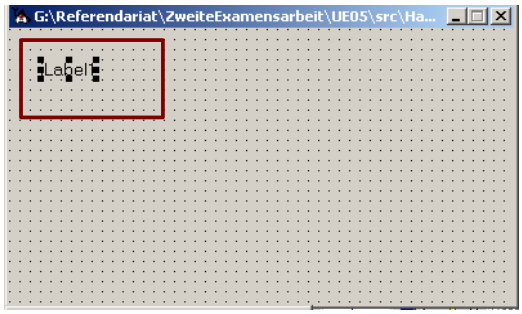
Hinweis: Falls der *Objekt-Inspektor* nicht sichtbar ist, können Sie ihn jederzeit über die Menü-Leiste: *Fenster >> Objekt-Inspektor ein/aus* anzeigen.

Schritt 6: Eine Auswahl wichtiger *Komponenten* einfügen.



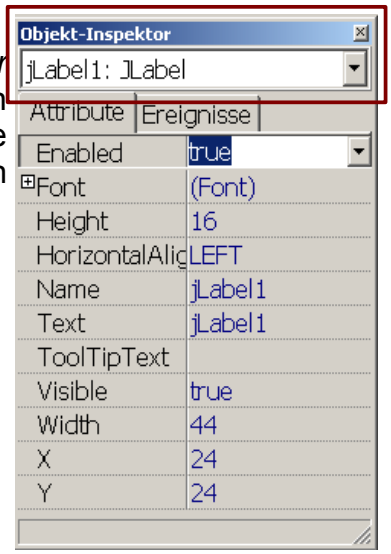
Wählen Sie in der *Symbol-Leiste* des *Java-Editors* den Reiter *Swing 1* aus. Hier sind einzelne *Komponenten* aufgeführt. Es gibt u.a. Bezeichnungsfelder (*JLabel*), Texteingabefelder (*JTextfield*), Textbereichsfelder (*JTextarea*) und Schaltflächen (*JButton*).

Schritt 7: *Komponenten* einfügen.

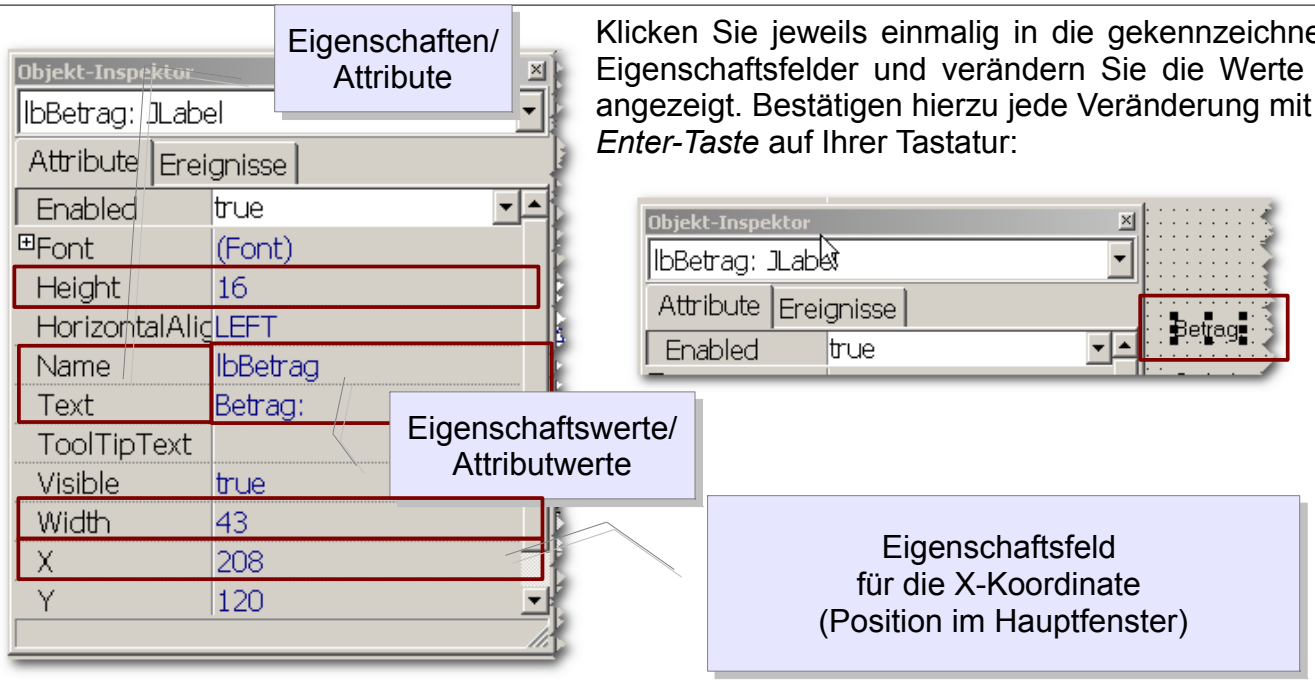


Fügen Sie nun Ihre erste Komponente ein. Klicken Sie hierzu in der *Symbol-Leiste* das Symbol *JLabel* und ziehen Sie mit gedrückter linker Maustaste ein Bezeichnungsfeld in der Datei `Hauptfenster.jfm` auf. Es erscheint ein Feld *Label1.*

Das *Objekt-Inspektor-Fenster* schiebt sich gleichzeitig in den Vordergrund und zeigt die Eigenschaften des neuen Bezeichnungsfeldes (*Label1*) an:

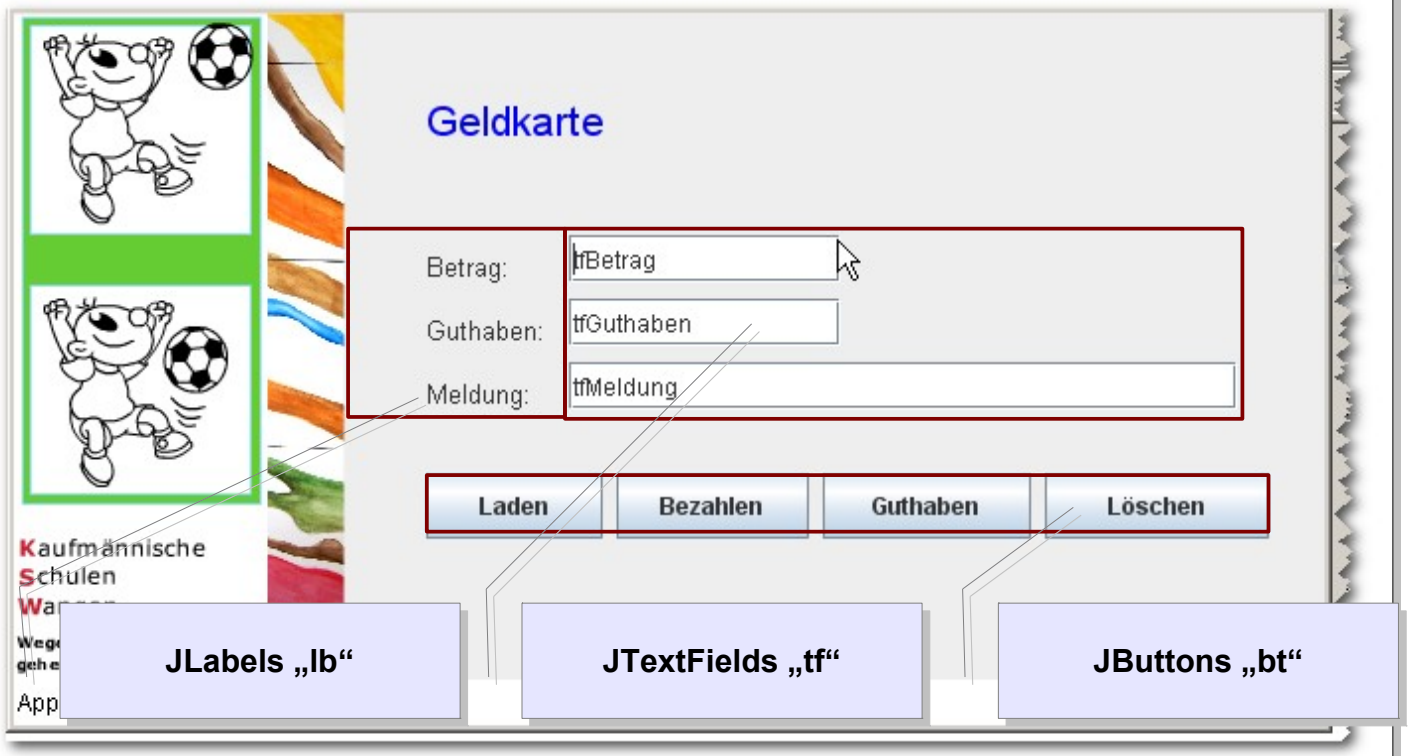


Schritt 8: *Komponenten*-Eigenschaften verändern.



Klicken Sie jeweils einmalig in die gekennzeichneten Eigenschaftsfelder und verändern Sie die Werte wie angezeigt. Bestätigen hierzu jede Veränderung mit der *Enter-Taste* auf Ihrer Tastatur:

Schritt 9: *Komponenten* selbstständig im Hauptfenster platzieren.

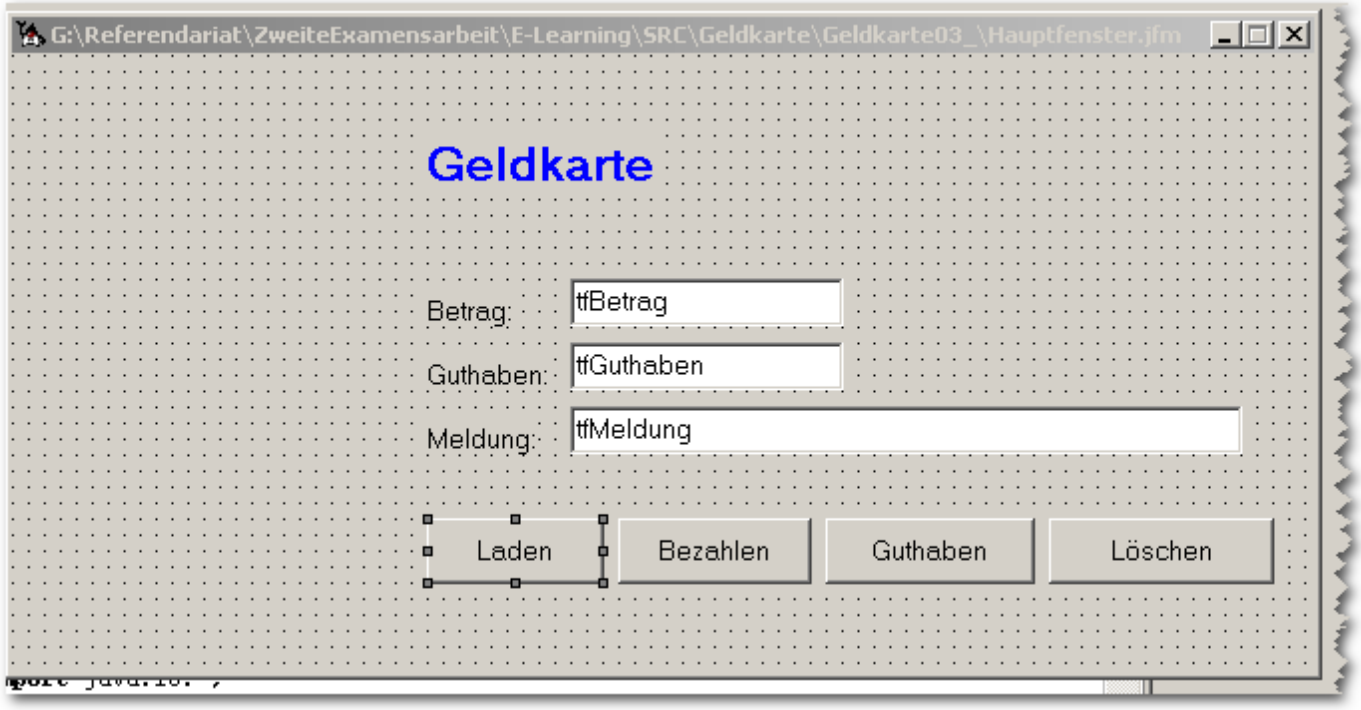


Ergänzen Sie, auf die gleiche Weise, wie in *Schritt 6 bis 8* beschrieben, die noch fehlenden *Komponenten* der grafischen Benutzeroberfläche.

Zu Schritt 9: *Komponenten* selbstständig im Hauptfenster platzieren.

Verändern Sie dabei die Eigenschaften der einzelnen *Komponenten*. Die Einstellungsmöglichkeiten für die *Komponenten* finden Sie jeweils im *Objekt-Inspektor-Fenster*. Die Veränderungen der *Eigenschaften (Name, Typ, Text, Width, Height, X, Y)* sollten wie folgt erfolgen:

Name	*Typ*	*Text*	*Width*	*Height*	*X*	*Y*
lbBetrag	JLabel	Betrag:	43	16	208	120
lbGuthaben	JLabel	Guthaben:	61	16	208	152
lbMeldung	JLabel	Meldung:	55	15	208	184
tfBetrag	JTextField	tfBetrag	137	24	280	112
tfGuthaben	JTextField	tfGuthaben	137	24	280	144
tfMeldung	JTextField	tfMeldung	337	24	280	176
btLaden	JButton	Laden	89	33	208	232
btBezahlen	JButton	Bezahlen	97	33	304	232
btGuthaben	JButton	Guthaben	105	33	408	232
btLoeschen	JButton	Löschen	113	33	520	232



[Ergebnis: Hauptfenster.jfm – Hauptfenster.java]

Schritt 10: *Methode* der GUI-Aktion „Inhalte der Textfelder leeren“.

Suchen Sie in der Hauptfenster-Klasse (Hauptfenster.java) die Methode:

```
public void btLoeschen_ActionPerformed(ActionEvent evt){
          \\ TODO hier Quelltext einfügen
     }
```

```
136  public void btLoeschen_ActionPerformed(ActionEvent evt) {
137      tfBetrag.setText("");
138      tfGuthaben.setText("");
139      tfMeldung.setText("");
140  }
```

und fügen Sie die fehlenden Quelltextelemente hinzu.

Ergänzung:
Mit den Aufrufen `<Textfeldname>.setText("");` löschen wir den aktuellen Inhalt des Textfeldes (Textfield). Im Ergebnis werden hier also alle aktuellen Inhalte der vorhandenen Textfelder nacheinander mit „nichts“ überschrieben.

Schritt 11: Testen Sie ihr erstes Zwischenergebnis.

Klicken Sie einmalig in der Symbol-Leiste des Java-Editors auf das grüne Pfeil-Symbol, um die Datei zu kompilieren und auszuführen:

Erläuterung:
Beim kompilieren übersetzt der Compiler (ist Bestandteil des Java Developement Kits, JDK) den Inhalt einer *.java-Datei* in eine *.class-Datei,* diese Datei wird dann ausgeführt und das Ergebnis wir angezeigt. In unserem Fall wird die gerade erstellte Benutzeroberfläche angezeigt.

Testen Sie die Schaltfläche *„Löschen“.* Werden die Inhalte der Textfelder, geleert? *Wenn ja, herzlichen Glückwunsch!!*

GRAFIKEN AUF DER BENUTZEROBERFLÄCHE EINES JAPPLETS

Schritt 12: Kopieren der *Grafik*-Datei in ihr Arbeitsverzeichnis.

Wir möchten links in der *GUI* die *Grafik* angezeigt bekommen.

Dazu sind die kommenden 5 Schritte notwendig!

Kopieren Sie die Datei `background.png` (*Janischek/Arbeitsmaterial/Geldkarte03*) in Ihr Verzeichnis `Geldkarte03`.

Schritt 13: *Importieren* der benötigten Grafik-Bibliotheken.

Ganz oben in der Klasse Hauptfenster.java finden Sie eine Liste bereits importierter Bibliotheken (z.B. import javax.swing.*;). Ergänzen Sie direkt unter den aufgeführten *imports* folgende Zeilen:

```
//image einbetten
import javax.imageio.*;
import java.net.*;
import java.io.*;
```

Schritt 14: *Deklaration* der für die Grafik notwendigen Attribute.

Auch in Ihrer Hauptfenster-Klasse (`Hauptfenser.java`) finden Sie die Liste der bereits definierten Attribute gleich am Anfang der Klasse ganz oben unter: >> `//Anfang Attribute`. Fügen Sie folgende zusätzlichen Attribute ein:

```
private Image image;
BackgroundPanel bg = new BackgroundPanel();
```

Hinweis: Eine Erklärung folgt *weiter unten*.

Schritt 15: Die init()-Methode eines JApplet

Suchen Sie in der Datei Hauptfenster.java nach der *init()-Methode*. Ergänzen Sie im Anschluss an den Methoden-Aufruf:

```
cp.setBounds(0, 0, 661, 336);
```

folgenden Quellcode:

```
41    //Hintergrundgrafik laden
42    try {
43      image = ImageIO.read(getClass().getResource("background.png"));
44    }
45    catch(IllegalArgumentException iae) {
46      JOptionPane.showMessageDialog(this, "Grafikdatei nicht gefunden!");
47    }
48    catch(IOException ioe) {
49      JOptionPane.showMessageDialog(this, "Fehler beim Einlesen der Grafikdatei!");
50    }
51    bg.setSize(167,336);
52    bg.setVisible(true);
53    cp.add(bg);
```

Dateiname der Grafik

Größe des Bildes
`setSize(Width, Height)`

Hinweis: Die *try-catch-Anweisung* (eine Kontrollstruktur) übernimmt die Fehlerbehandlung, falls beim Laden der Grafik etwas schief läuft. Eine Erklärung zur `init()`-Methode folgt ebenfalls in *Schritt 20*.

Schritt 16: Einbettung der BackgroundPanel-Klasse

Suchen Sie in der Datei Hauptfenster.java nach dem Kommentar :

```
// Ende Methoden
```

Kopieren Sie den Inhalt der Datei BackgroundPanel.java unter den Kommentar:

```
//Eingebettete Klasse fuer das Panel der Hindtergrundgrafik-Komponente
public class BackgroundPanel extends JPanel
{
      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         if(image != null) {
            g.drawImage(image, 0, 0, this);
         }
      }
}
```

Mit dieser Klasse Erstellen eine noch nicht vorhandene Komponente. Diese (bg) enthält unsere Grafik

Image enthält die Grafik
`background.png`

Speichern Sie alle Veränderungen und wiederholen Sie anschließend *Schritt 11*. Öffnen Sie die automatisch erzeugte Datei `Hauptfenster.html` mit Ihrem Browser *(Firefox, IE)*

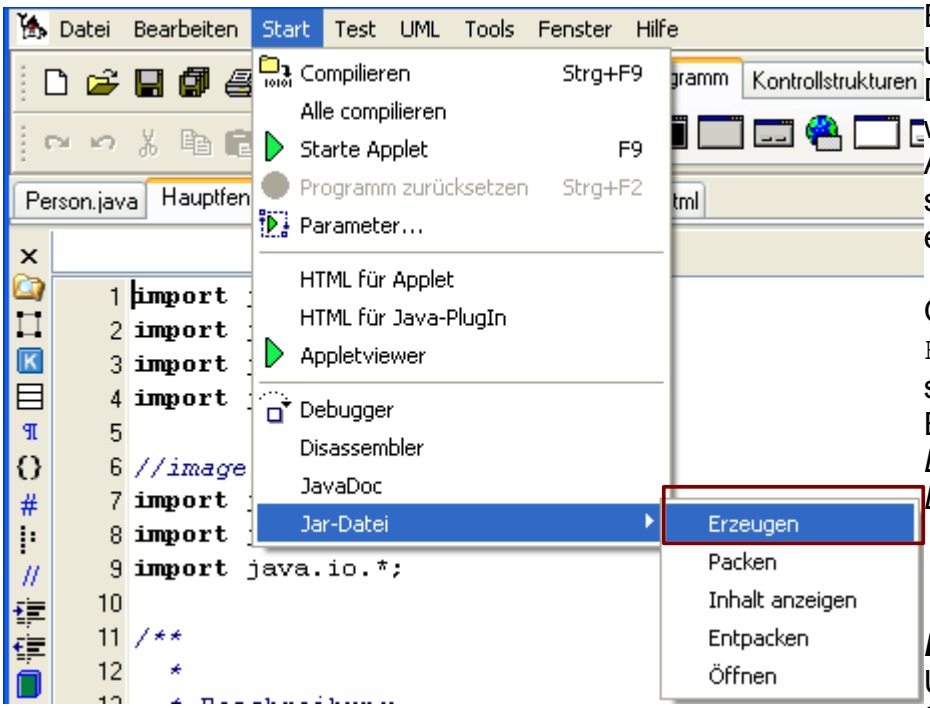
Schritt 17: Ergebnis

[Ergebnis: erzeugte grafische Benutzeroberfläche]

Im Ergebnis sollte die Grafik angezeigt werden. Falls Fehler angezeigt werden, sind diese häufig auf fehlende oder überflüssige Klammern zurückzuführen. Überprüfen Sie gegebenenfalls die Klammersetzung in der Datei „`Hauptfenster.java`“.

JAVA UND DAS WEB

Schritt 18: Die jar-Datei erstellen.



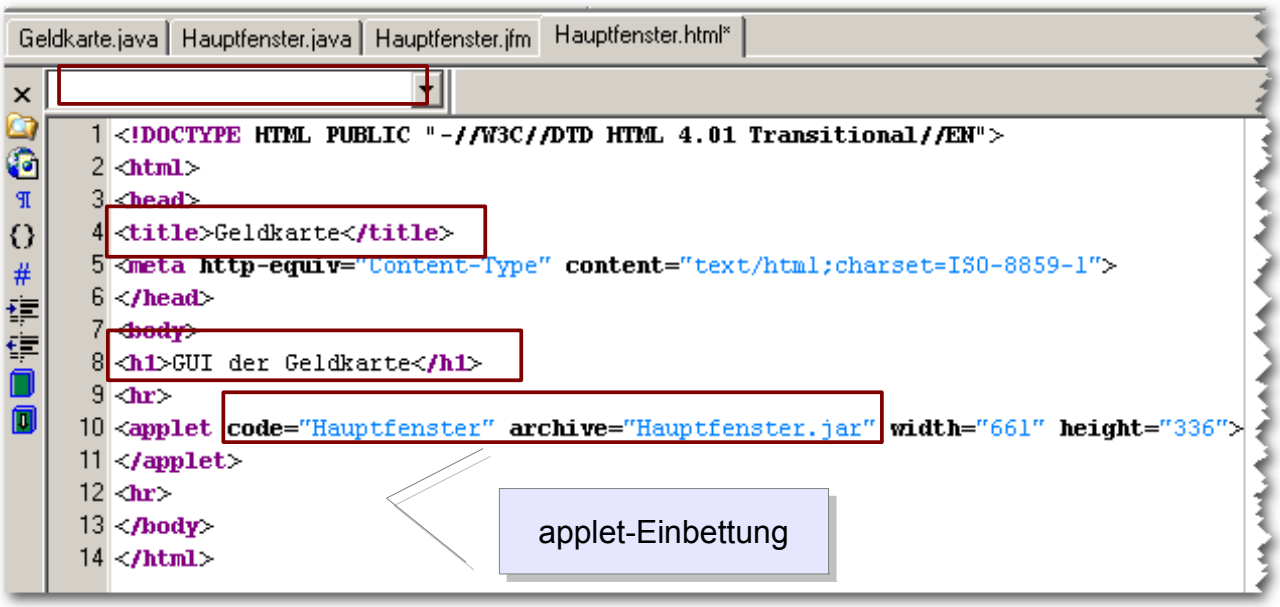
Eine *Jar-Datei* soll alle Dateien unseres Projektes enthalten inkl. Die erzeugten *.class-Dateien*. Da wir im letzten Schritt die Anwendung getestet haben, sollten alle notwendigen Dateien enthalten sein.

Öffnen Sie die Datei `Hauptfenser.java` und nutzen sie den *Jar-Generator* des Java-Editors, indem Sie in der *Menü-Leiste* die Option: *Start >> Jar-Datei >> Erzeugen*

Erläuterung:
Um den Java-Konventionen zu folgen werden wir für Projekte die wir im Internet veröffentlichen eine *.jar-Datei* erzeugen und diese anstelle der *.class-Datei* im HTML-Dokument einbinden. *Schritt 11* mit dem Ergebnis aus *Schritt 17* ist dazu zwingend notwendig.

Schritt 19: Die HTML-Datei modifizieren (verändern)

Öffnen Sie die erzeugte Datei `Hauptfenster.html` und führen Sie die folgenden Veränderungen im HTML-Quellcode durch:



Schritt 20: Erläuterung

Es ist ganz normal, dass Sie in den vergangenen Schritten nicht alles verstanden haben!

Stellen Sie sich vor Ihre Benutzeroberfläche (Hauptfenster.java) sei eine Pinwand (content pane, cp). Auf dieser Pinwand (content pane, cp) werden die einzelnen *Komponenten* u.a. Bezeichnungsfelder (*JLabel*), Texteingabefelder (*JTextfield*), Textbereichsfelder (*JTextarea*) und Schaltflächen (*JButton*) angehängt.

Erklärung anhand der Methoden:

```
82    btLaden.setBounds(208, 232, 89, 33);
83    btLaden.setText("Laden");
84    btLaden.addActionListener(new ActionListener() {
85      public void actionPerformed(ActionEvent evt) {
86        btLaden_ActionPerformed(evt);
87      }
88    });
89    cp.add(btLaden);
```

`<objektname>.setBounds(X, Y, Width, Height)`: Die Methode legt die Platzierung für die Komponente auf der Pinwand (content pane, cp).

`<objektname>.setText("Aussagekräftiger Text")`: Die Methode legt den Wert des Attributs *Text* (siehe Tabelle) der Komponente fest. Der Anwender der Benutzeroberfläche sieht diesen Text als Bezeichnung für die Komponente.

JButton-Komponenten haben im Gegensatz zu den anderen Komponenten einen *ActionListener*:

`btAnmelden.addActionListener(...);`

Der *ActionListener* ist eine Art Fühler. Der Fühler bemerkt als Erster, wenn der Benutzer die Schaltfläche btAnmelden anklickt und erzeugt im gleichen Moment ein neues Ereignis, das er an die Ereignis-Methode **`actionPerfomed(...){...}`** übermittelt. In dieser Methode bestimmen wir was passieren soll. Dafür finden wir weiter unten im Quellcode nach dem Kommentar //Anfang Methoden den zugehörigen Methoden-Rumpf für das Ereignis Anmelden:.

```
118  public void btLaden_ActionPerformed(ActionEvent evt) {
119        //TODO hier Quelltext einfuegen
120  }
```

`cp.add(<komponentenname>)`: Die Methode „hängt“ die Komponente an die Pinwand (content pane, cp).

zu Schritt 20: Erläuterung

Die init()-Methode als Konstruktor eines *JApplets* der Hauptfenster-Klasse.

init()-Methode

An die Stelle der Frame Initialisierung (bei JFrame) tritt bei einem JApplet die Methode init().

Diese Methode enthält ein ContentPane-Container (`cp`). Er dient dazu die einzelnen Komponenten der grafischen Benutzeroberfläche zu einem Fenster zusammen zu fassen.

Wozu dient die init()-Methode?

javax.swing

Class JApplet

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ java.awt.Panel
        └ java.applet.Applet
          └ **javax.swing.JApplet**

Ist eine Methode der Klasse Applet und diese gibt die Methode nach unten weiter (vererbt). Das hat den Vorteil, dass JApplet theoretisch alle Methoden von Object, Component, Container, Panel und Applet verwenden kann. Prima oder 😎?

Applets werden nicht dem Java-Interpreter übergeben, sondern in eine HTML-Datei eingebettet, die in javafähigen Browsern (Firefox, IE) geladen und angezeigt wird. Der Browser übernimmt die Kontrolle über das Applet. Applets können außerdem mit dem AppletViewer des JDK angezeigt werden. Dazu muss es ebenfalls in eine HTML-Seite eingebettet sein. (Das machen wir gleich, freu 😊).

Wozu dient der Java-Interpreter

Der Interpreter java dient dazu, kompilierte Java-Programme auszuführen, die als Bytecode in .class-Dateien vorliegen. Bei Internetbrowsern, wie dem Firefox oder Internet Explorer kann man eine Erweiterung (nennt man auch Java VM oder virtuelle Maschine) installieren. Sie enthält den Java-Interpreter.

Merke:

Applets brauchen keinen Konstruktor. Ein Default-Konstruktor würde so aussehen:

```
public <Klassenname>() {…}
```