

[Die Benutzeroberfläche erstellen – „Profil anlegen“ und „Anmelden“]

Eine Testoberfläche „Mockup“ simuliert die Prozesse „*Profil anlegen*“ und „*Anmelden*“ als Benutzer des Produkt-Pflege-Systems.

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



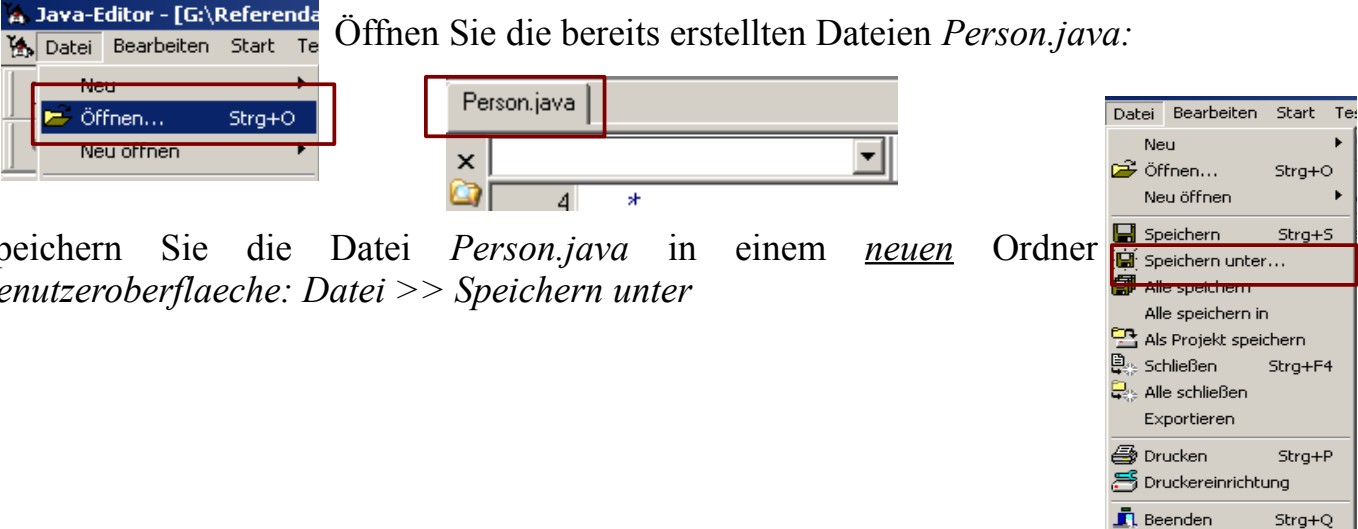
**Java-Editor**

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



## Schritt 2: die *Fachklasse* „*Person.java*“ öffnen.

Öffnen Sie die bereits erstellten Dateien *Person.java*:

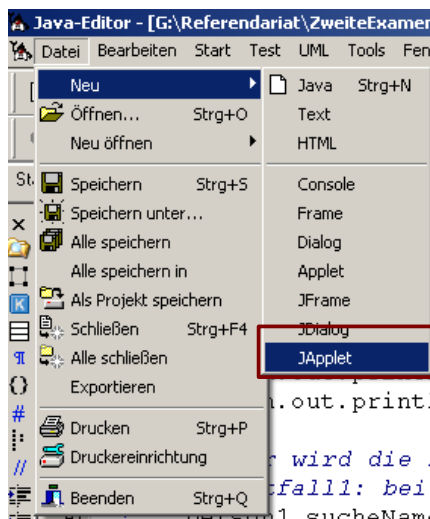


Speichern Sie die Datei *Person.java* in einem neuen Ordner *Benutzeroberflaeche*: *Datei* >> *Speichern unter*

## Schritt 3<sup>1</sup>: *Hauptfenster*-Klasse (grafische Benutzeroberfläche, GUI) erstellen.

Der Vorgang nennt sich auch „*Profil anlegen*“ und „*anmelden*“. Dazu erstellen wir im nächsten Schritt eine Benutzeroberfläche auf der sich der Benutzer des Produkt-Pflege-Systems mit seiner *PersonID*, *Namen*, *Vornamen*, *Berufsstatus*, *Benutzername* und *Passwort* hinzufügen und anschließend mit seinem *Benutzernamen* und *Passwort* und *anmelden* kann.

Für die Web-Anwendungen wählen wir die Klasse *JApplet*. Wählen Sie dazu in der Menü-Leiste die Option: *Datei* >> *Neu* >> *JApplet*

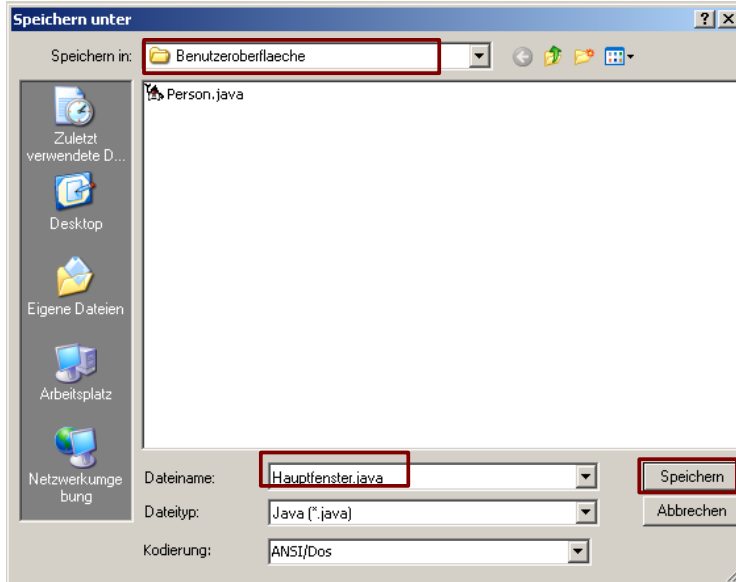


Ein *JApplet* ist das Gerüst für das *Hauptfenster* (die *Benutzeroberfläche*, *GUI*) einer *Web-Anwendung*. Für Anwendungen anderer Art, wie beispielsweise Software oder Mobile Anwendungen nutzen wir andere Gerüste (*JFrame* oder eine *XML-Datei*) für die *GUI*.

<sup>1</sup> *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 bitte *Hauptfenster.java* und bestätigen Sie Ihre Eingabe mit einem Klick auf *Speichern*.

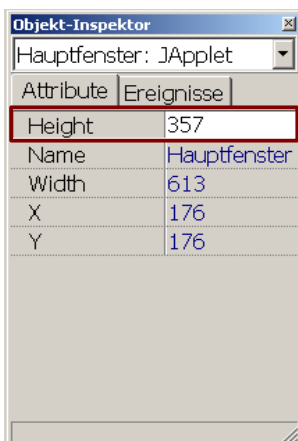
Mit dem Klick auf *Speichern* erstellt der Java-Editor zwei Dateien. Eine mit der Endung *.jfm* und eine mit der Endung *.java*.

*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 *Hauptfenster.java* enthält den dafür übersetzten 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 Code-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 357 (Pixel) und die Breite (*Width*) auf 613 (Pixel), wie im Bild angezeigt. 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.





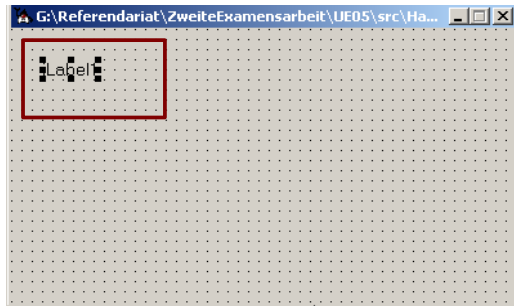
## Schritt 6: Eine Auswahl wichtiger *Komponenten* einfügen.



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:

Objekt-Inspektor	
jLabel1: JLabel	
Attribute	Ereignisse
Enabled	true
Font	(Font)
Height	16
HorizontalAlignment	LEFT
Name	jLabel1
Text	jLabel1
ToolTipText	
Visible	true
Width	44
X	24
Y	24



## 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:

Objekt-Inspektor

Attribute	Ereignisse
Enabled	true
Font	(Font)
Height	16
HorizontalAlignment	LEFT
Name	lbName
Text	Name:
ToolTipText	
Visible	true
Width	40
X	186
Y	16

Eigenschaften/  
Attribute

Eigenschaftsfeld  
für X

## 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.

Name: tfName

Vorname: tfVorname

Berufsstatus: tfBerufsstatus

Person ID: tfPersonID

Benutzername: tfBenutzername

Passwort: tfPasswort

Benutzer hinzufügen Eingabefelder leeren Anmelden

Meldung: tfMeldung

Kaufmännische  
Schule  
Wangen  
Wege zeigen, öffnen,  
gehen

Jlabels „lb“

JtextFields „tf“

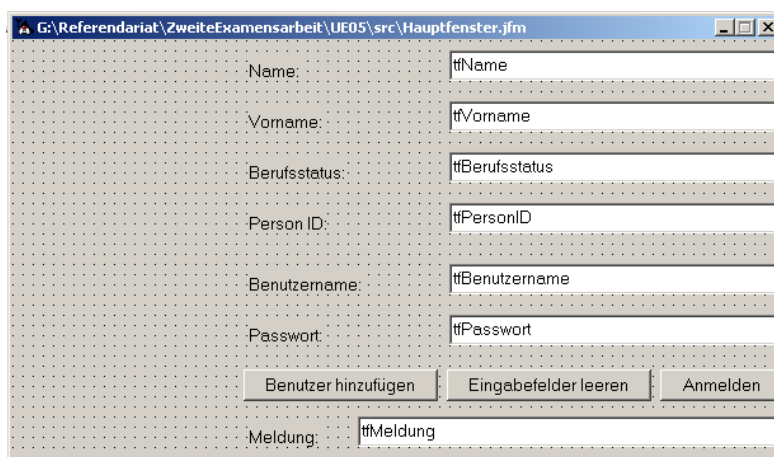
Jbuttons „bt“



### 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:

<i>Name</i>	<i>Typ</i>	<i>Text</i>	<i>Width</i>	<i>Height</i>	<i>X</i>	<i>Y</i>
lbName	JLabel	Name:	40	16	186	16
lbVorname	JLabel	Vorname:	58	16	186	56
lbBerufsstatus	JLabel	Berufsstatus:	76	16	186	96
lbPersonID	JLabel	Person ID:	62	16	186	136
lbBenutzername	JLabel	Benutzername:	89	16	186	184
lbPasswort	JLabel	Passwort:	58	16	186	224
lbMeldung	JLabel	Meldung:	55	16	186	304
tfName	JTextField	tfName	265	24	344	8
tfVorname	JTextField	tfVorname	265	24	344	48
tfBerufsstatus	JTextField	tfBerufsstatus	265	24	344	88
tfPersonID	JTextField	tfPersonID	265	24	344	128
tfBenutzername	JTextField	tfBenutzername	265	24	344	176
tfPasswort	JTextField	tfPasswort	265	24	344	216
tfMeldung	JTextField	tfMeldung	337	24	272	296
btBenutzerHinzufuegen	JButton	Benutzer hinzufügen	153	25	182	259
btEingabenLeeren	JButton	Eingabefelder leeren	161	25	342	259
btAnmelden	JButton	Anmelden	97	25	510	259



[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 btEingabenLeeren_ActionPerformed(ActionEvent evt) {  
    \\ TODO hier Quelltext einfügen  
}
```

```
public void btEingabenLeeren_ActionPerformed(ActionEvent evt) {  
    tfPersonID.setText("");  
    tfName.setText("");  
    tfVorname.setText("");  
    tfBerufsstatus.setText("");  
    tfBenutzername.setText("");  
    tfPasswort.setText("");  
    tfMeldung.setText("");  
    // TODO hier Quelltext einfügen  
}
```

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 Development Kits, JDK) den Inhalt einer *.java-Datei* in eine *.class-Datei*, diese Datei wird dann ausgeführt und das Ergebnis wird angezeigt. In unserem Fall wird die gerade erstellte Benutzeroberfläche angezeigt.

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





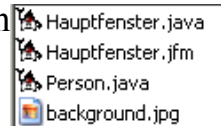
## GRAFIKEN AUF DER BENUTZEROBERFLÄCHE EINES JAPPLETS

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

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

***Dazu sind die kommenden 5 Schritte notwendig!***

Kopieren Sie die Datei *background.jpg* (in *Eigene Dateien\JanC\UE05*) in Ihr Verzeichnis *Benutzeroberflaeche*. Überprüfen Sie ob alle folgenden Dateien enthalten sind:



### 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* (*Hauptfenster.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, 613, 357);
```

folgenden Quellcode:

```
//Hintergrundgrafik laden
try {
    image = ImageIO.read(getClass().getResource("background.jpg"));
}
catch (IllegalArgumentException iae) {
    JOptionPane.showMessageDialog(this, "Grafikdatei nicht gefunden!");
}
catch (IOException ioe) {
    JOptionPane.showMessageDialog(this, "Fehler beim Einlesen der Grafikdatei!");
}
bg.setSize(179, 357);
bg.setVisible(true);
cp.add(bg);
```

Diagramme zur Erklärung:

- Ein Kasten mit dem Text "Dateiname der Grafik" zeigt auf den String `"background.jpg"` in der `getResource`-Methode.
- Ein Kasten mit dem Text "Größe des Bildes  
setSize(Width, Height)" zeigt auf die `setSize(179, 357)`-Anweisung.

*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 17*.

## 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 Hintergrundgrafik-Komponente
public class BackgroundPanel extends JPanel
{
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if(image != null) {
            g.drawImage(image, 0, 0, this);
        }
    }
}
```

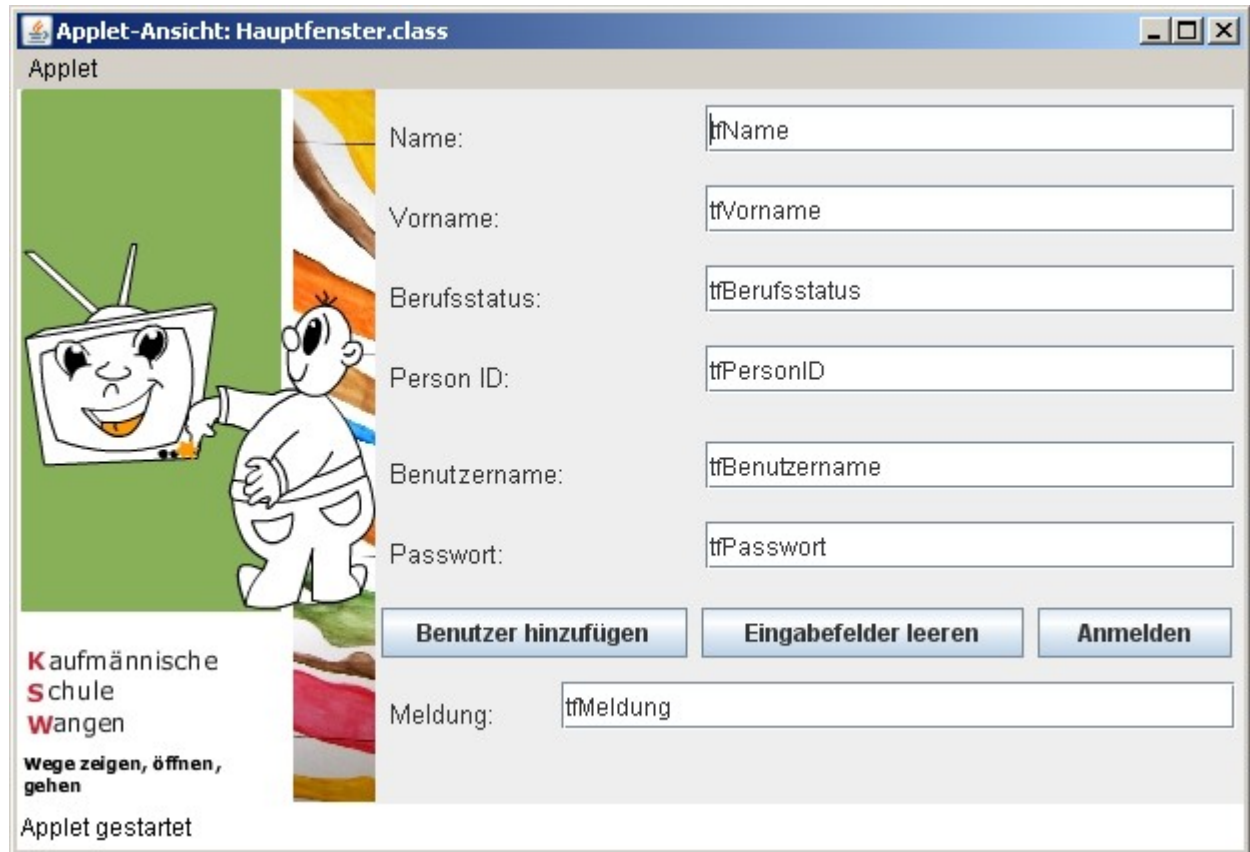
Diagramme zur Erklärung:

- Ein Kasten mit dem Text "Mit dieser Klasse Erstellen eine noch nicht vorhandene Komponente. Diese (bg) enthält unsere Grafik" zeigt auf die `BackgroundPanel`-Klasse.
- Ein Kasten mit dem Text "Image enthält die Grafik background.jpg" zeigt auf die `image`-Variable in der `drawImage`-Methode.

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]

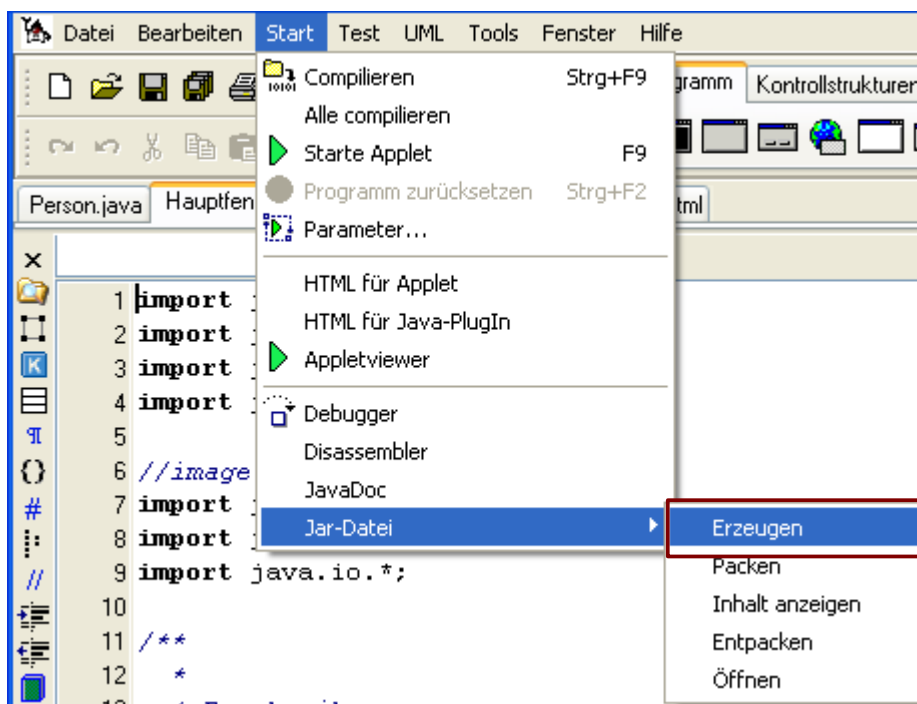
Im Ergebnis sollte die Grafik angezeigt werden. Falls Fehler angezeigt werden, sind diese häufig auf fehlende oder überflüssige Klammern zurückzuführen. Überprüfen Sie gegebenenfalls die Klammersetzung in der Datei „Hauptfenster.java“.



## 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. Überprüfen Sie, ob alle Dateien vorhanden sind:



Falls ja, öffnen Sie die Datei Hauptfenster.java und nutzen sie den Jar-Generator des Java-Editors, indem Sie in der Menü-Leiste die Option:

#### 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 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:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Log-In-GUI</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<h1>Log-In-GUI</h1>
<hr>
<applet code="Hauptfenster" archive="Hauptfenster.jar" width="613" height="357">
</applet>
<hr>
</body>
</html>
```

applet-Einbettung



## 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:

```
btAnmelden.setBounds(510, 259, 97, 25);
btAnmelden.setText("Anmelden");
btAnmelden.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        btAnmelden_ActionPerformed(evt);
    }
});
cp.add(btAnmelden);
```

**<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:

```
public void btAnmelden_ActionPerformed(ActionEvent evt) {
    // TODO hier Quelltext einfügen
}
```

**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> () {...}
```



## REFLEXION

### ZWEIWERDIGE (BOOLSCH) ATTRIBUTE: Fachklasse `Person.java`

#### Boolsche Attribute: Methode `identitaetPruefen(...){...}` der Fachklasse `Person.java` nachrüsten.

Wenn der Benutzer auf die *Schaltfläche Anmelden* klickt, soll die Identität anhand des *Benutzernamens* und des *Passworts* geprüft werden. Die Prüfung erfolgt anhand der *IF-Anweisung* (einer Kontrollstruktur) in der Methode `identitaetPruefen(...){...}`. Damit wir feststellen können ob die Prüfung erfolgreich war, ergänzen wir in der Klasse `Person.java` ein Attribut:

```
private boolean erfolgreich = false;
```

Das Attribut steht damit standardmäßig auf dem Wert *falsch*. Ergänzen Sie dann die dazugehörige *Getter- und Setter-Methode*:

```
public boolean getErfolgreich() {  
    return erfolgreich;  
}  
  
public void setErfolgreich(boolean pErfolgreich) {  
    this.erfolgreich = pErfolgreich;  
}
```

Im *IF-Zweig* der Methode `identitaetPruefen(...){...}` setzen wir für den Fall, dass die Prüfung erfolgreich war auf den Wert *true*, ergänzen sie deshalb im *IF-Zweig* die Zeile: `this.setErfolgreich(true);`

#### Der String-Vergleich: Die `equals()`-Methode

Methode `identitaetPruefen(...)`:

```
//Variante: IF-Anweisung  
public void identitaetPruefen(String pBenutzername, String pPasswort) {  
    String mBn = this.getBenutzername();  
    String mPs = this.getPasswort();  
    if(mBn.equals(pBenutzername) && mPs.equals(pPasswort)){  
        this.setErfolgreich(true);  
        System.out.println("Anmeldung erfolgreich!!");  
    }else{  
        System.out.println("Anmeldung fehlgeschlagen!!");  
    }  
}
```

*Erläuterung:*

Im Gegensatz zu den *primitiven Datentypen* (*boolean, char, int, long, float und double*) ist *String* ein komplexer Datentyp. Bei komplexen Datentypen benötigen wir für den Vergleich (Gleichheit) die Methode: `<objektname1>.equals(objektname2)`





## Verwendung von equals(): Die Methode `sucheName (...)` { ... }

In Ihrem letzten Arbeitsblatt haben Sie im Rahmen der Zusatzaufgabe die Methode `sucheName(...){...}` in der Fachklasse `Person.java` implementiert (programmiert). Verändern bzw. ergänzen Sie diese Methode, wie folgt:

```
public void sucheName(String pName) {
    String mName = this.getName();
    if (mName.equals(pName)) {
        System.out.println("Name gefunden!");
        System.out.println("PersonID: "+this.getPersonID());
        System.out.println("Name: "+this.getName());
        System.out.println("Vorname: "+this.getVorname());
        System.out.println("Berufsstatus: "+this.getBerufsstatus());
    } else {
        System.out.println("Name nicht gefunden!");
    }
}
```

Hinweis:

- `pName` ist die über die Benutzeroberfläche erfolgte Eingabe des Benutzers.
- `mName` ist der im System gesetzte Wert für den Namen einer Person.

Wir vergleichen in der Bedingung wieder zwei Attribute des komplexen Datentyps String, deshalb verwenden wir auch hier die Methode:  
`<objektname1>.equals(objenname2)`

## Frage: Wie realisieren wir den Vergleich bei einem primitiven Datentyp (z.B. `int alter`)?

Angenommen das Attribut `alter` vom Datentyp Integer existiert (inkl. der Getter- und Setter-Methode)

```
public void sucheAlter(int pAlter) {
    int mAlter = this.getAlter();
    if (this.getAlter() == pAlter) {
        System.out.println("Person mit dem Alter " + this.getAlter() + " gefunden.");
    } else {
        System.out.println("Es wurde keine Person mit dem Alter " + this.getAlter() + " gefunden.");
    }
}
```

Hinweis:

- `pAlter` ist die über die Benutzeroberfläche erfolgte Eingabe des Benutzers.
- `mAlter` ist der im System gesetzte Wert für das Alter einer Person.

Wir vergleichen in der Bedingung zwei Attribute des primitiven Datentyps Integer, deshalb verwenden wir hier den Vergleichsoperator `==` :  
`<objektname1> == <objenname2>`