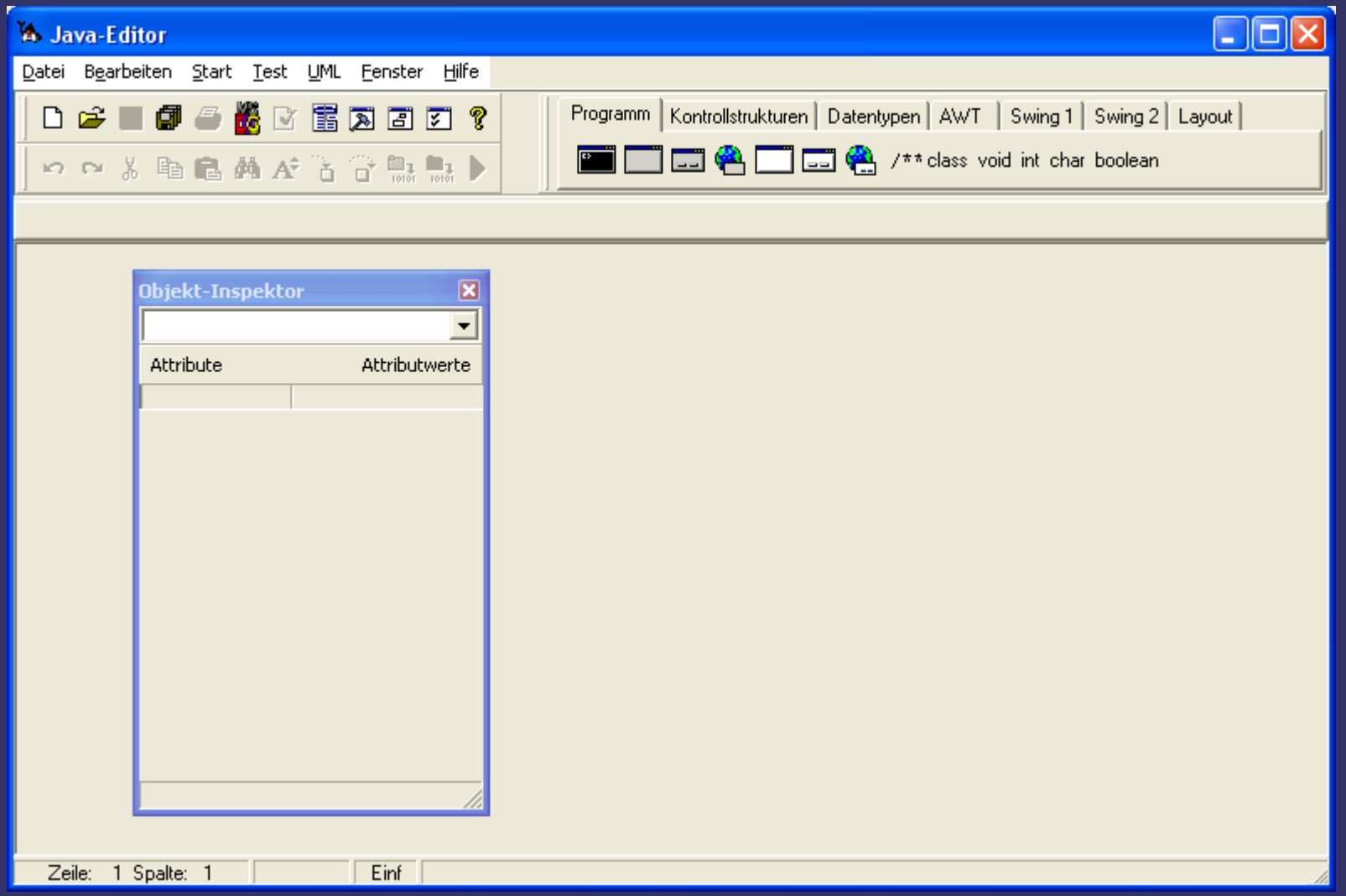
GUI mit Java - Editor

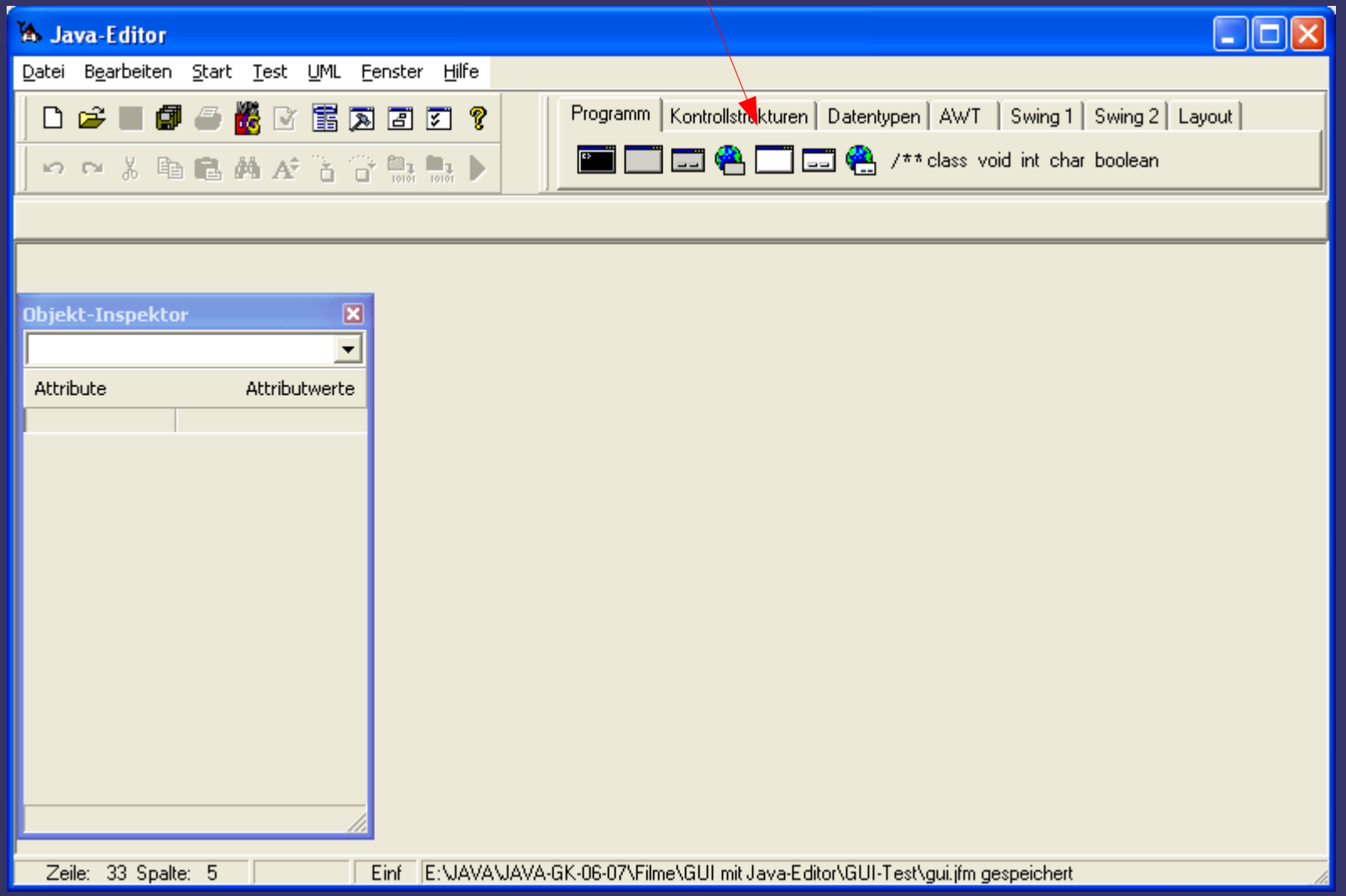
GUI mit Java - Editor

Java – Editor starten



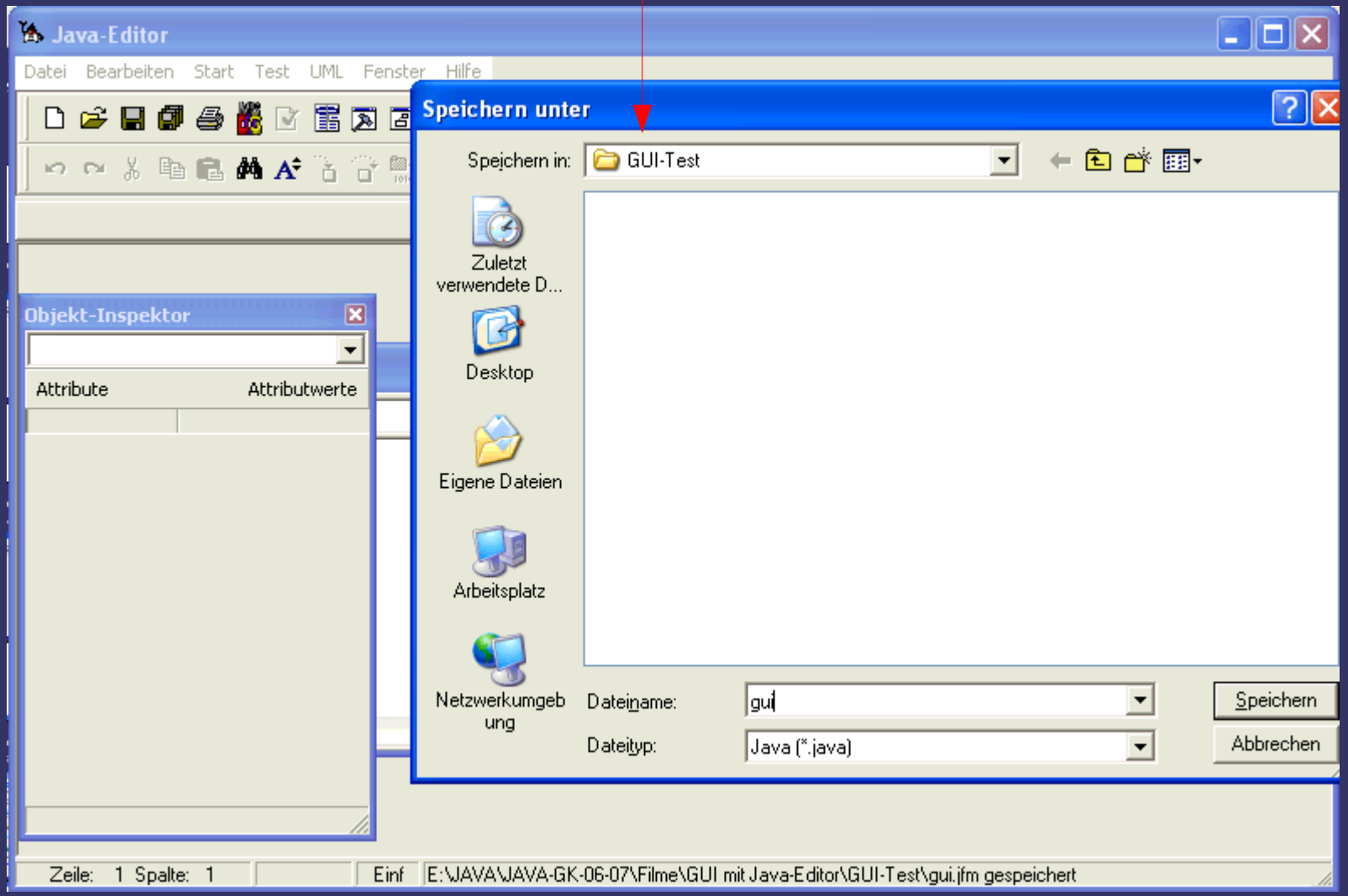
GUI mit Java - Editor

neues Formular mit Button JFrame



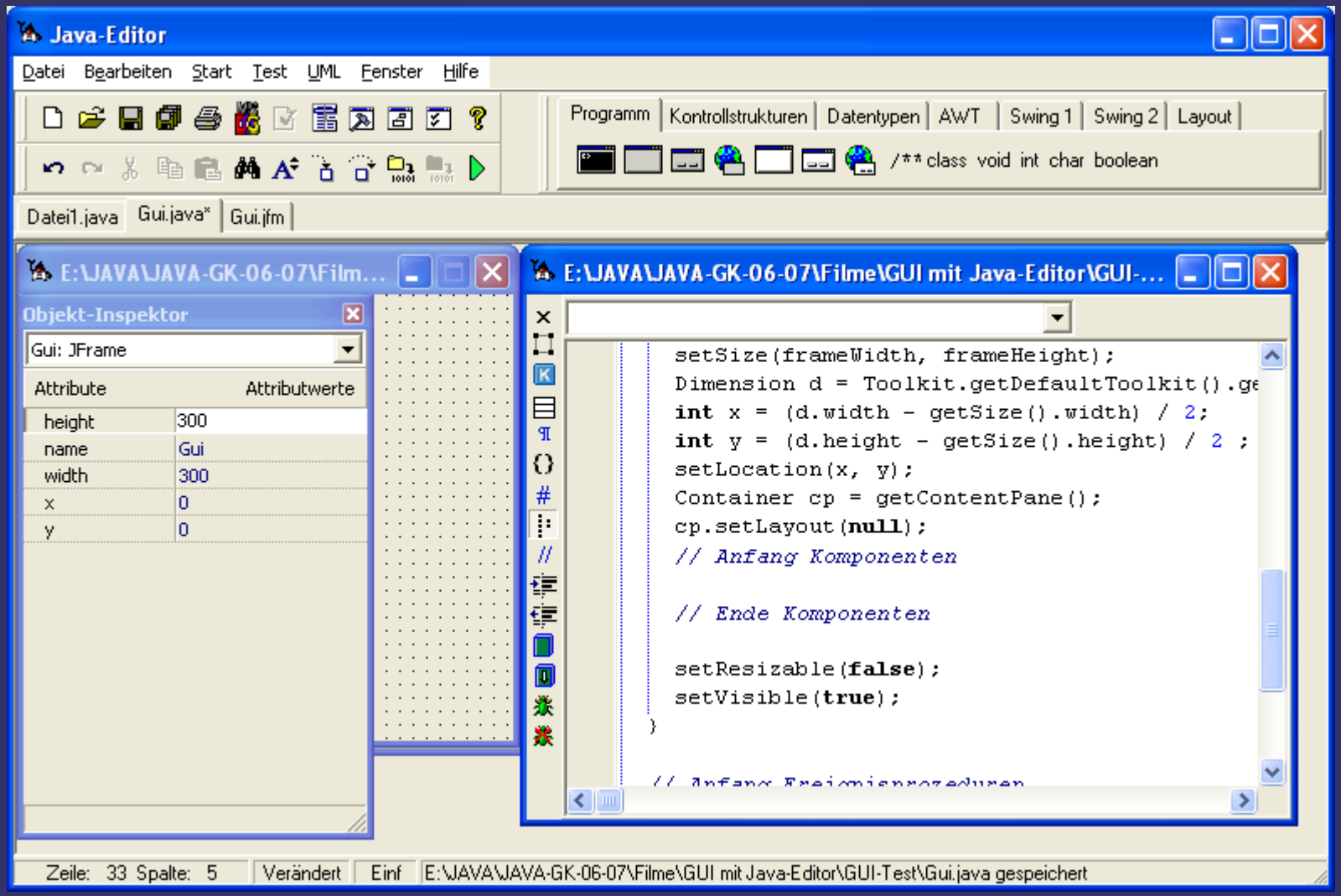
GUI mit Java - Editor

neues Formular in neuem Projektordner abspeichern



GUI mit Java - Editor

Das Projekt lässt sich schon übersetzen und starten!



GUI mit Java - Editor

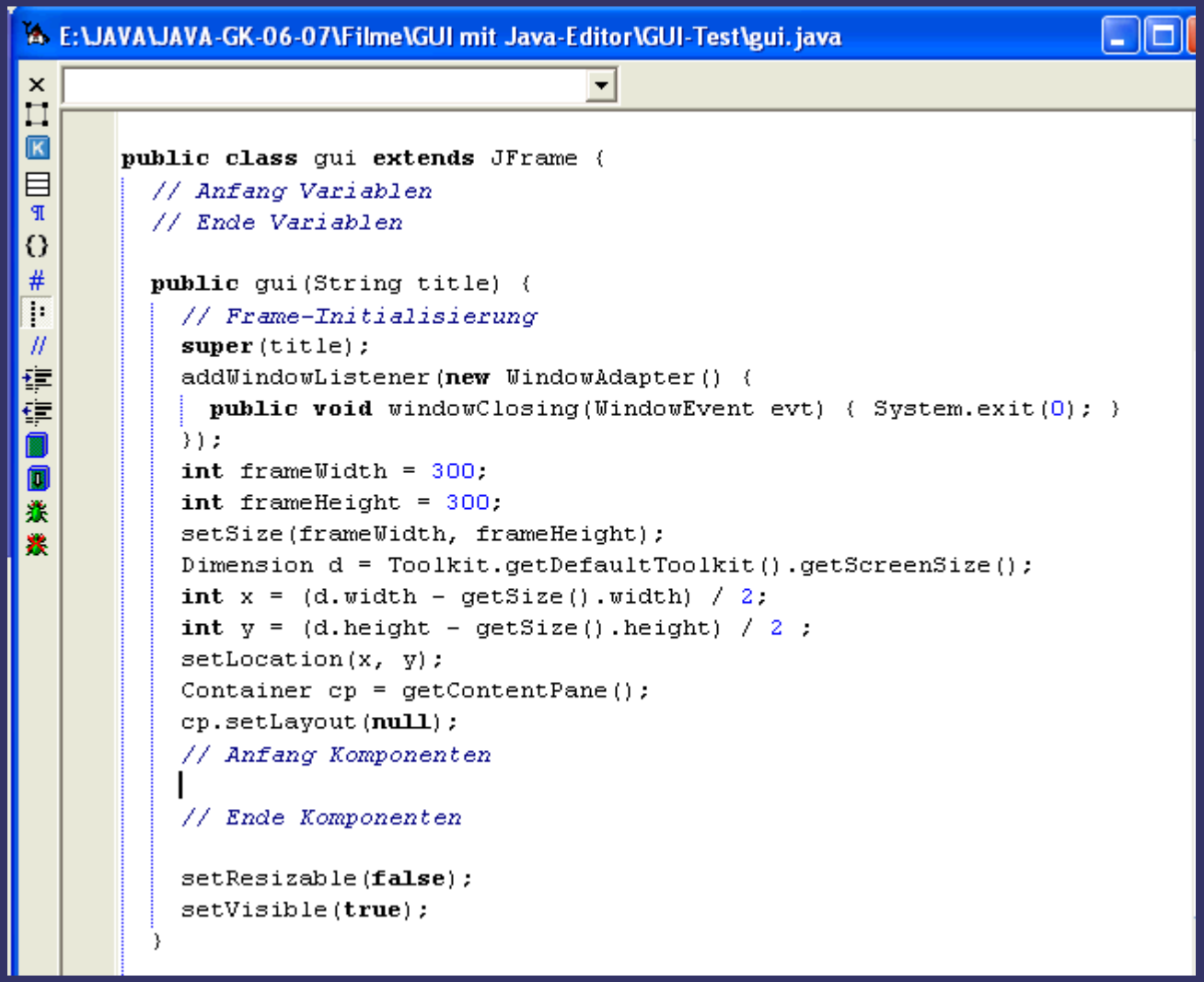
Das Projekt lässt sich schon übersetzen und starten, ...

allerdings macht es noch nichts.

Das überrascht aber ja auch nicht.

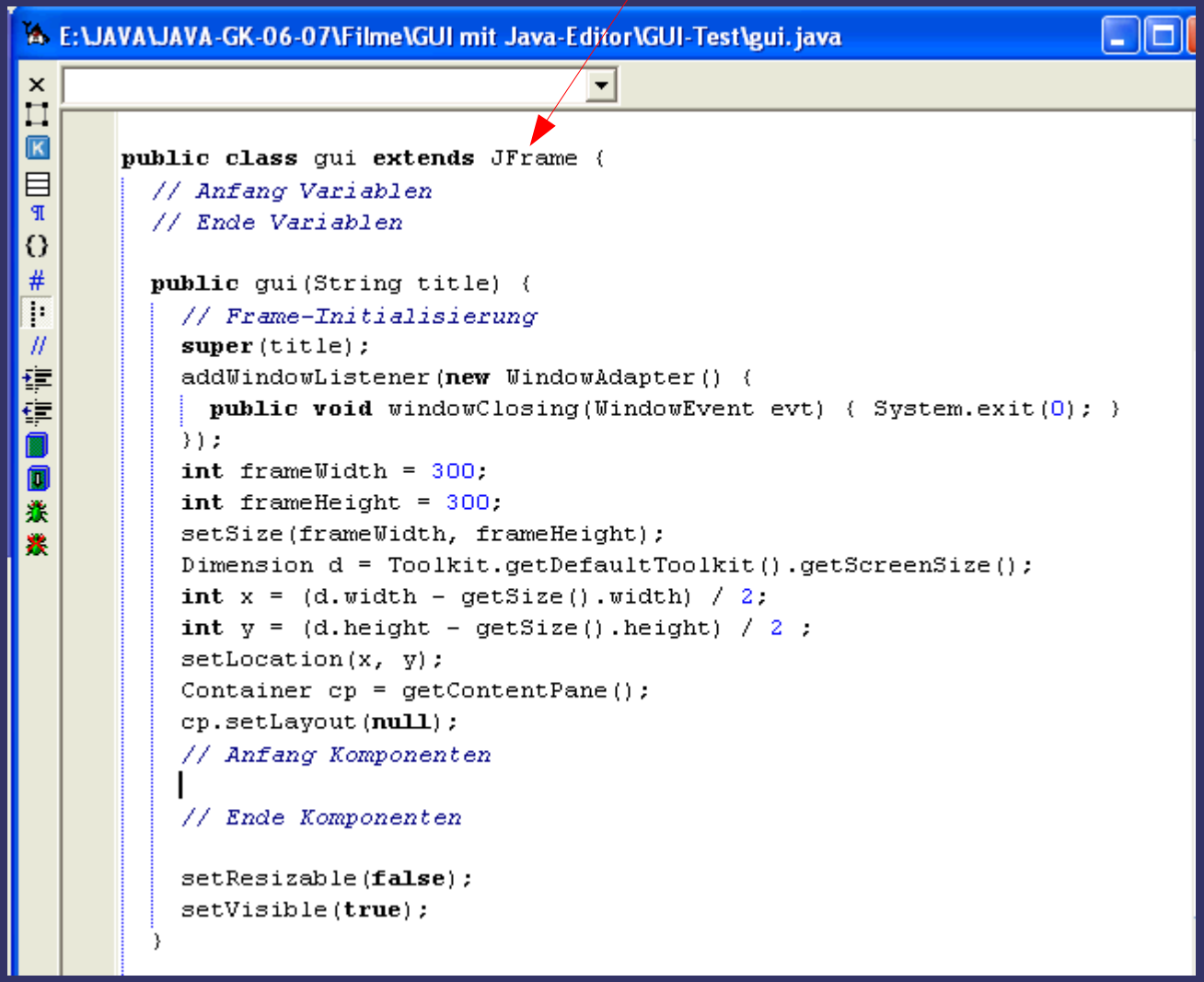
GUI mit Java - Editor

Der Text enthält aber schon alle Grundlagen:



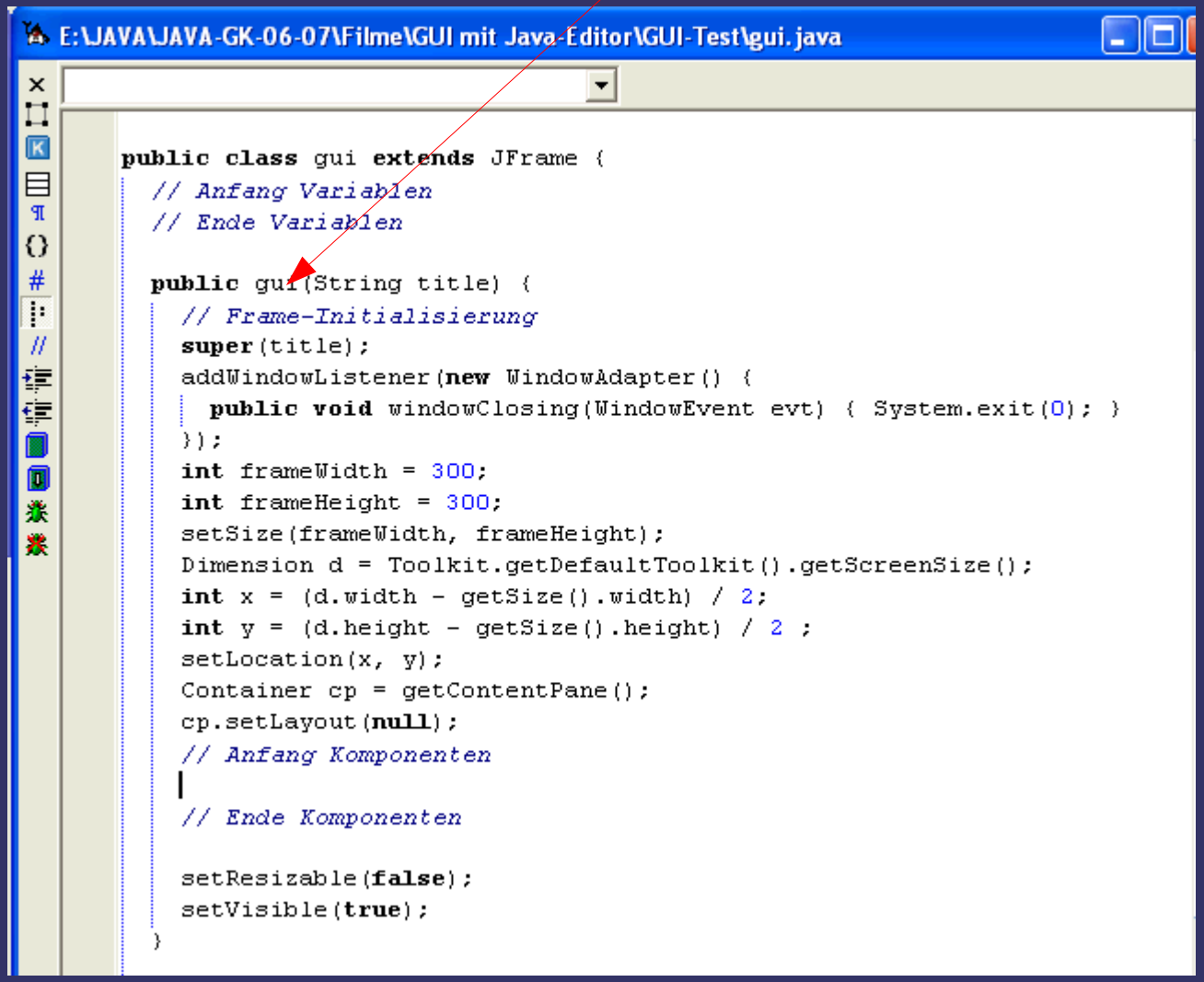
GUI mit Java - Editor

Unsere Klasse erbt von JFrame, ...



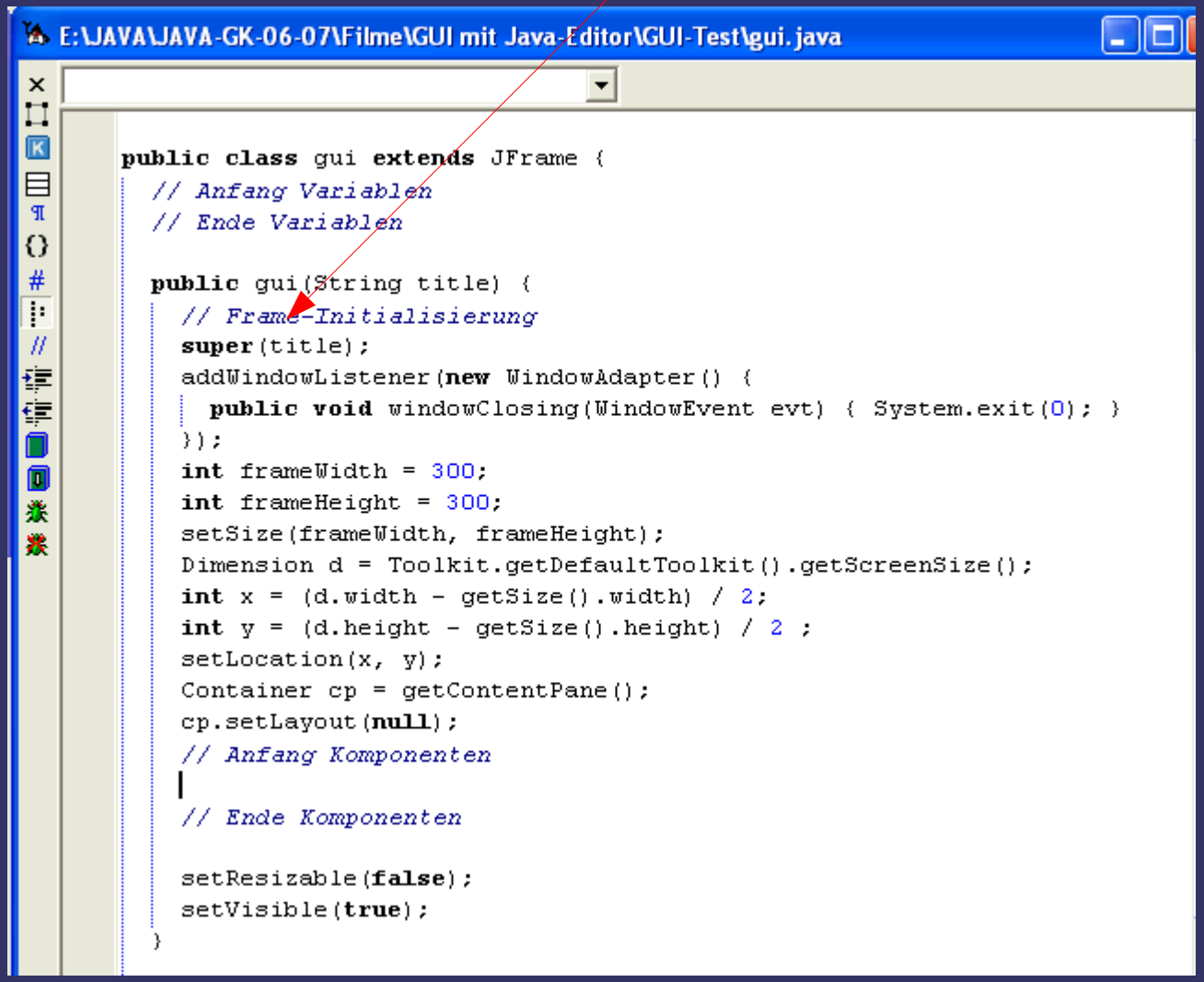
GUI mit Java - Editor

... enthält einen Konstruktor, der ...



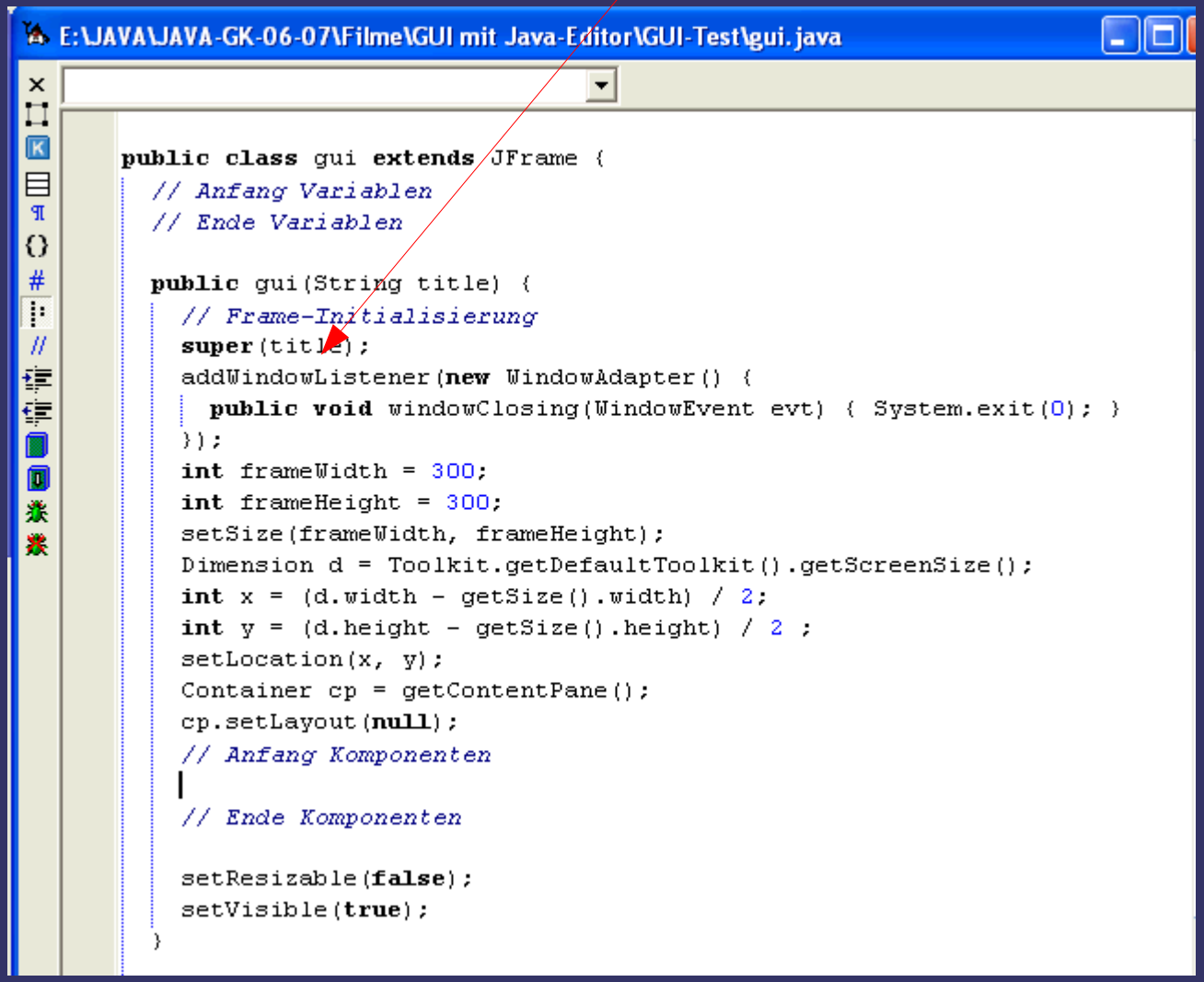
GUI mit Java - Editor

... den Konstruktor von JFrame aufruft, ...



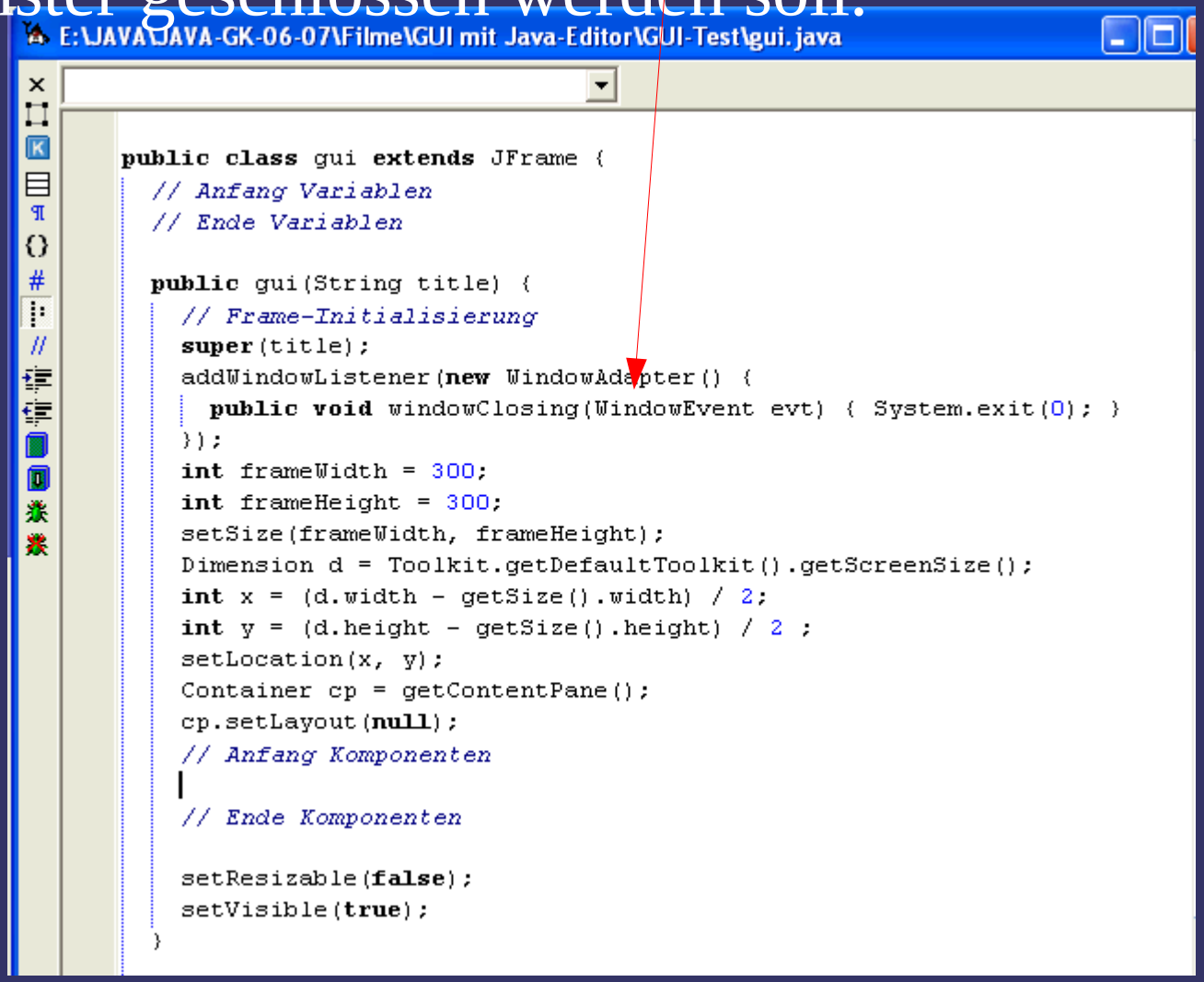
GUI mit Java - Editor

... ihm einen WindowListener hinzufügt, ...



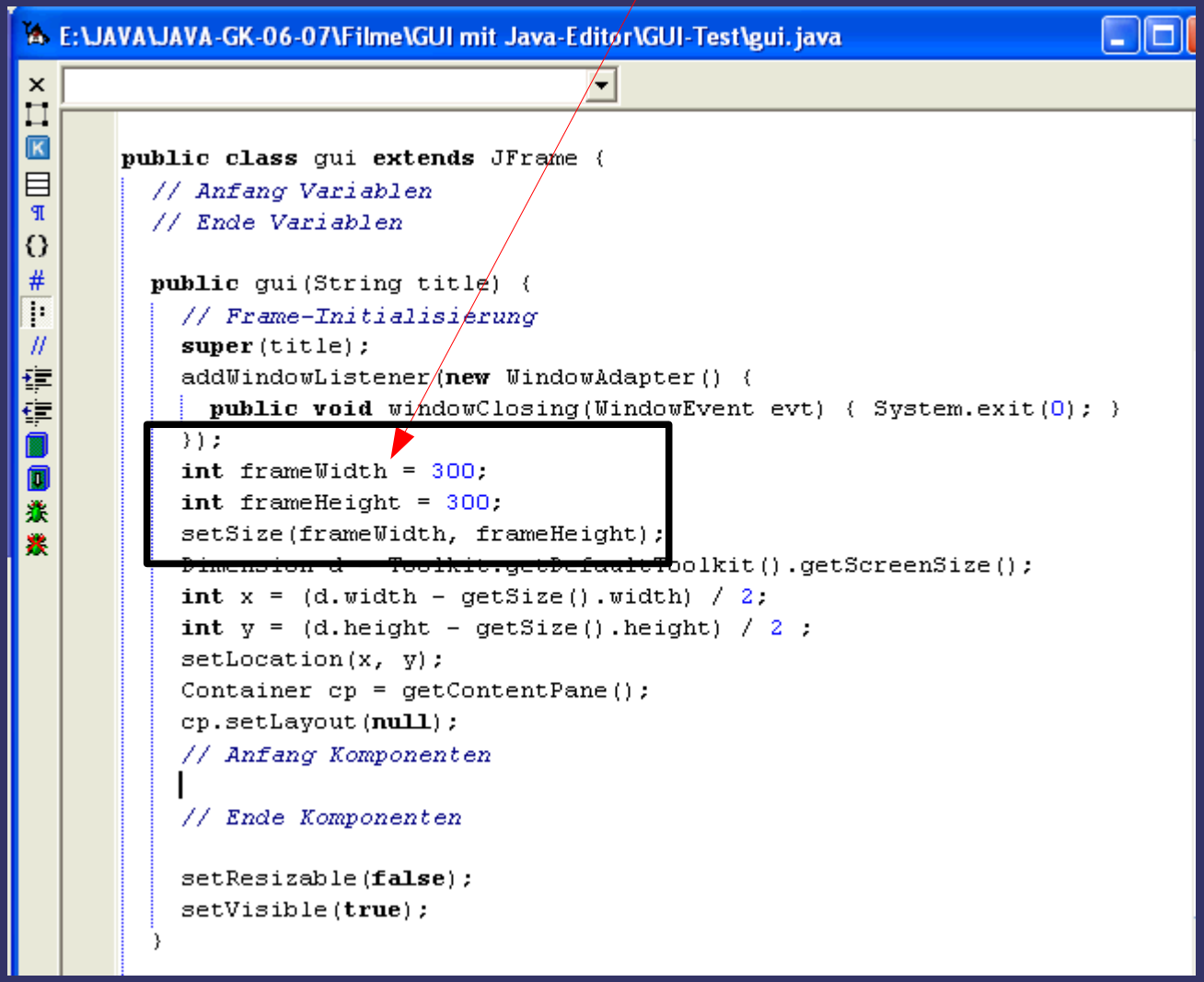
GUI mit Java - Editor

... damit wir das Ereignis verarbeiten können, dass unser Fenster geschlossen werden soll.



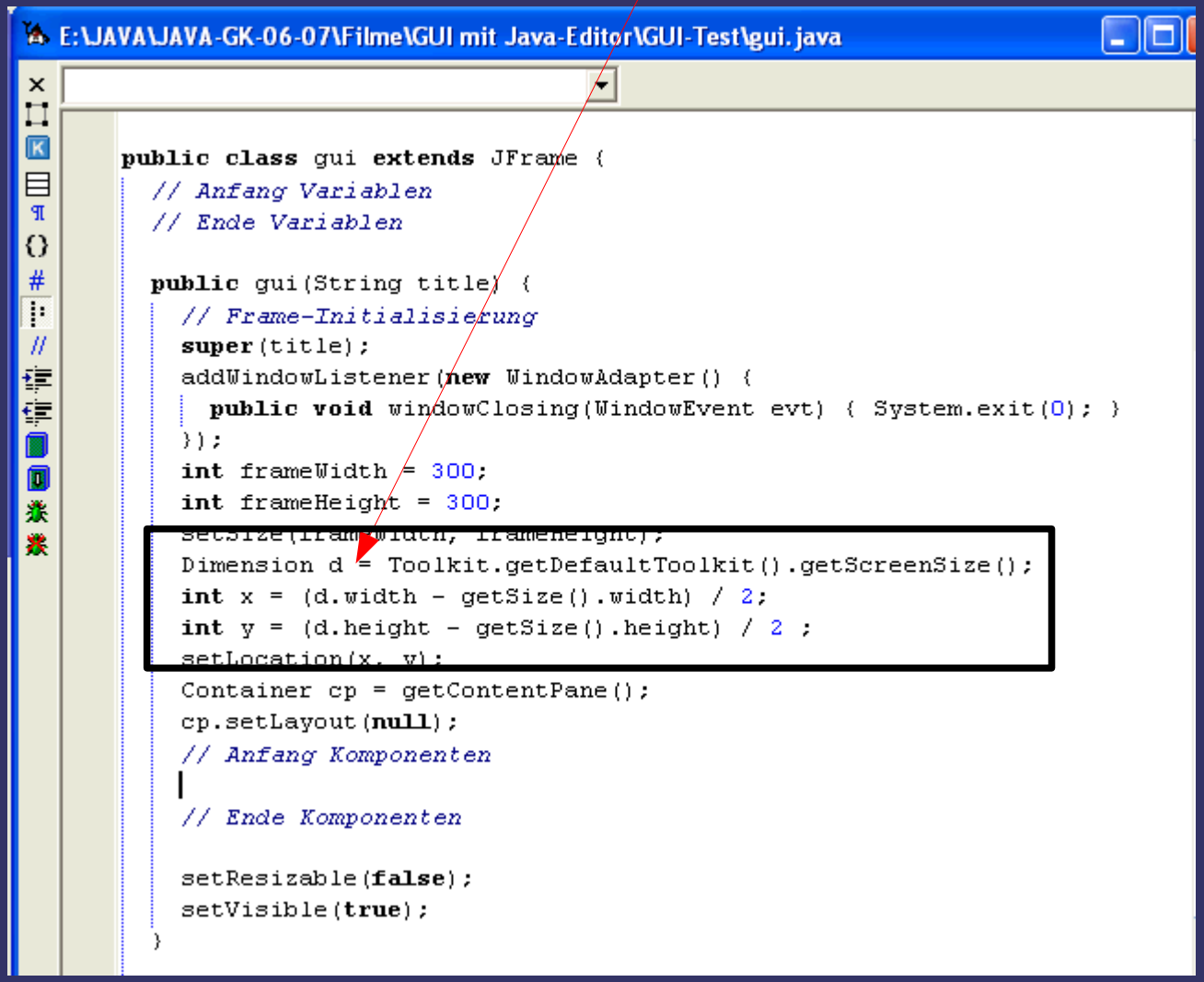
GUI mit Java - Editor

Im Konstruktor wird die Fenstergröße vorgegeben, ...



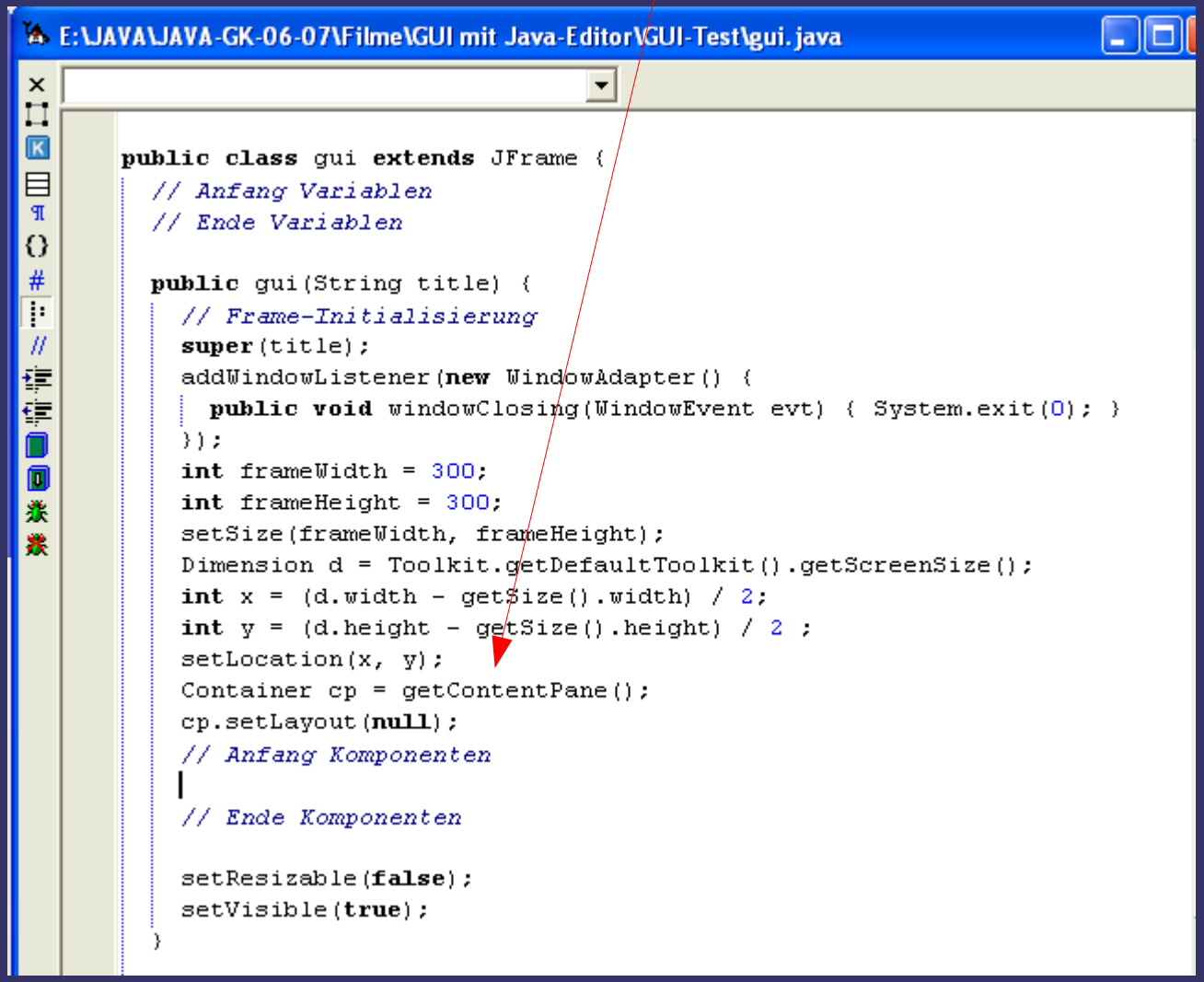
GUI mit Java - Editor

... und die Fensterposition, ...



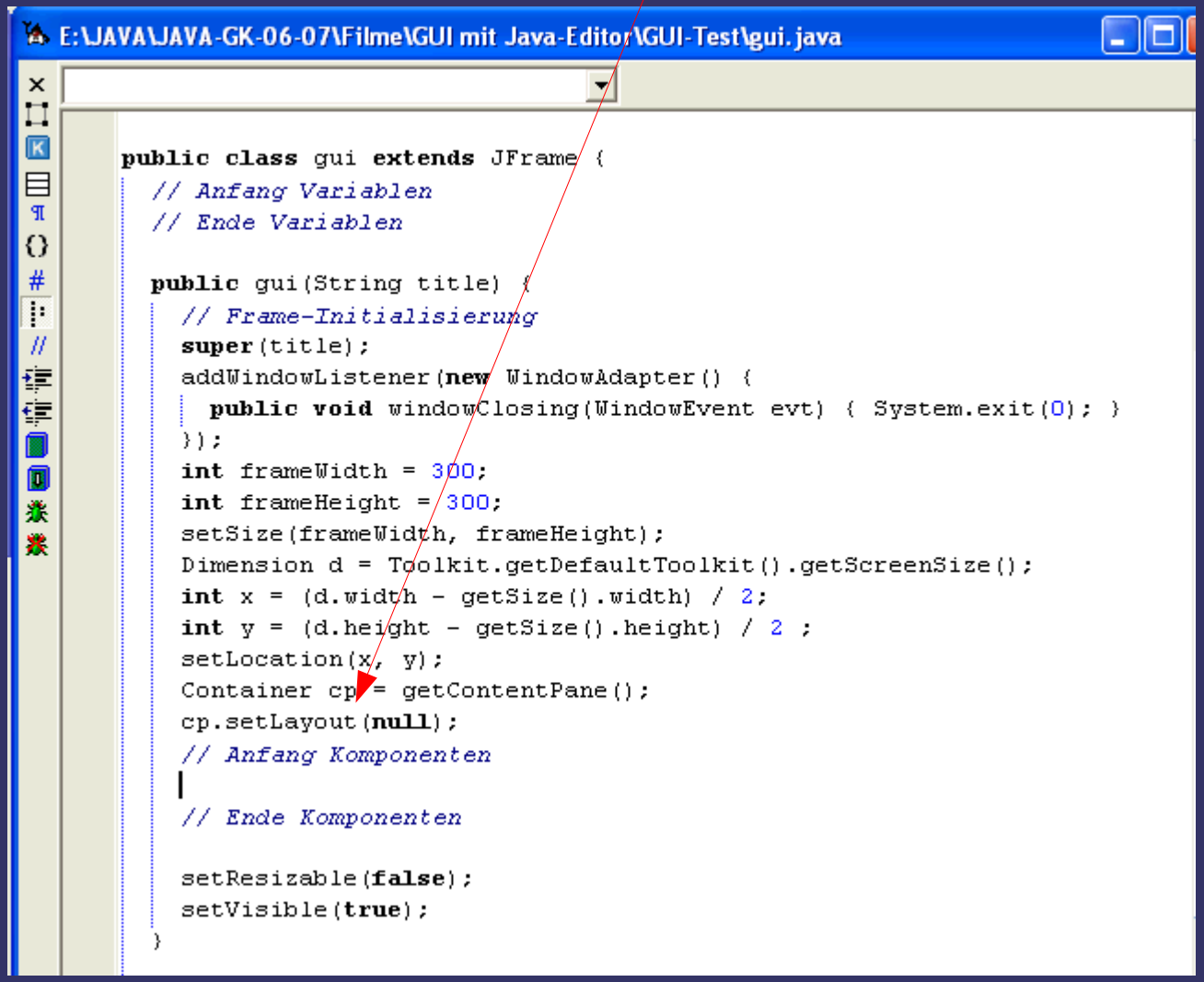
GUI mit Java - Editor

ContentPane ist die Inhaltsfläche des Fensters, ...



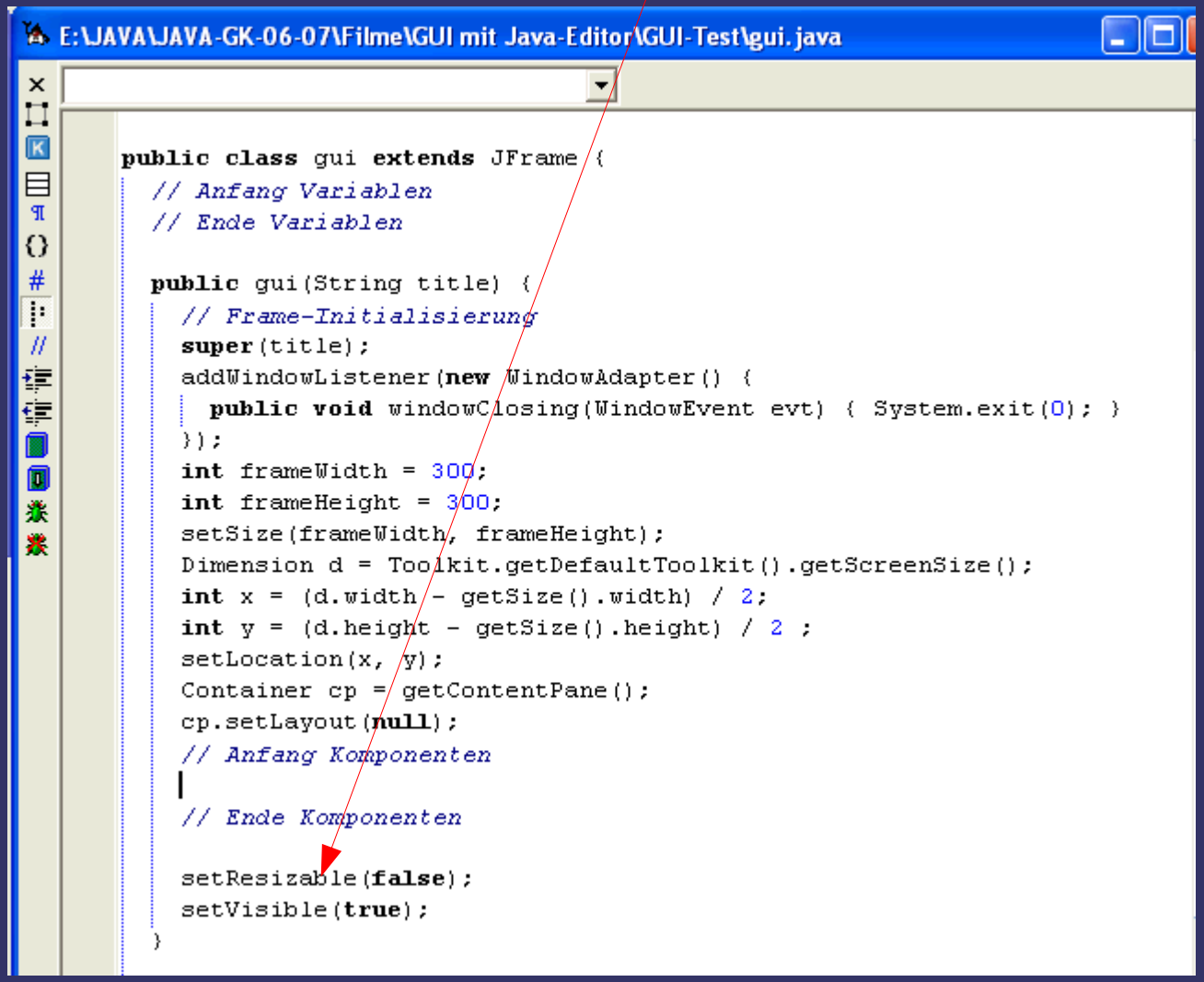
GUI mit Java - Editor

... die keinen LayoutManager bekommt.



GUI mit Java - Editor

Sie wird mit unveränderlicher Größe sichtbar gemacht.



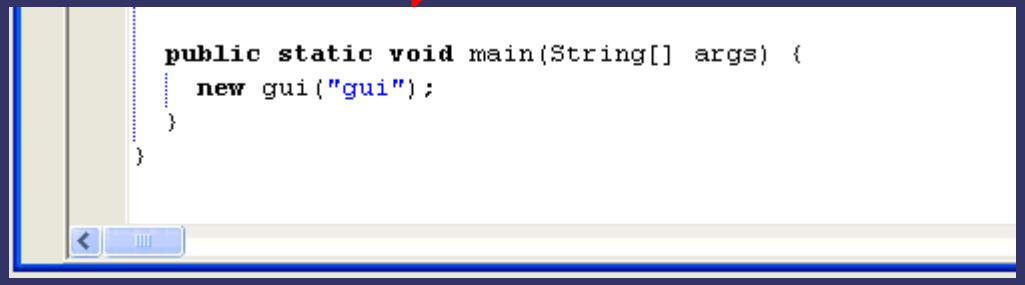
GUI mit Java - Editor

Im Kopf finden wir noch die notwendigen Importe, hier mit .* , um jeweils alles zu importieren.

Am Ende der Klassendefinition finden wir die

main – Methode

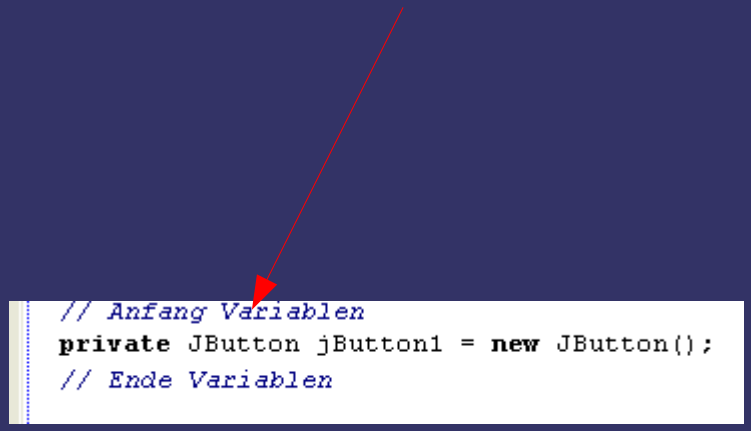
Sie ermöglicht das Erstellen eines ausführbaren Programmes.



GUI mit Java - Editor

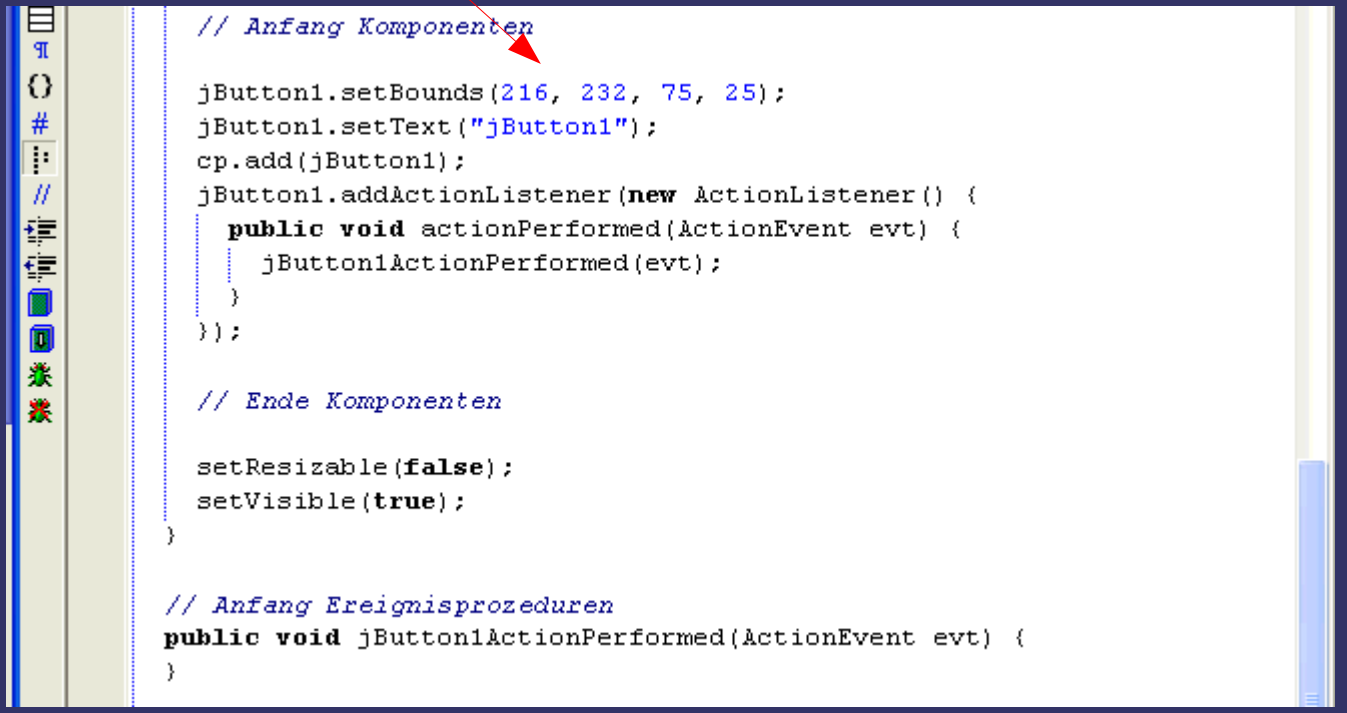
Wir klicken die Lasche Swing1 an und können nun einen JButton hinzufügen, indem wir ihn irgendwo auf dem Formular einfügen.

Im Programmtext wird er erzeugt, ...



GUI mit Java - Editor

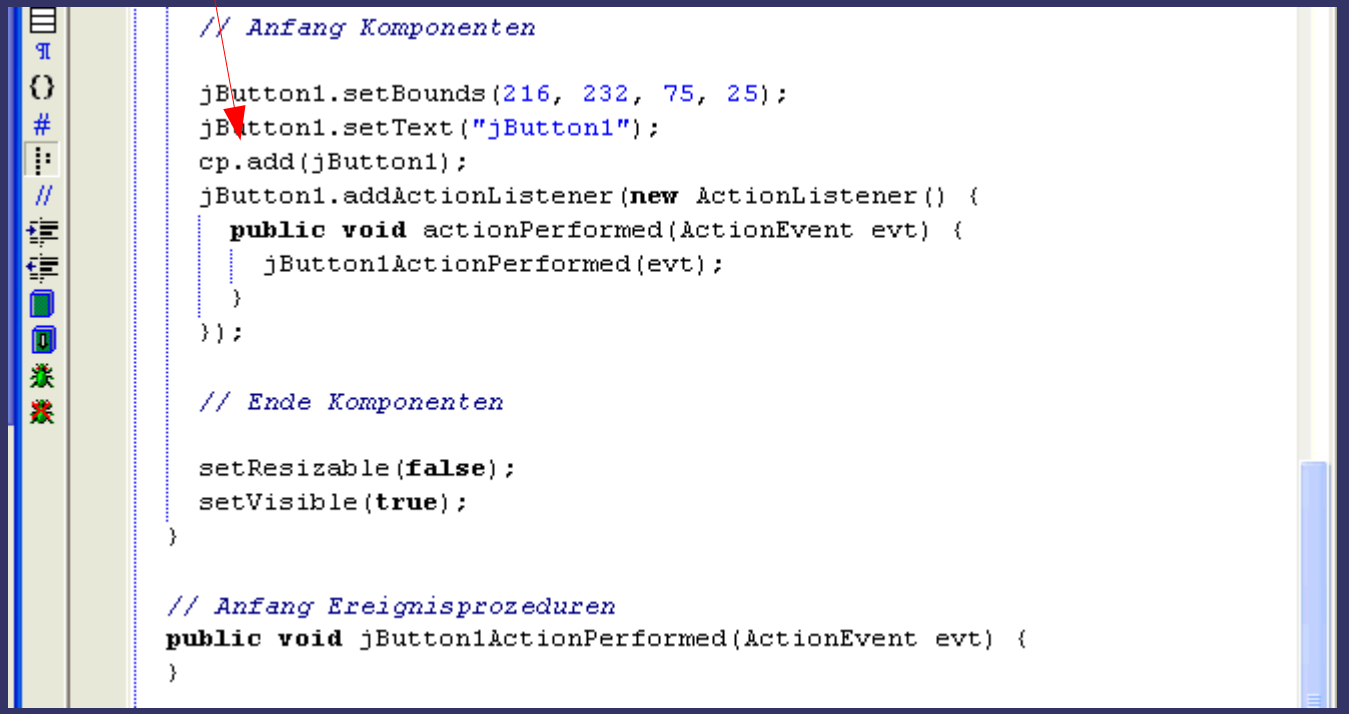
... Position und Beschriftung festgelegt, ...



```
    // Anfang Komponenten

    jButton1.setBounds(216, 232, 75, 25);
    jButton1.setText("jButton1");
    cp.add(jButton1);
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        jButton1ActionPerformed(evt);
      }
    });

    // Ende Komponenten

    setResizable(false);
    setVisible(true);
  }

  // Anfang Ereignisprozeduren
  public void jButton1ActionPerformed(ActionEvent evt) {
  }
```

GUI mit Java - Editor

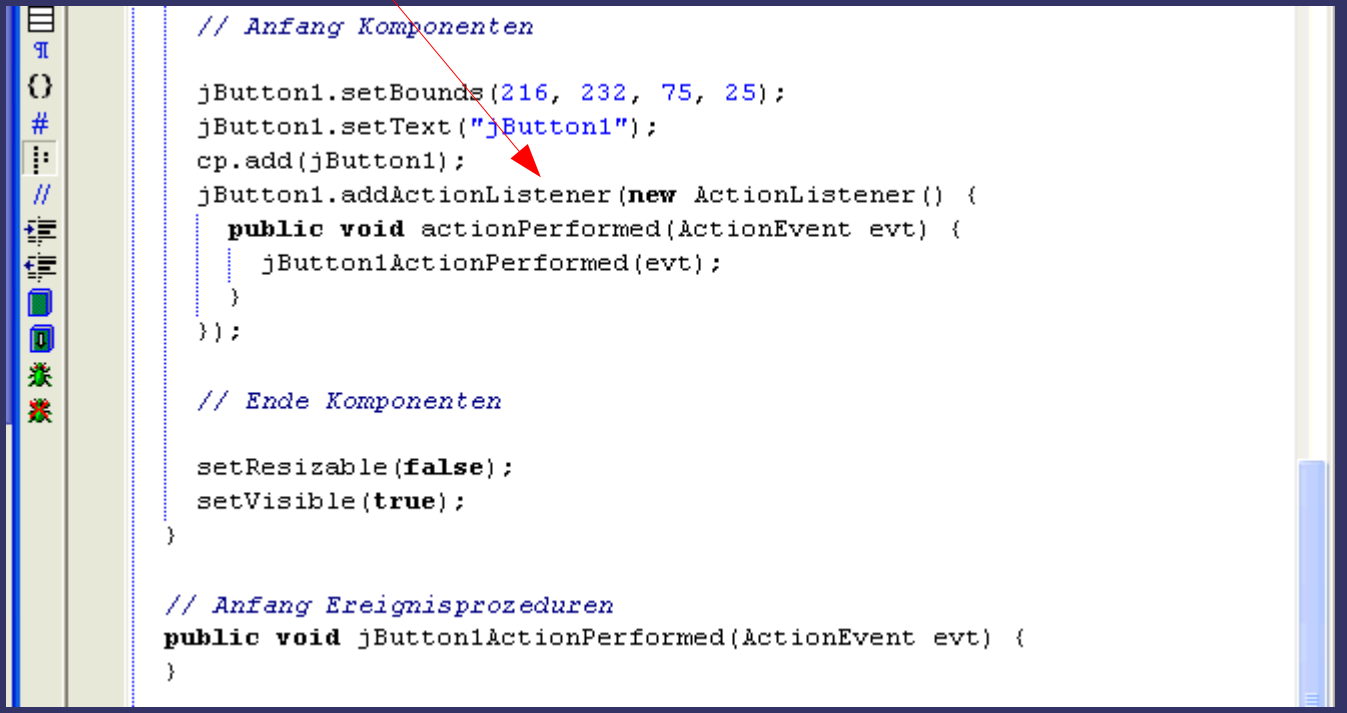
... er wird der ContentPane hinzugefügt, ...



```
    // Anfang Komponenten

    jButton1.setBounds(216, 232, 75, 25);
    jButton1.setText("jButton1");
    cp.add(jButton1);
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        jButton1ActionPerformed(evt);
      }
    });

    // Ende Komponenten

    setResizable(false);
    setVisible(true);
  }

  // Anfang Ereignisprozeduren
  public void jButton1ActionPerformed(ActionEvent evt) {
  }
```

GUI mit Java - Editor

... und ein ActionListener hinzugefügt.



```
    // Anfang Komponenten

    jButton1.setBounds(216, 232, 75, 25);
    jButton1.setText("jButton1");
    cp.add(jButton1);
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        jButton1ActionPerformed(evt);
      }
    });

    // Ende Komponenten

    setResizable(false);
    setVisible(true);
  }

  // Anfang Ereignisprozeduren
  public void jButton1ActionPerformed(ActionEvent evt) {
  }
```

GUI mit Java - Editor

ActionListener ist eine Schnittstelle, die allein eine Methode besitzt, die implementiert werden muss:

```
public void actionPerformed(ActionEvent evt) {
        jButton1ActionPerformed(evt);
}
```

Hier ruft sie allein eine andere Methode auf, die wir unten bei den „Ereignismethoden“ finden.

GUI mit Java - Editor

Sie ist ebenfalls nur angelegt, macht also noch nichts.

```
public void jButton1ActionPerformed(ActionEvent evt)
{
}
```

An dieser Stelle ist die tatsächliche Aktion zu programmieren, die das Anklicken des Buttons auslösen soll.

GUI mit Java - Editor

Erläuterung zum ActionListener

ActionListener

Wir gehen zurück zu einer vorigen Folie.
Dort hieß es:

ActionListener ist eine Schnittstelle, die ...

Wir fügen aber dem JButton mit der Methode

addActionListener(...);

ein Objekt hinzu.

Ein Objekt muss mit new erzeugt werden, üblicherweise rufen wir damit den in der Klassendefinition gegebenen Konstruktor auf.

ActionListener

Nun finden wir aber keine in unserem Projekt definierte Klasse dazu.

Der ActionListener wird statt dessen mit Hilfe einer

anonymen

inneren

Klasse definiert!

ActionListener

anonyme Klasse:

Die Klasse bekommt keinen Namen.

Sie kann daher genau einmal instanziiert (=ein Objekt erzeugt) werden, da ich diesen Konstruktor nur an dieser Stelle benutzen kann.

ActionListener

innere Klasse:

Die Klasse ist innerhalb der Gui-Klasse definiert!

Sie kann nur dort verwendet werden.

WICHTIG:

Jede innere Klasse hat -da sie zu der Umgebung der umgebenden Klasse gehört- Zugriff auf alle Attribute und Methoden dieser Klasse.

ActionListener

```
jButton1.addActionListener(      Methode vom JButton
    new ActionListener()     Konstruktoraufruf (interface-Name!)
    {                        Beginn der Klassendefinition
     public void actionPerformed(ActionEvent evt)
        {                    Beginn der Methodendefinition
      jButton1ActionPerformed(evt);
            Aufruf  einer Methode der umgebenden Klasse
            (hier könnte auch direkt stehen, wie das Ereignis
            behandelt werden soll)
     }                       Ende der Methodendefinition
    }                        Ende der Klassendefinition
);  Ende des Methodenaufrufs vom JButton
```

GUI mit Java - Editor

Ereignisbehandlung

Ereignisbehandlung

Die Ereignisbehandlung bei einem Button ist sehr einfach, da ein JButton nur ein mögliches Ereignis kennt:

Er kann angeklickt werden.

Wir brauchen daher nicht abzufragen, welches Ereignis der JButton ausgelöst hat, was wir mit Hilfe des übergebenen Parameters (ActionEvent evt) könnten.

In der aufgerufenen Methode der Gui-Klasse steht also einfach, was beim Anklicken geschehen soll.

Ereignisbehandlung

Schwieriger wird es beim am Anfang schon gezeigten WindowListener, der dem JFrame hinzugefügt wurde:

```
addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent evt)
          { System.exit(0); }
});
```

Ereignisbehandlung

Dem WindowListener wird ein WindowAdapter hinzugefügt.

In JAVA verwendet man Adapterklassen bei interfaces, von denen mehrere Methoden implementiert werden können, aber nicht notwendig müssen.

Das einzige Fensterereignis, das wir behandeln wollen, ist das Ereignis, dass der Benutzer das Fenster schließen will, also das windowClosing - Ereignis.

Ereignisbehandlung

Adapterklassen sind Klassen, nicht mehr interfaces.

Adapterklassen implementieren die möglichen Methoden durch leere Methodenrümpfe.

Damit sind prinzipiell alle Methoden definiert – auch wenn sie eben nichts tun – und die Implementation des interfaces ist zulässig.

So ist man nicht gezwungen, Methoden zu implementieren, die man gar nicht benötigt, sondern implementiert genau die benötigten neu.

Ereignisbehandlung

Wenn man Methoden der Adapterklassen in seiner anonymen inneren Klasse neu implementiert, überschreiben diese Definitionen die leeren von der AdapterKlasse vorgegebenen Methodenrümpfe.

Überschreiben (override) bedeutet, dass die neu geschriebenen Methoden die vorher definierten der vererbenden Klasse ersetzen.