Projekt „FragenTester“

Fragen Beantworten erstellt in Java

von Tobias Grönhagen

Inhaltsverzeichnis

Projekt “FragenTester”

Zusammenfassung

Das Folgende Projekt soll es ermöglichen Fragen zu verschiedenen Themenrichtungen zu testen.
Die Forderungen:

- Es soll drei Typen von Fragen geben:
  - ⇒ Fragen mit nur einer korrekten Antwort
  - ⇒ Fragen mit mehreren Korrekten Antworten (eine Frage ist erst dann korrekt beantwortet, wenn alle korrekten Antworten ausgewählt sind)
  - ⇒ Fragen mit textueller Antwort, das heißt der Geprüfte muss einen Text eingeben, der auf Richtigkeit geprüft wird. (Dazu sollen Reguläre Ausdrücke Verwendet werden)
- Fragen bekommen eine Punktzahl, Alle Punkte von korrekt beantwortetet Fragen werden zum Ende eines Testes aufsummiert und ergeben die Gesammtpunktzahl.
- Es soll zwei Modi für Tests geben. In einem soll es möglich sein, sich die Hilfe zu einer Frage anzeigen lassen zu können, im anderen soll das Unterbunden sein (reellere Testbedingungen)
- Es soll eine Zusammenfassung der beantworteten Fragen geben.
- Die Texte von Fragen und Antworten erlauben eine begrenzte Formatierung. Mit einem oder mehreren Leerzeichen eingerückte Passagen in den Quelltexten werden mit Schrift fester Breite und anderer Farbe dargestellt
- Laden von Fragen aus einer Datei soll möglich sein.

Das Projekt wurde von Tobias Grönhagen in „Eclipse SDK Version 3.4.0 erstellt.
Beim Öffnen der Hilfe zu einer Frage soll es möglich sein, Weiterhin Fragen zu beantworten. Bei geöffneten ConfiguartionsMenus und der Zasmmenfassung soll das nicht möglich ein.

Projekt “FragenTester”

Die Dateiformate

Einfaches Textuelles Format:

```
QUESTION 1:
Ist das eine Frage?
A.  ja
B.  nein
Answer: A

Points: 1

Explanation:
Das ist eine Unsinnige Ernährung für eine unsinnige Frage

QUESTION 2:
gib einen Text ein
Answer: .
Points: 5
Explanation:
Es wird auf einen Regulären Ausdruck geprüft
in diesem fall wird einfach geschaut ob ein Zeichen in der
Antwort ist

QUESTION 3:
Welche der Aussagen sind korrekt?
A. Bäume sind grün
B. Blumen sind bunt
B. und riechen gut
C. Steine wachsen
C. und trocken aus,
C. wenn man sie nicht gießt
Answer: A,B

Points: 3

Explanation:
Beispiel einer Frage, die mehrere Richtige Antworten hat.
```

Beginn eines Fragenblocks. Die Nummer ist die eindeutige ID der Frage

Der Fragentext

Antworten

korrekte Antwort(en). Komma separiert wenn mehrere Wahr sind.

Die mit der Frage erreichbaren Punkte

Erklärungen zu der Frage

Auch mehrzeilige Antworten sind möglich

Die erste Frage ist eine Frage mit nur einer richtigen Antwort

Bei der zweiten Frage muss der Nutzer einen Text eingeben.

Bei der Dritten sind mehrere Antworten korrekt.

XML Format:

```
<?xm version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE questions [
  <!ELEMENT questions   (question+)>
  <!ELEMENT question    (nr, text, answers, explanation? )>
  <!ELEMENT nr          (#PCDATA)>
  <!ELEMENT text        (#PCDATA)>
  <!ELEMENT answers     (answer+)>
  <!ELEMENT answer      (#PCDATA)>
  <!ELEMENT explanation (#PCDATA)>
  <!ATTLIST question points CDATA                       #REQUIRED>
  <!ATTLIST answer   status ( correct | wrong | regexp) #REQUIRED>
  <!ATTLIST answers  type   ( single | multi | string ) #IMPLIED>
]>
<questions>
  <question points="5">
    <nr>1</nr>
    <text><![CDATA[Ist das eine Frage?]]></text>
    <answers type="single">
      <answer status="correct"><![CDATA[ja]]></answer>
      <answer status="wrong"><![CDATA[nein]]></answer>
    </answers>
    <explanation><![CDATA[Das ist eine Unsinnige Ernährung für eine
unsinnige Frage]]></explanation>
  </question>
  <question points="5">
    <nr>2</nr>
    <text><![CDATA[gib Eigen Text ein]]></text>
    <answers type="string">
      <answer status="regexp"><![CDATA[.]]></answer>
    </answers>
    <explanation><![CDATA[Es wird auf einen Regulären Ausdruck geprüft.
in diesem Fall wird einfach geschaut, ob ein Zeichen in der Antwort
ist]]></explanation>
  </question>
  <question points="2">
    <nr>2</nr>
    <text><![CDATA[gib einen Text ein]]></text>
    <answers type="single">
      <answer status="correct"><![CDATA[Bäume sind grün]]></answer>
      <answer status="correct"><![CDATA[Blumen sind bunt
Und richen gut]]></answer>
      <answer status="wrong"><![CDATA[Steine wachsen
Und trocknen aus,
wenn man sie nicht gießt]]></answer>
    </answers>
    <explanation><![CDATA[Beispiel einer Frage, die mehrere richtige
Antworten hat.]]></explanation>
  </question>
</questions>
```

Die DTD

einleitender Tag mit den erreichbaren Punkten

Der Fragentext

Die möglichen Antworten, korrekte tragen den Wert „correct“

Die Erklärung

Die erste Frage ist eine Frage mit nur einer richtigen Antwort

Bei der zweiten Frage muss der Nutzer einen Text eingeben.

Bei der Dritten sind mehrere Antworten korrekt.

Projekt “FragenTester”

Verwendete Symbole

Die Bilder der Symbole die im Programm verwendet wurden stammen entweder aus dem Sortiment des GTK+ GUIToolkit und sind unter der GPL Lizenziert oder sind von mir Erstellt oder modifiziert worden.


Logo des FragenTesters


Einstellungen übernehmen und Fenster schließen


Einstellungen übernehmen


Aktion Abbrechen und Fenster schließen.
Keine Veränderung Daten


Ein Fehler ist Aufgetreten

Eine Information wird mitgeteilt


Datei öffnen


Datei Speichern


Programmeinstellungen verändern


Programm beenden


Datenbank laden


Datenbank konfigurieren


Anzeige der Zusammenfassung


Ein Übungstest beginnen


Einen Test ohne Hilfen beginnen


Eine Frage mit textueller Antwort


Eine Frage mit einzelner Auswahl


Frage mit Mehrfachauswahl


Eine nicht beantwortete Frage


Eine korrekt beantwortete Frage


Eine falsch beantwortete Frage


Eine beantwortete Frage sowie die Fragenauswahl


Die Aktuelle Frage


Eine Frage zurück gehen


Eine Frage weiter gehen

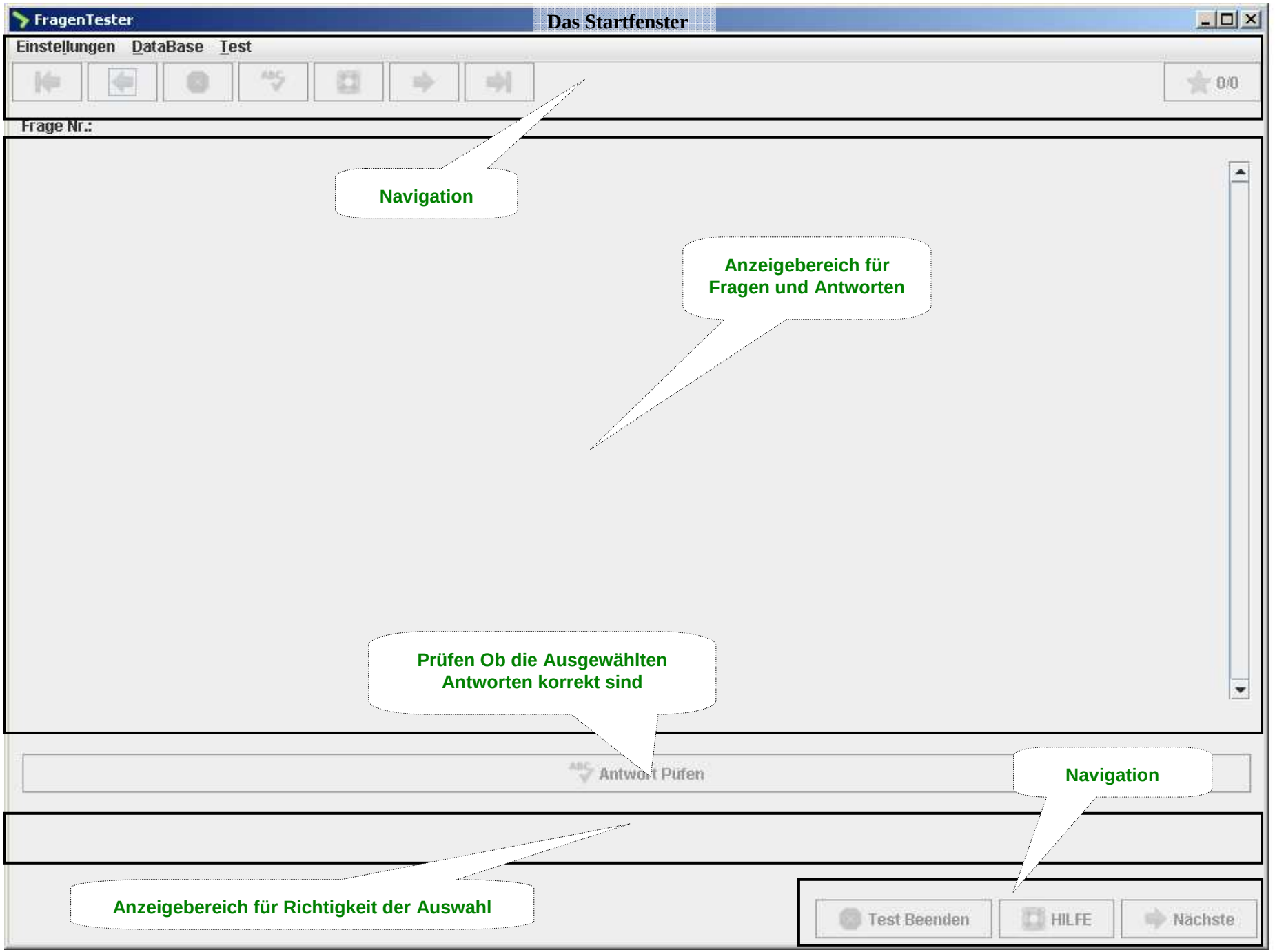

Zur Ersten Frage im Test springen


Zur Letzten Frage im Test springen


Hilfe zu einer Frage anzeigen


Frage auf Richtigkeit prüfen

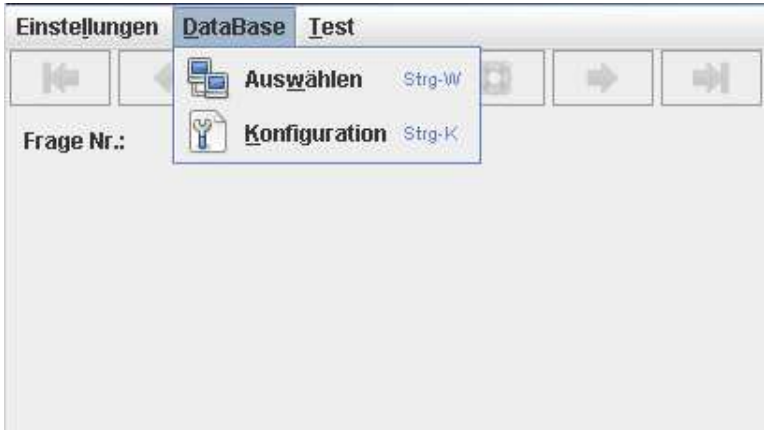
Das Startfenster



Das Menu

Der Menueintrag „Einstellungen“ dient zum Öffnen, Speichern und Bearbeiten der Configurationsdatei.Zu dem ist der Eintrag „Exit“ hier untergebracht und dient zum Beenden des Programms.

„Bearbeiten“ öffnet das Fenster zur Konfiguration der programmweiten Einstellungen.
Näheres zu Bearbeiten findet man unter „Programmkonfiguration“



Der Menueintrag „DataBase“ dient zum laden einer Datenbankdatei und Bearbeiten der Datenbankbezogenen Einstellungen.

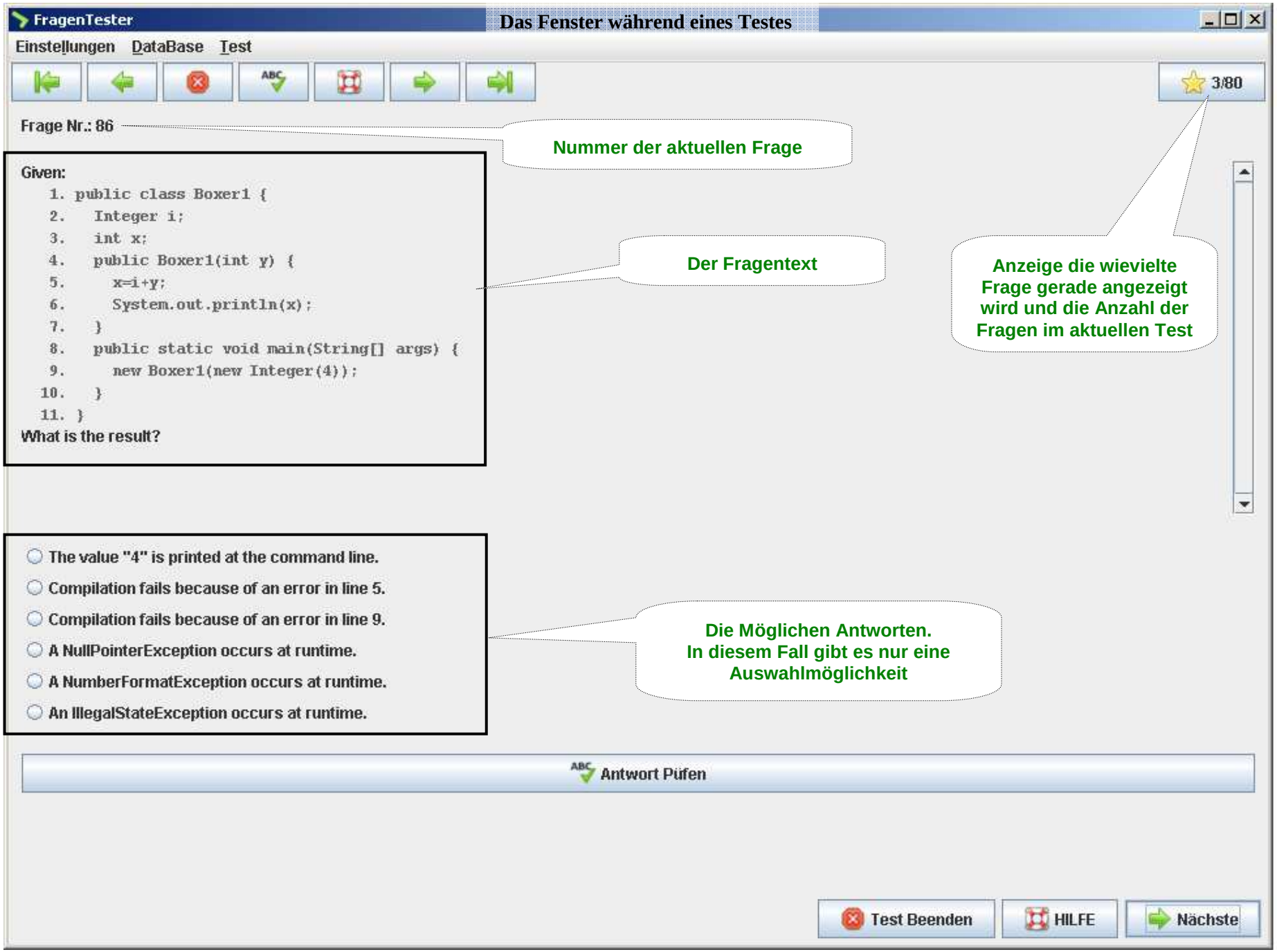
Mit „Auswählen kann man eine Datenbank laden.“

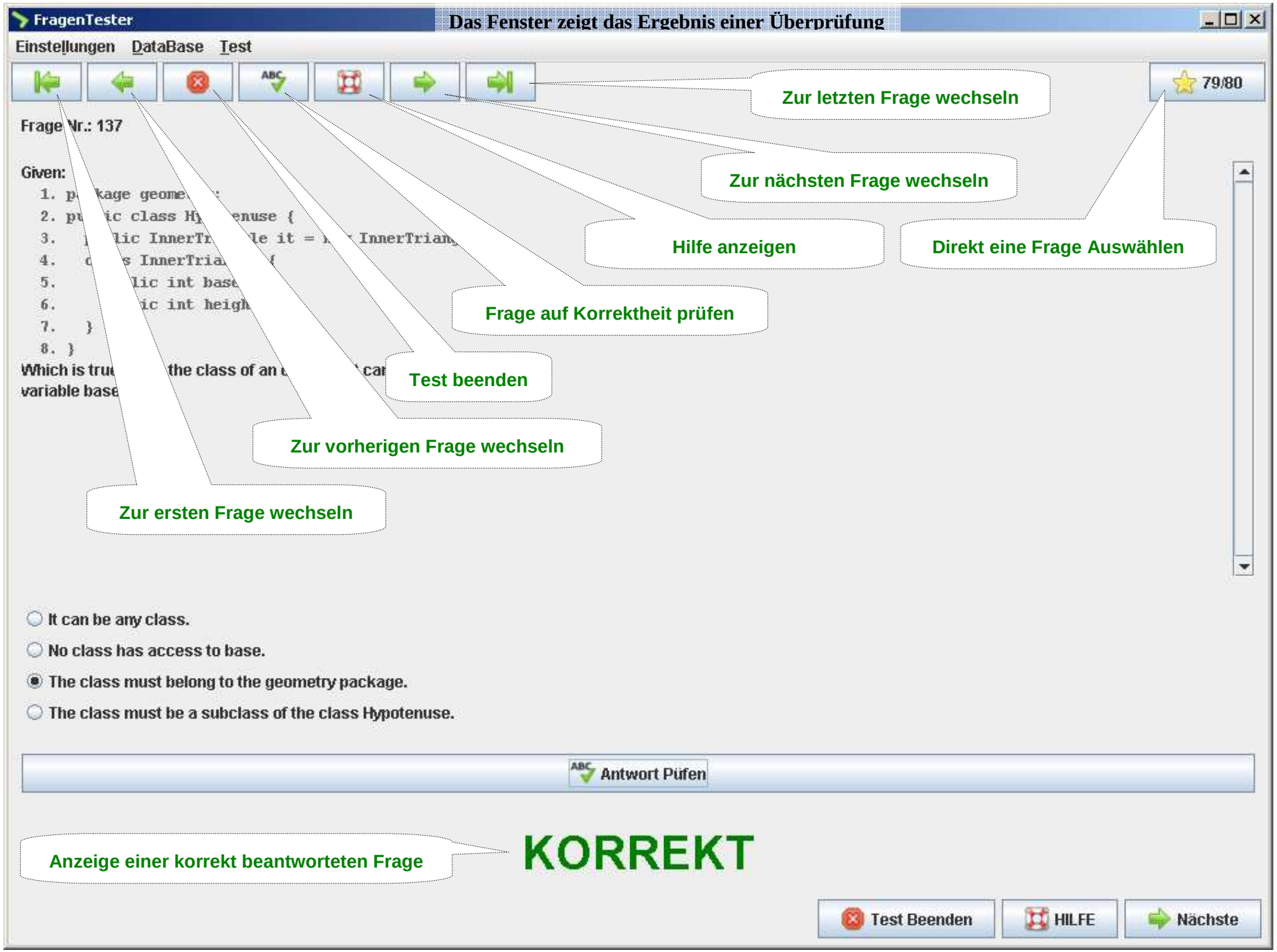
Mit „Konfiguration“ Öffnet man das Fenster zur Datenbankkonfiguration
Näheres dazu findet man unter „Datenbankkonfiguration“

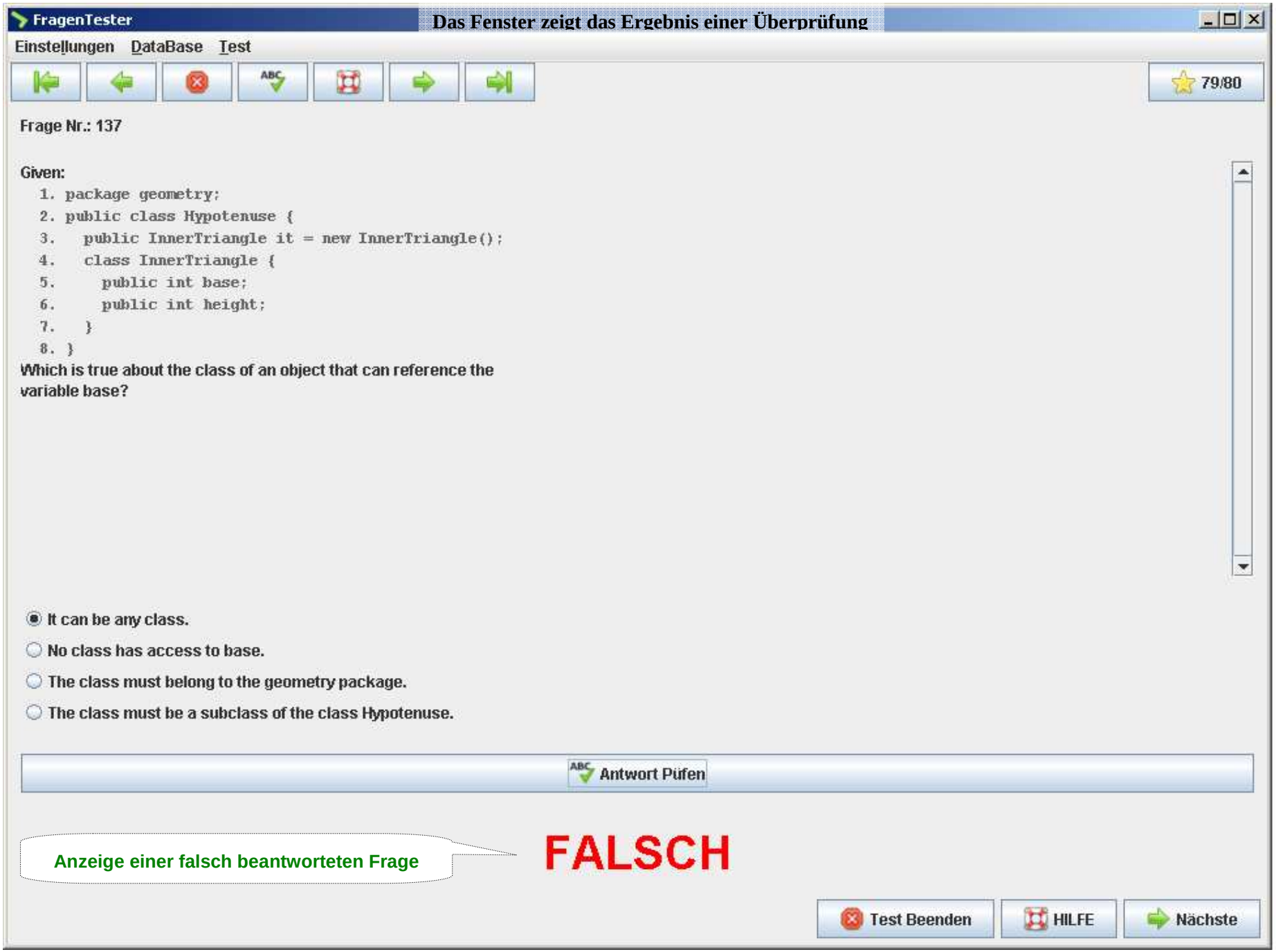
Der letzte Eintrag enthält alle testbezogenen Elemente.
Mittels „UEbung“ wird ein Übungstest gestartet, bei dem man die Hilfe nutzen und seine Antworten direkt Prüfen kann

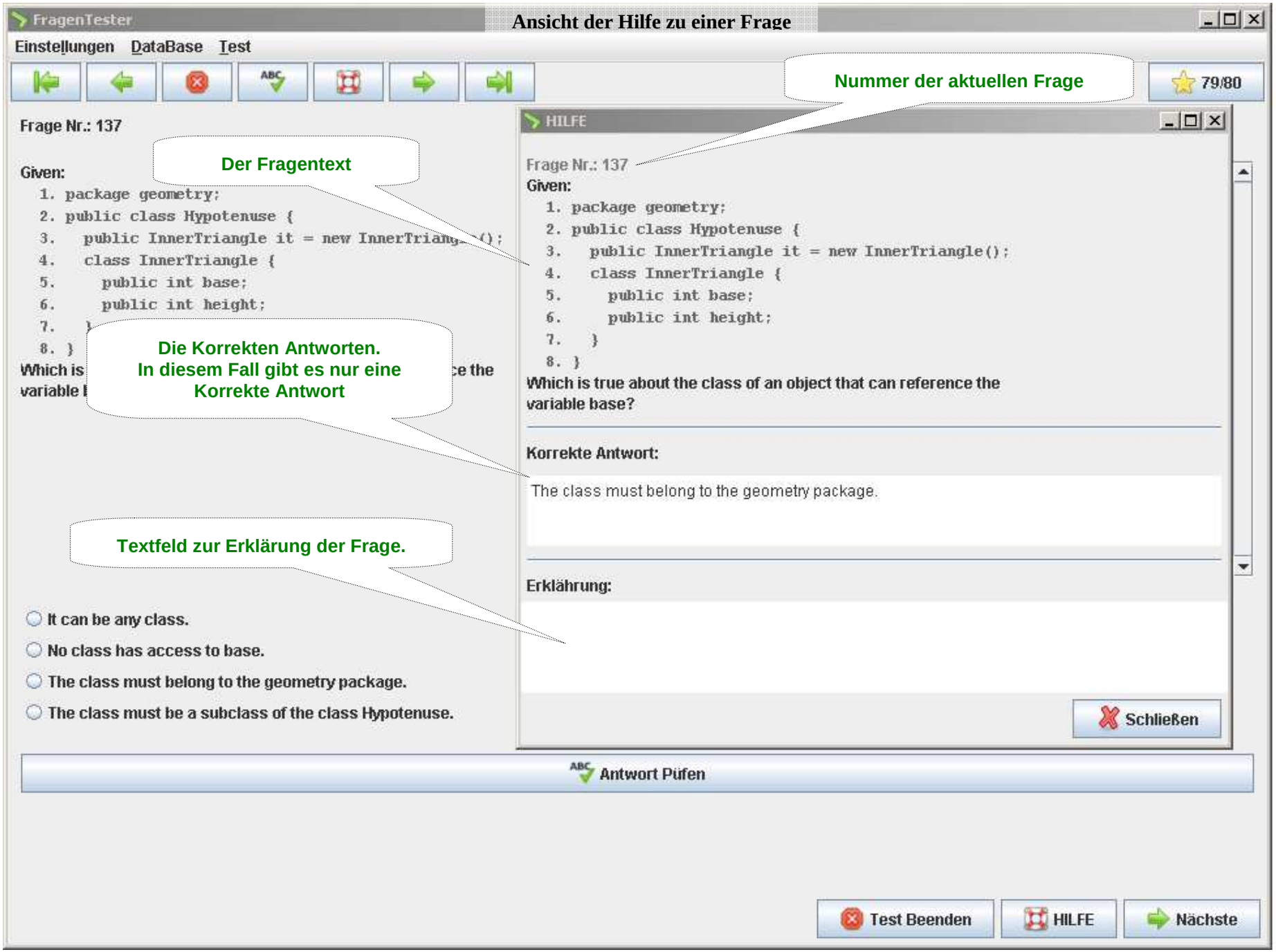
„Test Beenden“ und „Hilfe“ sind nur aktiv während eines Testes.

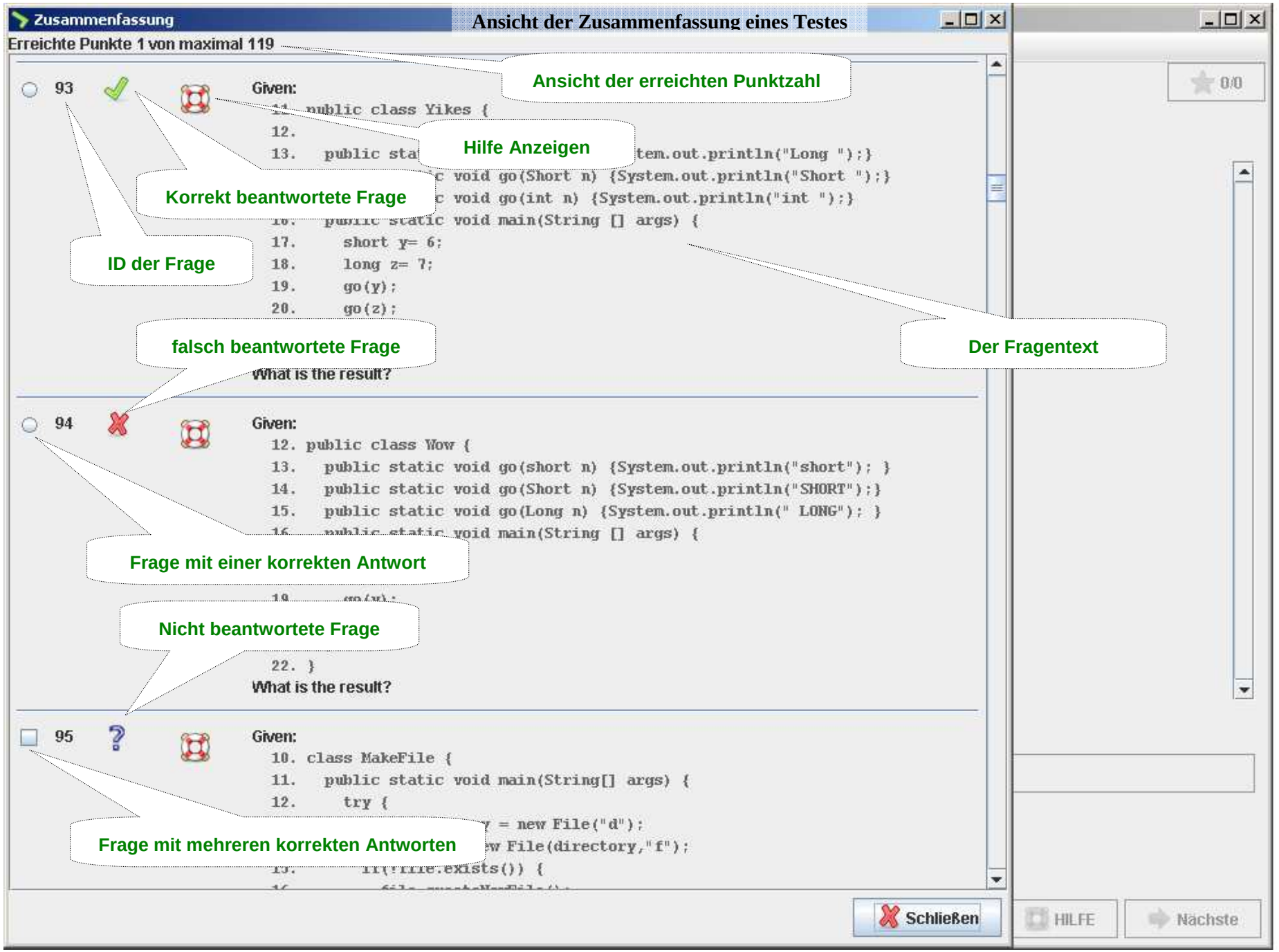
„Ergebnisse“ zeigt eine Auflistung der Leistungen beim letzten Test.
Näheres dazu findet man unter „Testzusammenfassung“

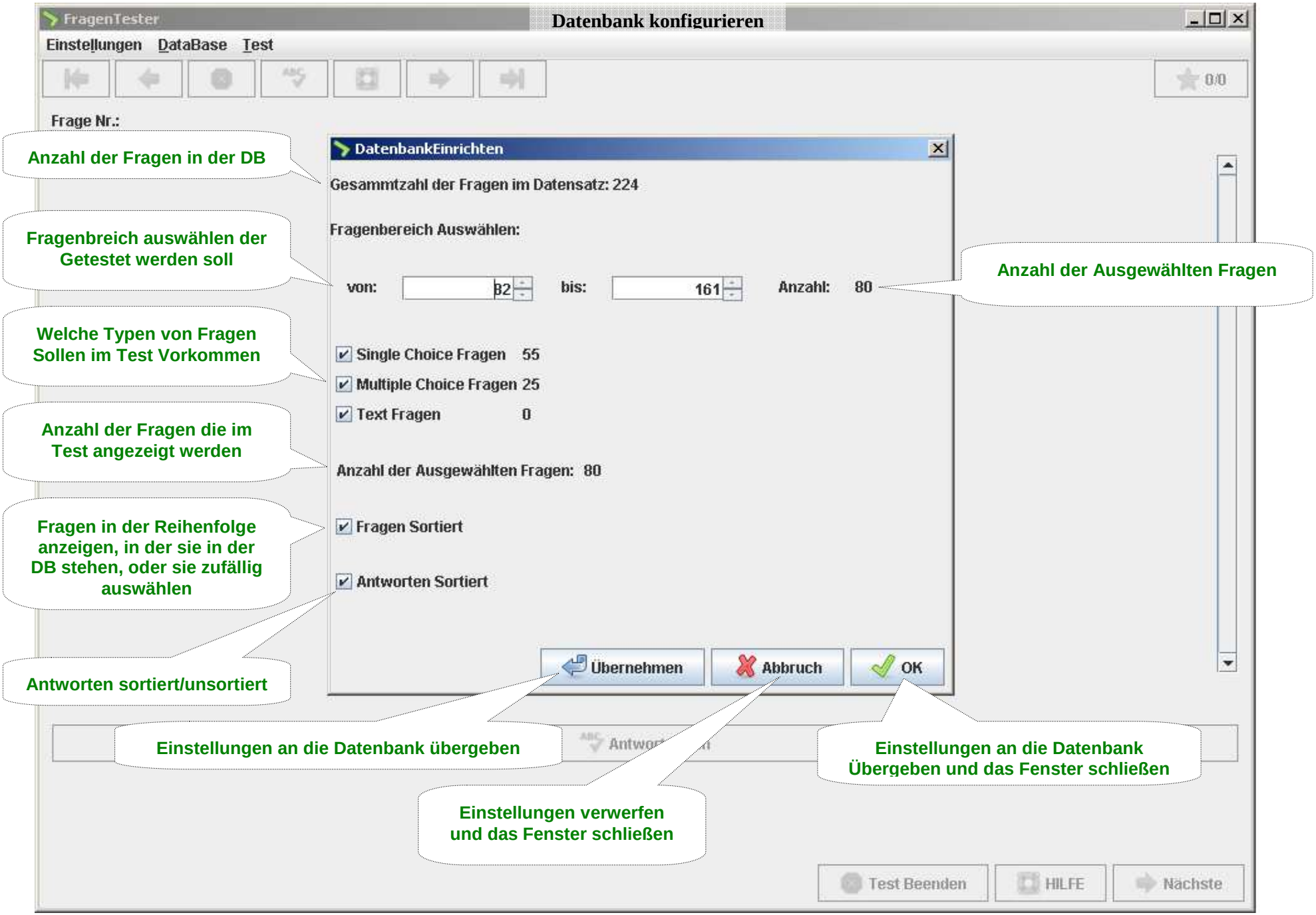

FragenTester
Das Fenster während eines Testes
Einstellungen DataBase Test
3/80
Frage Nr.: 86
Nummer der aktuellen Frage
Given:
1. public class Boxer1 {
2. Integer i;
3. int x;
4. public Boxer1(int y) {
5. x=i+y;
6. System.out.println(x);
7. }
8. public static void main(String[] args) {
9. new Boxer1(new Integer(4));
10. }
11. }
What is the result?
Der Fragentext
Anzeige die wievielte Frage gerade angezeigt wird und die Anzahl der Fragen im aktuellen Test
The value "4" is printed at the command line.
Compilation fails because of an error in line 5.
Compilation fails because of an error in line 9.
A NullPointerException occurs at runtime.
A NumberFormatException occurs at runtime.
An IllegalStateException occurs at runtime.
Die Möglichen Antworten.
In diesem Fall gibt es nur eine Auswahlmöglichkeit
Antwort Püfen
Test Beenden
HILFE
Nächste


FragenTester
Das Fenster zeigt das Ergebnis einer Überprüfung
Einstellungen DataBase Test
79/80
Frage Nr.: 137
Given:
Which is true
variable base
It can be any class.
No class has access to base.
The class must belong to the geometry package.
The class must be a subclass of the class Hypotenuse.
Antwort Püfen
KORREKT
Test Beenden
HILFE
Nächste
Zur letzten Frage wechseln
Zur nächsten Frage wechseln
Hilfe anzeigen
Direkt eine Frage Auswählen
Frage auf Korrektheit prüfen
Test beenden
Zur vorherigen Frage wechseln
Zur ersten Frage wechseln
Anzeige einer korrekt beantworteten Frage

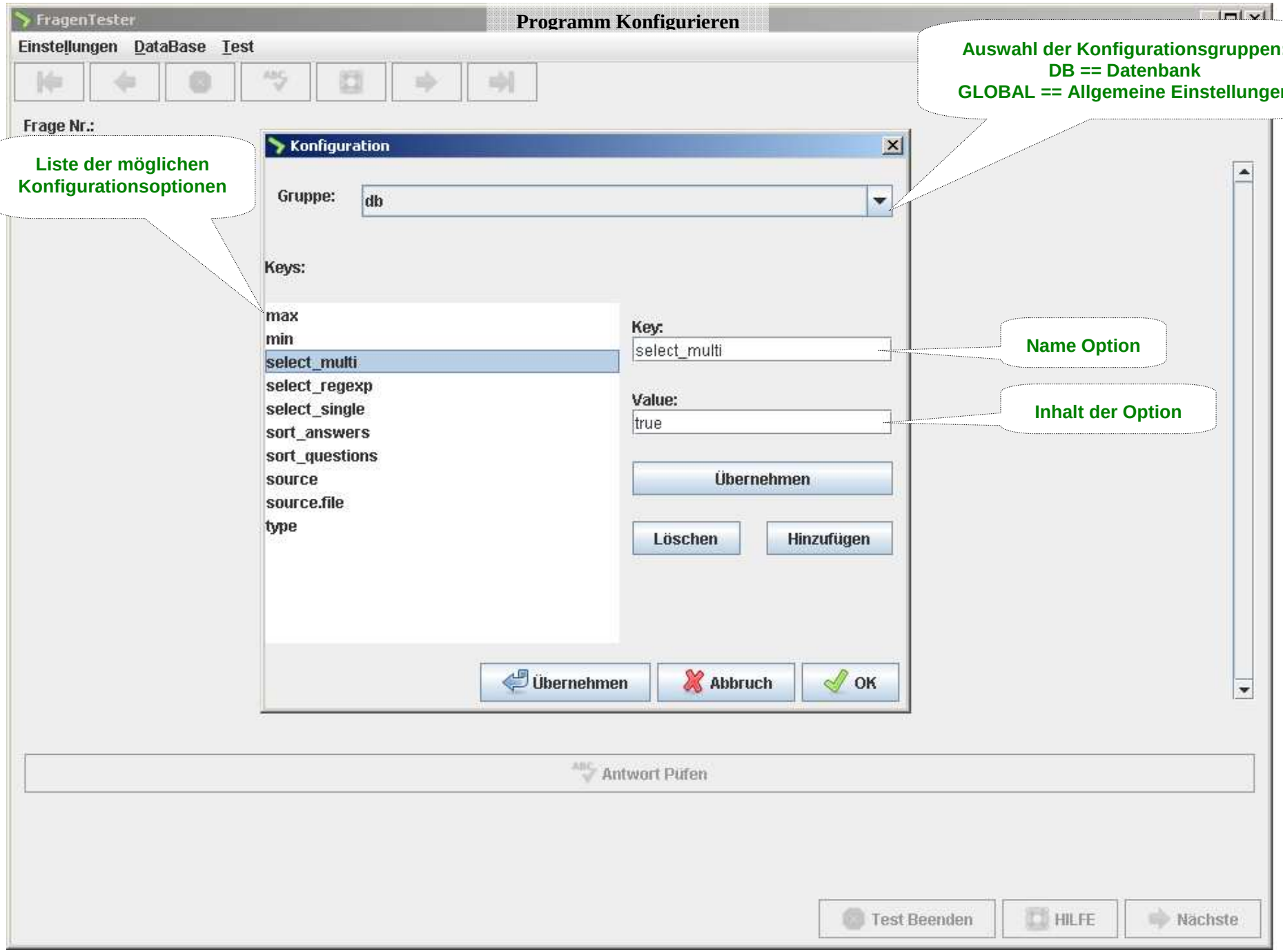

FragenTester
Das Fenster zeigt das Ergebnis einer Überprüfung
Einstellungen DataBase Test
79/80
Frage Nr.: 137
Given:
1. package geometry;
2. public class Hypotenuse {
3. public InnerTriangle it = new InnerTriangle();
4. class InnerTriangle {
5. public int base;
6. public int height;
7. }
8. }
Which is true about the class of an object that can reference the variable base?
It can be any class.
No class has access to base.
The class must belong to the geometry package.
The class must be a subclass of the class Hypotenuse.
Antwort Püfen
Anzeige einer falsch beantworteten Frage
FALSCH
Test Beenden
HILFE
Nächste


FragenTester
Ansicht der Hilfe zu einer Frage
Einstellungen DataBase Test
Nummer der aktuellen Frage
79/80
Frage Nr.: 137
Der Fragentext
Given:
1. package geometry;
2. public class Hypotenuse {
3. public InnerTriangle it = new InnerTriangle();
4. class InnerTriangle {
5. public int base;
6. public int height;
7. }
8. }
Which is true about the class of an object that can reference the variable base?
Die Korrekten Antworten.
In diesem Fall gibt es nur eine
Korrekte Antwort
Textfeld zur Erklärung der Frage.
It can be any class.
No class has access to base.
The class must belong to the geometry package.
The class must be a subclass of the class Hypotenuse.
HILFE
Frage Nr.: 137
Given:
1. package geometry;
2. public class Hypotenuse {
3. public InnerTriangle it = new InnerTriangle();
4. class InnerTriangle {
5. public int base;
6. public int height;
7. }
8. }
Which is true about the class of an object that can reference the variable base?
Korrekte Antwort:
The class must belong to the geometry package.
Erklährung:
Schließen
Antwort Püfen
Test Beenden
HILFE
Nächste

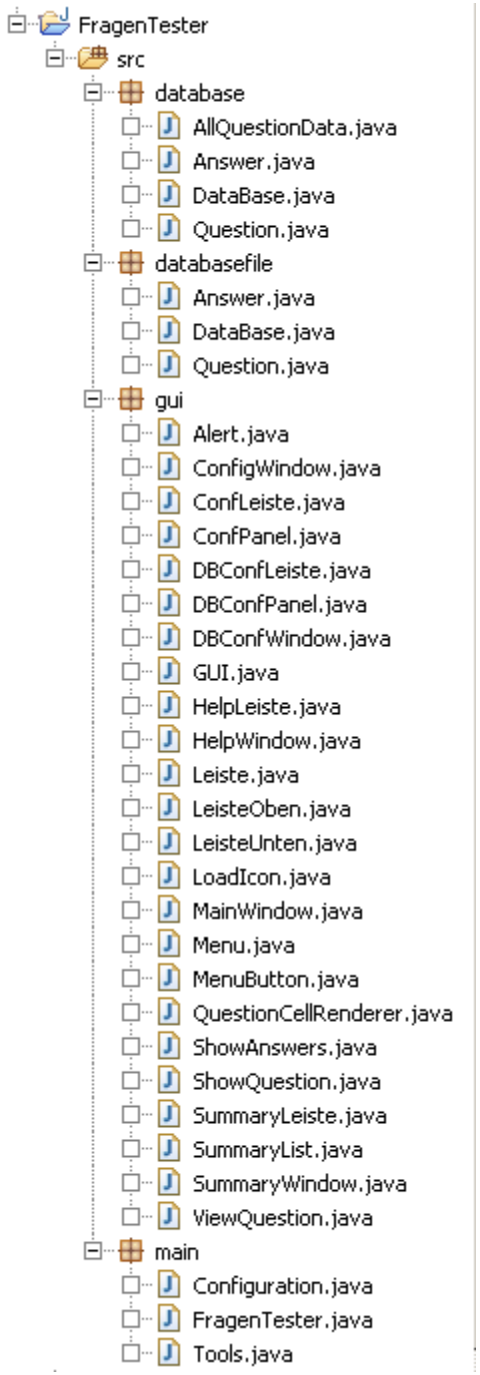

Zusammenfassung
Ansicht der Zusammenfassung eines Testes
Erreichte Punkte 1 von maximal 119
Ansicht der erreichten Punktzahl
93
Given:
public class Yikes {
Hilfe Anzeigen
Korrekt beantwortete Frage
ID der Frage
short y= 6;
long z= 7;
go(y);
go(z);
falsch beantwortete Frage
Der Fragentext
What is the result?
94
Given:
public class Wow {
public static void go(short n) {System.out.println("short"); }
public static void go(Short n) {System.out.println("SHORT");}
public static void go(Long n) {System.out.println(" LONG"); }
public static void main(String [] args) {
Frage mit einer korrekten Antwort
Nicht beantwortete Frage
}
What is the result?
95
Given:
class MakeFile {
public static void main(String[] args) {
try {
Frage mit mehreren korrekten Antworten
Schließen
HILFE
Nächste


FragenTester
Datenbank konfigurieren
Einstellungen DataBase Test
0/0
Frage Nr.:
DatenbankEinrichten
Gesammtzahl der Fragen im Datensatz: 224
Fragenbereich Auswählen:
von: 82
bis: 161
Anzahl: 80
Single Choice Fragen 55
Multiple Choice Fragen 25
Text Fragen 0
Anzahl der Ausgewählten Fragen: 80
Fragen Sortiert
Antworten Sortiert
Übernehmen
Abbruch
OK
Test Beenden
HILFE
Nächste
Anzahl der Fragen in der DB
Fragenbreich auswählen der Getestet werden soll
Welche Typen von Fragen Sollen im Test Vorkommen
Anzahl der Fragen die im Test angezeigt werden
Fragen in der Reihenfolge anzeigen, in der sie in der DB stehen, oder sie zufällig auswählen
Antworten sortiert/unsortiert
Anzahl der Ausgewählten Fragen
Einstellungen an die Datenbank übergeben
Einstellungen verwerfen und das Fenster schließen
Einstellungen an die Datenbank Übergeben und das Fenster schließen

Programm Konfigurieren



Der Programmcode



Das Programm unterteilt sich in vier Pakete:

- database
  Das Paket enthält drei Interfaceklassen und eine Hilfsklasse
  Interfaces: DataBase,Question,Answer
  Hilfsklasse: AllquestionData

- databasefile
  Das Paket ist eine Implementation der zuvor definierten Interfaces
  Es wird eine Datei basierte Datenbank implementiert, das ist zurzeit auch die einzige vollständige Implementation des „database“ Interface

- gui
  Enthält alle nötigen Objekte für die graphische Oberfläche
  Auf das MVC Konzept achtend ist hier ur der Code enthalten, der zur Erzeugung und Verwaltung der GUI nötig ist
- main
  Enthält die Klasse zur Verwaltung der Konfiguration und den zentralen Controller mit der main-Methode

Anmerkungen zur Implementierung.

Aus Zeitmangel wurde das angestrebte MVC Konzept nicht vollständig umgesetzt.
Mehrer Klassen bekommen Referenzen von Objekte aus den Klassen „Question“, „Answer“, „DataBase“ und „Configuration“ übergeben, die sie selber verwalten.
Ansonsten werden die meisten Benutzeraktionen Über Signale an die nächst höhere Instanz weitergereicht, oder selber verarbeitet, sofern die nötigen Informationen vorliegen. Dazu implementieren viele Klassen die Interfaces „ActionEvent“ und „ActionListener“ aus „java.awt.event“

Die Aktionsnamen

Aus databasefile.DataBase:
- DBQLAST wird von main. FragenTester verarbeitet
- DBQFIRST wird von main. FragenTester verarbeitet
- DBQMIDDLE wird von main. FragenTester verarbeitet

Aus main.Configuration:
- SAVECONFFAIL wird von main. FragenTester verarbeitet
- CONFLOADERROR wird von main. FragenTester verarbeitet
- CONFCHANGED wird von main. FragenTester verarbeitet

Aus gui.ConfLeiste:
- CWACEPT wird von gui. ConfWindow verarbeitet
- CWCLOSE wird von gui. ConfWindow verarbeitet
- CWOK wird von gui. ConfWindow verarbeitet

Aus gui.ConfPanel:
- UPDATEVALUE wird von gui. ConfWindow verarbeitet

Aus gui.DBConfLeiste:
- DBCWACEPT wird von gui. DBConfWindow verarbeitet
- DBCWCLOSE wird von gui. DBConfWindow verarbeitet
- DBCWOK wird von gui. DBConfWindow verarbeitet

Aus gui.HelpLeiste:
- HWCLOSE wird von gui. HelpWindow verarbeitet

Aus gui.LeisteOben:
- QFIRST wird von main. FragenTester verarbeitet
- QBEVORE wird von main. FragenTester verarbeitet
- TEND wird von main. FragenTester verarbeitet
- QTEST wird von main. FragenTester verarbeitet
- THELP wird von main. FragenTester verarbeitet
- QNEXT wird von main. FragenTester verarbeitet
- QLAST wird von main. FragenTester verarbeitet

Aus gui.LeisteUnten:
- TEND wird von main. FragenTester verarbeitet
- THELP wird von main. FragenTester verarbeitet
- QNEXT wird von main. FragenTester verarbeitet

Aus gui.Menu:
- LOAD wird von main. FragenTester verarbeitet
- SAVE wird von main. FragenTester verarbeitet
- CONF wird von main. FragenTester verarbeitet
- EXIT wird von main. FragenTester verarbeitet
- DBSEL wird von main. FragenTester verarbeitet
- DBCONF wird von main. FragenTester verarbeitet
- TPROBE wird von main. FragenTester verarbeitet
- TSTART wird von main. FragenTester verarbeitet
- TEND wird von main. FragenTester verarbeitet
- THELP wird von main. FragenTester verarbeitet
- TSHOW wird von main. FragenTester verarbeitet

Aus gui.MainWindow
- EXIT wird von main. FragenTester verarbeitet

Aus gui.QuestionCellRenderer
- QJUMP wird von main. FragenTester verarbeitet

Aus gui.SummaryLeiste
- SWCLOSE wird von gui. SummaryWindow verarbeitet

Projekt “FragenTester”

Übersicht der KlassenHierarchie innerhalb des Paketes „gui“

- java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
  - java.awt.Container
    - javax.swing.JComponent (implements java.io.Serializable)
      - javax.swing.AbstractButton (implements java.awt.ItemSelectable, javax.swing.SwingConstants)
        - javax.swing.JButton (implements javax.accessibility.Accessible)
          - gui.MenuButton (implements java.awt.event.ActionListener)
      - javax.swing.JLabel (implements javax.accessibility.Accessible, javax.swing.SwingConstants)
        - gui.QuestionCellRenderer (implements javax.swing.ListCellRenderer)
        - gui.ShowQuestion
      - javax.swing.JMenuBar (implements javax.accessibility.Accessible, javax.swing.MenuElement)
        - gui.Menu
      - javax.swing.JPanel (implements javax.accessibility.Accessible)
        - gui.ConfPanel (implements java.awt.event.ActionListener, javax.swing.event.ListSelectionListener)
        - gui.DBConfPanel (implements java.awt.event.ActionListener)
        - gui.Leiste (implements java.awt.event.ActionListener)
          - gui.ConfLeiste
          - gui.DBConfLeiste
          - gui.HelpLeiste
          - gui.LeisteOben
          - gui.LeisteUnten
          - gui.SummaryLeiste
        - gui.ShowAnswers (implements java.awt.event.ActionListener, javax.swing.event.CaretListener, java.awt.event.InputMethodListener)
        - gui.SummaryList
        - gui.ViewQuestion
    - java.awt.Window (implements javax.accessibility.Accessible)
      - java.awt.Dialog
        - javax.swing.JDialog (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
          - gui.Alert
          - gui.ConfigWindow
          - gui.DBConfWindow
      - java.awt.Frame (implements java.awt.MenuContainer)
        - javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
          - gui.HelpWindow
          - gui.MainWindow
          - gui.SummaryWindow
- gui.GUI (implements java.awt.event.ActionListener)
- javax.swing.ImageIcon (implements javax.accessibility.Accessible, javax.swing.Icon, java.io.Serializable)
  - gui.LoadIcon

Projekt “FragenTester”

Übersicht der KlassenHirarchie innerhalb des Paketes „databasefile“

- databasefile.Answer (implements database.Answer)
- databasefile.DataBase (implements database.DataBase)
- databasefile.Question (implements database.Question)

Übersicht der KlassenHirarchie innerhalb des Paketes „main“

- main.Configuration
- main.FragenTester (implements java.awt.event.ActionListener)
- main.Tools

Inhaltsverzeichnis der Klassen

Projekt: FragenTester
Paket: main
Klasse : FragenTester.java

FragenTester.java

```
package main;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import gui.GUI;
import database.DataBase;
import database.Question;

/**
 * @author ToPeG
 * Mainclass and Controller
 */
public class FragenTester implements ActionListener{

  private DataBase db;
  private GUI gui = new GUI();
  private Configuration conf=new Configuration();

  public static void main(String[] args)
  {
    new FragenTester();
  }

  private FragenTester()
  {
    gui.addActionListener(this);
    conf.addActionListener(this);
    //Load Configurations.
    conf.load();
    try {
      //Load Database
      DataBase neu=conf.loadDB();
      if(neu != null)
      {
        //integrate Database
        db=neu;
        conf.syncDB(db);
        db.addActionListener(this);
      }
    }
    catch (Exception e)
    {
      gui.alert("ERROR LOAD DATABASE","<html>Error while open file: "+conf.get("db","source")+"<br> <br><pre>"+
                e.getLocalizedMessage()+"</pre></html>","dialog-error");
    }
  }
```

```
  private void exit(){ System.exit(0); }

  public void actionPerformed(ActionEvent e)
  {
    String txt=e.getActionCommand();
    Object o=e.getSource();

    //Exit on signal
    if(txt.equals("EXIT")) exit();
    //End Test
    else if(txt.equals("TEND"))
    { gui.endTest(); }
    //ÜbungsTest
    else if(txt.equals("TPROBE"))
    {
      db.start();
      gui.setQuestion(db.getQuestion());
      gui.startProbeTest(db.countAllPossibleQuestions());
      gui.showQuestion();
      gui.disablePrevious(true);
      gui.hideSummary();
    }
    //echter Test (keine Hilfen)
    else if(txt.equals("TSTART"))
    {
      db.start();
      gui.setQuestion(db.getQuestion());
      gui.startRealTest(db.countAllPossibleQuestions());
      gui.showQuestion();
      gui.disablePrevious(true);
      gui.hideSummary();
    }
    //is answer correct?
    else if(txt.equals("QTEST"))
    {
      gui.testCorrect();
    }
    //show help to the question
    else if(txt.equals("QINFO"))
    {
      gui.showQuestionInfos(db.getAllPossibleQuestions());
    }
    //show next Question
    else if(txt.equals("QNEXT"))
    {
      db.next();
      gui.setQuestionInfoText(db.countPossibleQuestion(),db.countAllPossibleQuestions());
      updateQuestion();
    }
    //show previous question
    else if(txt.equals("QBEVORE"))
    {
      db.previous();
      gui.setQuestionInfoText(db.countPossibleQuestion(),db.countAllPossibleQuestions());
      updateQuestion();
```

```
    }
    //show last Question
    else if(txt.equals("QLAST"))
    {
      db.last();
      gui.setQuestionInfoText(db.countPossibleQuestion(),db.countAllPossibleQuestions());
      updateQuestion();
    }
    //show first Question
    else if(txt.equals("QFIRST"))
    {
      db.first();
      gui.setQuestionInfoText(db.countPossibleQuestion(),db.countAllPossibleQuestions());
      updateQuestion();
    }
    //jump to selected question
    else if(txt.equals("QJUMP"))
    {
      Question q=(Question)o;
      if(q!=null)
      {
        int pos=db.getQuestionPosition(q.getId());
        if(db.isPossibleQuestion(pos))
        {
          db.setPosition(pos);
          gui.setQuestionInfoText(db.countPossibleQuestion(),db.countAllPossibleQuestions());
          updateQuestion();
        }
      }
    }
    //is the last Question
    else if(txt.equals("DBQLAST"))
    {
      gui.disableNext(true);
      gui.disablePrevious(false);
    }
    //is the first Question
    else if(txt.equals("DBQFIRST"))
    {
      gui.disablePrevious(true);
      gui.disableNext(false);
    }
    // is not the last or the first question
    else if(txt.equals("DBQMIDDLE"))
    {
      gui.disablePrevious(false);
      gui.disableNext(false);
    }
    //Show summary
    else if(txt.equals("SSHOW"))
    {
      gui.setSummaryDataBase(db);
      gui.showSummary();
    }
    //show Database configuration
    else if(txt.equals("DBCONF"))
```

```
    {
      gui.DBConf(db);
      conf.syncConf(db);
    }
    //load a Database
    else if(txt.equals("DBSEL"))
    {
      loadFileDB(gui.openFile());
    }
    //Save Configuration
    else if(txt.equals("SAVE"))
    {
      saveAsConf(gui.saveAsFile());
    }
    //load Configuartion
    else if(txt.equals("LOAD"))
    {
      loadConf(gui.openFile());
    }
    //show Application configuration
    else if(txt.equals("CONF"))
    {
      gui.showConf(conf);
      conf.syncDB(db);
    }
    //Configuration changed
    else if(txt.equals("CONFCHANGED"))
    {
      //gui.alert("CHANGE","Konfiguartion hat sich geändert","dialog-info");
    }
    //Coul not load Configfile
    else if(txt.equals("CONFLOADERROR"))
    {
      IOException ee=(IOException)o;
      gui.alert("ERROR LOAD CONF","<html>Fehler beim Laden der Konfiguration<br> <br><pre>"+ee.getLocalizedMessage()+
                "</pre><br>Es steht nur die Defaultkonfiguration zur Verfügung</html>","dialog-error");
    }
    // unknow message
    else
    { gui.alert("MESSAGE",txt+" ("+o+")","dialog-info"); }
  }

  private void updateQuestion()
  {
    gui.setQuestion(db.getQuestion());
    gui.showQuestion();
  }

  private void loadConf(File file)
  { conf.load(""+file); }

  //private void saveConf(){ conf.save(); }

  private void saveAsConf(File file)
  { conf.save(""+file); }
```

```
  public void loadFileDB(File file)
  {
    //System.out.println(file.getName());
    DataBase dbneu=new databasefile.DataBase();
    if(file != null) try{
      HashMap<String,String> map = new HashMap<String,String>();
      map.put("file",""+file);
      dbneu.setSource(map);
    }
    catch(Exception e)
    {
      gui.alert("ERROR LOAD DATABASE","<html>Error while open file: "+file+"<br> <br><pre>"+e.getLocalizedMessage()+"</pre></html>","dialog-error");
      dbneu=null;
      //exit();
    }
    if(dbneu != null)
    {
      if(db!=null)db.removeActionListener(this);
      db=dbneu;
      db.addActionListener(this);
      db.setMax(db.size()-1);
      db.sortQuestions();
      db.sortAnswers();
      conf.syncConf(db);
    }
    else
      gui.alert("ERROR LOAD DATABASE","<html>Databse not changed.</html>","dialog-info");
  }
}
```

Projekt: FragenTester
Paket: main
Klasse : Tools.java

Tools.java

```
package main;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

/**
 *
 * @author ToPeG
 * HelperClass
 *
 */
public abstract class Tools {

  /**
   *
   * @param txt QuestionText
   * @return pseudoHTML formated String of the QuestionText
   */
  public static String preformatQuestion(String txt)
  {
    String out="<html><p>";
    String[] lst=txt.split("[\n\r]+");
    boolean pre=false;
    for(String line:lst)
    {
      line=line.replaceAll("&","&amp;");
      line=line.replaceAll("<","&lt;");
      line=line.replaceAll(">","&gt;");
      if(!pre)
      {
        if(line.startsWith(" "))
        {
          pre=true;
          line=line.replaceAll("\\s"," ");
          out+="</p><font color=#666666 face=COURIER> "+line+"<br>\n";
        }
        else
        {
          out+=line+"<br>";
        }
      }
      else
      {
        if(!line.startsWith(" "))
        {
```

```
          pre=false;
          out+="</font><p>"+line+"<br>";
        }
        else
        {
          line=line.replaceAll("\\s"," ");
          out+=" "+line+"<br>\n";
         }
      }
    }
    txt.replaceAll("\n","<br>");
    txt="<html><body><p>"+txt+"</p></body></html>";
    return out+"</p></html>";
  }

  public static URL getPath(String path)
  {
    URL url = Tools.class.getResource("/"+path);
    if (url == null) try
      { url = new File(path).toURI().toURL(); }
      catch (MalformedURLException e)
      { url=null; }
    return url;
  }

  public static String slurpFile(String file) throws IOException
  {
    String data="";
    //System.out.println("FILE:"+file);
    File dat=null;
    InputStreamReader in=null;
    if(file!= null && !file.equals(""))
    {
      URL url =getPath(file);
      System.out.println(url.getProtocol());
      if( url.getProtocol().equals("jar") )
      {
        //System.out.println("ARCHIV");
        //in = new InputStreamReader(url.openStream(),"UTF8");

        in=new InputStreamReader(Tools.class.getResourceAsStream("/"+file));
      }
      else
      {
        dat =new File(file);
        //System.out.println("PLAIN");
        in = new InputStreamReader(new FileInputStream(dat.getPath()),"UTF8");
      }

      if(in!=null)
      {
        System.out.println("lese Datei ... ");
        if(dat != null)
        {
          char[] dt = new char[(int)dat.length()];
          int ok=in.read(dt,0,(int)dat.length());
```

```
          //System.out.println("SIZE="+ok);
          data=String.copyValueOf(dt,0,ok);
          in.close();
        }
        else
        {
          while(in.ready())
          {
            char[] dt=new char[10000];
            int ok=in.read(dt,0,(int)dt.length);
            //System.out.println("SIZE="+ok);
            data+=String.copyValueOf(dt,0,ok);
          }
          in.close();
        }
        //System.out.println("\nabgeschlossen.");
      }
      else throw new IOException("Datei konnte nicht geöffnet werden");
    }

    if(data.equals("")) throw new IOException("Datei enthielt keine Daten!");
    return data;
  }
}
```

Projekt: FragenTester
Paket: main
Klasse : Configuartion.java

Configuration.java

```java
package main;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import database.DataBase;

/**
 * @author ToPeG
 * handle Configuartion
 */
public class Configuration {

  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();
  private HashMap<String,HashMap<String,String>> conf = new HashMap<String,HashMap<String,String>>();

  //#####################################################
  public Configuration(){ reset(); }
  public Configuration(String f)
  { load(f); }
  //#####################################################

  //#####################################################
  public String get(String group,String key)
  {
    if(conf.get(group) == null ) return null;
    return conf.get(group).get(key);
  }

  public String[] getGroups()
  { return conf.keySet().toArray(new String[conf.keySet().size()]); }

  public String[] getKeys(String group)
  { return conf.get(group).keySet().toArray(new String[conf.get(group).keySet().size()]); }

  public int getAsInt(String g,String k)
  {
    if(get(g,k) != null) return Integer.parseInt(get(g,k));
    return 0;
  }
  public boolean getAsBoolean(String g,String k)
  {
    if(get(g,k)!= null) return Boolean.parseBoolean(get(g,k));
    return false;
  }
```

```
  public File getAsFile(String g,String k)
  {
   if(get(g,k) != null) return new File(get(g,k));
   return null;
  }
  public String getAsString(String g,String k) { return get(g,k); }
  public double getAsDouble(String g,String k)
  {
    if(get(g,k)!= null) return Double.parseDouble(get(g,k));
    return 0;
  }

  public String[] getAsStringArray(String g,String k)
  {
    if(get(g,k)!= null) return get(g,k).split("\\s*,\\s*");
    return null;
  }

  public void set(String group,String key,String value)
  {
    if(conf.get(group)==null) conf.put(group, new HashMap<String,String>());
    conf.get(group).put(key,value);
    doAction("CONFCHANGED",this);
  }

  public void remove(String group,String key)
  {
    if(conf.get(group)!=null)
    {
      if(conf.get(group).get(key) != null)
      {
        conf.get(group).remove(key);
      }
    }
  }
  //#####################################################

  //#####################################################
  private void reset()
  {
    conf.clear();
    set("conf","file","data/FragenTester.conf");
    set("global","sort","db,test");
    set("test","timout","3600");
    set("db","min","1");
    set("db","max","80");
    set("db","type","file"); // file,sql,net
    set("db","source","data/DefaultFragen.db"); // quelle...
    set("db","sort_questions","false");
    set("db","sort_answers","false");
    set("db","select_single","true");
    set("db","select_multi","true");
    set("db","select_regexp","true");
  }

  public void load(){ load(get("conf","file")); }
```

```
  public void load(String f)
  {
    reset();
    set("conf","file",f);
    try {
      parse(Tools.slurpFile(getAsString("conf","file")));
    }
    catch (IOException e)
    { doAction("CONFLOADERROR",e); }
    doAction("CONFCHANGED",this);
  }

  public void save(String f){set("conf","file",f);  save(); }
  public void save(){ if(get("conf","file") != null) writeFile(getAsFile("conf","file"),create()); }
  //#######################################################

  //#######################################################
  private void parse(String s)
  {
    //System.out.println("DAT:  "+s);
    String[] list= s.split("[\\r\\n]+");
    String group="global";
    for(String l: list)
    {
      if(!l.startsWith("#"))
      {
        if(l.matches(".+=.+"))
        {
          while(l.matches("\\A\\s+.+")) l=l.substring(1);
          l=l.split("#",2)[0];
          String key=l.split("=",2)[0];
          String value=l.split("=",2)[1];
          set(group,key,value);
        }
        else if(l.matches("^\\s*\\[\\s*\\w+\\s*\\]\\s*$"))
        { group=l.split("\\[\\s*",2)[1].split("\\s*\\]",2)[0]; }
      }
    }
  }

  public String create()
  {
    String out="";
    for(String group: conf.keySet())
    {
      out+="["+group+"]\n";
      for(String key: conf.get(group).keySet()){ out+="  "+key+"="+get(group,key)+"\n"; }
    }
    return out;
  }
  //#######################################################
```

```
  private void writeFile(File f, String s)
  {
    if(f!= null && s!=null)
    {
      try{
        FileWriter fw=new FileWriter(f);
        fw.write(s,0,s.length());
      }catch(IOException e)
      {
        doAction("SAVECONFFAIL",f);
      }
    }
  }
  //#####################################################

  public DataBase loadDB() throws Exception
  {
    DataBase db=null;

    if(get("db","type").equals("file"))
    {
      if(get("db","source") != null && !get("db","source").equals(""))
      {
        HashMap<String,String> map= new HashMap<String,String>();
        map.put("file",get("db","source"));
        if(get("db","source.type") != null && !get("db","source.type").equals("")) map.put("type",get("db","source.type"));
        db=new databasefile.DataBase();
        db.setSource(map);
      }
      else
        throw new Exception("no file set!");
    }
    else
      throw new Exception("not Supported yet");
    return db;
  }

  public void syncConf(DataBase db)
  {
    if(db != null)
    {
      reset();
      set("db","min",""+db.getMin());
      set("db","max",""+db.getMax());
      set("db","type",db.getType());

      Map<String,?> map =db.getSource();
      for(String key: map.keySet()){ set("db","source."+key,map.get(key).toString()); }

      set("db","sort_questions",db.isSortedQuestions()?"true":"false");
      set("db","sort_answers",db.isSortedAnswers()?"true":"false");
      set("db","select_single",db.isSingleExcuded()?"false":"true");
      set("db","select_multi",db.isMultiExcuded()?"false":"true");
      set("db","select_regexp",db.isRegexpExcuded()?"false":"true");
    }
  }
```

```
  public void syncConf(Configuration cnf)
  {
    if(cnf != null)
    {
      for(String group: cnf.conf.keySet())
      {
        for(String key: cnf.conf.get(group).keySet())
        { set(group,key,cnf.get(group,key)); }
      }
    }
  }

  public void syncDB(DataBase db)
  {
    if(db != null)
    {
      db.setMin(getAsInt("db","min"));
      db.setMax(getAsInt("db","max"));
      db.excludeSingle(!getAsBoolean("db","select_single"));
      db.excludeMulti(!getAsBoolean("db","select_multi"));
      db.excludeRegexp(!getAsBoolean("db","select_regexp"));
      if(getAsBoolean("db","sort_questions")) db.sortQuestions();
      else db.randomQuestions();
      if(getAsBoolean("db","sort_answers")) db.sortAnswers();
      else db.randomAnswers();
    }
  }

  public Configuration clone()
  {
    Configuration neu = new Configuration();
    for(String group: conf.keySet())
    {
      for(String key: conf.get(group).keySet())
      { neu.set(group,key,get(group,key)); }
    }
    return neu;
  }

  public String toString()
  {
    String out="";
    for(String group: conf.keySet())
    {
      for(String key: conf.get(group).keySet())
      { out+=group+"."+key+"="+get(group,key)+"\n"; }
    }
    return out;
  }
  //#####################################################
  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : Alert.java

Alert.java

```
package gui;

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

/**
 * @author ToPeG
 * Show an Alertbox
 */
public class Alert extends JDialog{
  private static final long serialVersionUID = -2450874651785820 36L;

  public Alert(Frame owner, String title, String text){this(owner,title,text,null);}
  public Alert(Frame owner, String title, String text, ImageIcon icon)
  {
    super(owner,title,true);
    setLayout(new BorderLayout());
    setSize(380,180);
    //setResizable(false);
    setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

    if(icon != null)
    {
      JLabel licon = new JLabel(text,icon,0);
      licon.setVerticalAlignment(0);
      add(licon,BorderLayout.CENTER);
      //setSize(licon.getWidth(),180);
    }

    {
      JPanel panel = new JPanel();
      panel.setLayout(new FlowLayout(FlowLayout.CENTER));
      JButton close = new JButton("OK",LoadIcon.load("apply"));
      close.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){ close(); }});
      panel.add(close);
      add(panel, BorderLayout.SOUTH);
    }

    setVisible(true);

  }

  private void close()
  {
    setVisible(false);
    dispose();
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : Leiste.java

Leiste.java

```
package gui;

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.HashMap;

import javax.swing.*;

/**
 *
 * @author ToPeG
 * Abstract class for generating ButtonPannels
 *
 */
public abstract class Leiste extends JPanel implements ActionListener{
  private static final long serialVersionUID = -9206744047639322439L;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();
  private HashMap<String,Component> list = new HashMap<String,Component>();
  Dimension dim=new Dimension(20,20);

  JButton createButton(String icon, String cmd){ return createButton(null,icon,cmd,-1);}
  JButton createButton(String icon, String cmd, int key){ return createButton(null,icon,cmd,key);}
  JButton createButton(String text ,String icon,String cmd){return createButton(text,icon,cmd, -1); }
  JButton createButton(String text ,String icon,String cmd, int key)
  {
    JButton button;
    if(text != null && text.length()>0)
      if(icon != null && icon.length()>0)
        button = new JButton(text, LoadIcon.scale(LoadIcon.load(icon),dim));
      else
        button = new JButton(text);
    else button = new JButton(LoadIcon.scale(LoadIcon.load(icon),dim));
    if(list.get(cmd) == null)
    {
      if(key >= 0) button.setMnemonic(key);
      button.addActionListener(this);
      button.setActionCommand(cmd);
      list.put(cmd,button);
      return button;
    }
    return null;
  }

  String getCommandLine(Object c)
  {
    for(String cmd: list.keySet()) if(list.get(cmd) == c) return cmd;
    return "NONE";
  }
```

```
  Component add(Component c, String cmd){ return add(c, null, cmd); }
  Component add(Component c, Object oc, String cmd)
  {
    if( list.get(cmd) == null)
    {
      //c.addActionListener(new ActionL(cmd));
      list.put(cmd,c);
      if(oc!= null) add(c,oc);
      else add(c);
      return c;
    }
    return null;
  }

  void setButtonEnabled(String cmd, boolean b)
  {
    if(cmd != null)
    {
      Component btn=list.get(cmd);
      if( btn != null ) btn.setEnabled(b);
    }
  }

  public void actionPerformed(ActionEvent e)
  { doAction(getCommandLine(e.getSource()), this); }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : LoadIcon.java

LoadIcon.java

```
package gui;

import java.awt.Dimension;
import java.awt.Image;
import java.io.IOException;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;

import static main.Tools.getPath;

/**
 * @author ToPeG
 * Helper Class for loading Icons
 */
public class LoadIcon extends ImageIcon{

  private static final long serialVersionUID = -1257340691914105336L;
  public static String base="imgs/gtk-";
  public static String ending=".png";


  LoadIcon(String name){super(getPath(completePath(name)));}

  private static String completePath(String name){ return base+name+ending; }

  public static ImageIcon load(String name)
  {
    URL imgURL = getPath(completePath(name));
    if (imgURL != null) return new ImageIcon(imgURL);
    return null;
  }

  public static ImageIcon scale(ImageIcon icon, Dimension dim)
  { return new ImageIcon(icon.getImage().getScaledInstance(dim.width,dim.height,Image.SCALE_SMOOTH)); }

  public static Image loadImage(String name)
  {
    URL imgURL = getPath(completePath(name));
    if (imgURL != null)
      try {
        return ImageIO.read(imgURL);
      } catch (IOException e) {
        return null;
      }
    return null;
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : GUI.java

GUI.java

```
package gui;

import java.awt.Dimension;
import java.awt.event.*;
import java.io.File;
import java.util.ArrayList;

import javax.swing.JFileChooser;

import main.Configuration;

import database.DataBase;
import database.Question;

/**
 *
 * @author ToPeG
 * Central Class controlling all GUI Elements
 *
 */
public class GUI implements ActionListener{

  private MainWindow win;
  private HelpWindow help;
  private SummaryWindow summ;
  private Question question;
  private JFileChooser filechooser = new JFileChooser();


  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();
  private boolean probe=true;

  public GUI()
  {
    win = new MainWindow();
    help = new HelpWindow();
    summ= new SummaryWindow();

    win.addActionListener(this);
    help.addActionListener(this);
    summ.addActionListener(this);

    win.endTest();
    hideHelp();
  }

  public void setQuestion(Question q){ question=q; }
  public Question getQuestion(){ return question; }
```

```
  public void startRealTest(int max)
  {
    probe=false;
    win.startTest(max);
  }

  public void startProbeTest(int max)
  {
    win.probeTest();
    probe=true;
    win.startTest(max);
  }

  public void showQuestion()
  { if(question != null) win.setQuestion(question,probe); }

  public void testCorrect()
  {if(question != null && probe) win.testCorrect();}

  public void showSummary(){ summ.setVisible(true); }
  public void setSummaryDataBase(DataBase db){ summ.setDataBase(db); }
  public void hideSummary()
  {
    summ.setDataBase(null);
    summ.setVisible(false);
  }

  public void showHelp(Question q)
  {
    help.setQuestion(q);
    help.setVisible(true);
  }

  public void hideHelp()
  {
    help.setQuestion(null);
    help.setVisible(false);
  }

  public void showConf(Configuration conf)
  {
    new ConfigWindow(win,conf);
    //System.out.println(conf.create());
  }

  public void endTest()
  {
    win.endTest();
    hideHelp();
  }

  public void setQuestionInfoText(int pos, int max){ win.setQuestionInfoText(pos,max); }
  public void disableNext(boolean b){win.disableNext(b);}
  public void disablePrevious(boolean b){win.disablePrevious(b);}
```

```
  public File openFile()
  {
    if (filechooser.showOpenDialog(win) == JFileChooser.APPROVE_OPTION) return filechooser.getSelectedFile();
    return null;
  }

  public File saveAsFile()
  {
    if (filechooser.showSaveDialog(win) == JFileChooser.APPROVE_OPTION) return filechooser.getSelectedFile();
    return null;
  }

  public void showQuestionInfos(Question[] q)
  { win.showQuestionInfoList(q); }

  public void actionPerformed(ActionEvent e)
  {
    String txt=e.getActionCommand();
    Object o= e.getSource();
    if(txt.equals("THELP"))
    { if(question != null)  showHelp(question); }
    else if(txt.equals("SWHELP"))
    { showHelp((Question)o); }
    else if(txt.equals("SWCLOSE"))
    { hideSummary(); }
    else if(txt.equals("HWCLOSE"))
    { hideHelp(); }
    else
    { doAction(txt,o); }
  }

  public void DBConf(DataBase db)
  {
    new DBConfWindow(win,db);
  }

  public void alert(String title, String text, String icon)
  { new Alert(win,title,text,LoadIcon.scale(LoadIcon.load(icon),new Dimension(100,100))); }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : MainWindow.java

MainWindow.java

```
package gui;

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

import database.Question;

import java.util.*;

/**
 *
 * @author ToPeG
 * Main Window
 * Central Window
 *
 */
public class MainWindow extends JFrame{

  private static final long serialVersionUID = -1084744559425615791;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private Menu menu = new Menu();
  private LeisteOben leisteoben = new LeisteOben();
  private ViewQuestion question = new ViewQuestion();
  private Leiste leisteunten = new LeisteUnten();


  public MainWindow()
  {
    super("FragenTester");
    setIconImage(LoadIcon.loadImage("tester"));
    addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e){ actionExit(); }});
    setLayout(new BorderLayout());

    ActionListener acl=new ActionListener(){public void actionPerformed(ActionEvent e){ actionCommand(e.getActionCommand(),e.getSource());}};
    menu.addActionListener(acl);
    leisteoben.addActionListener(acl);
    leisteunten.addActionListener(acl);

    setJMenuBar(menu);
    add(leisteoben,BorderLayout.NORTH);
    add(question,BorderLayout.CENTER);
    add(leisteunten,BorderLayout.SOUTH);
    endTest();

    pack();
    setVisible(true);
  }
```

```
  public void startTest(int max)
  {
    leisteoben.setButtonEnabled("QFIRST", true);
    leisteoben.setButtonEnabled("QBEVORE", true);
    leisteoben.setButtonEnabled("TEND", true);
    leisteoben.setButtonEnabled("QNEXT", true);
    leisteoben.setButtonEnabled("QLAST", true);
    leisteoben.setButtonEnabled("QINFO", true);
    setQuestionInfoText(1,max);

    leisteunten.setButtonEnabled("TEND", true);
    leisteunten.setButtonEnabled("QNEXT", true);

    menu.setItemEnabled("DBSEL",false);
    menu.setItemEnabled("DBCONF",false);
    menu.setItemEnabled("TPROBE",false);
    menu.setItemEnabled("TSTART",false);
    menu.setItemEnabled("TCONF",false);
    menu.setItemEnabled("SSHOW",false);
    menu.setItemEnabled("SSAVE",false);
    menu.setItemEnabled("TEND",true);
    menu.setItemEnabled("SAVE",false);
    menu.setItemEnabled("LOAD",false);
    menu.setItemEnabled("CONF",false);
  }

  public void probeTest()
  {
    menu.setItemEnabled("THELP",true);
    leisteunten.setButtonEnabled("THELP", true);
    leisteoben.setButtonEnabled("QTEST", true);
    leisteoben.setButtonEnabled("THELP", true);
    question.setDisabled(false);
  }


  public void endTest()
  {
    leisteoben.setButtonEnabled("QFIRST", false);
    leisteoben.setButtonEnabled("QBEVORE", false);
    leisteoben.setButtonEnabled("TEND", false);
    leisteoben.setButtonEnabled("QTEST", false);
    leisteoben.setButtonEnabled("THELP", false);
    leisteoben.setButtonEnabled("QNEXT", false);
    leisteoben.setButtonEnabled("QLAST", false);
    leisteoben.setButtonEnabled("QINFO", false);
    setQuestionInfoText(0,0);

    leisteunten.setButtonEnabled("TEND", false);
    leisteunten.setButtonEnabled("QNEXT", false);
    leisteunten.setButtonEnabled("THELP", false);

    menu.setItemEnabled("DBSEL",true);
    menu.setItemEnabled("DBCONF",true);
    menu.setItemEnabled("TPROBE",true);
    menu.setItemEnabled("TSTART",true);
```

```
    menu.setItemEnabled("TCONF",true);
    menu.setItemEnabled("SSHOW",true);
    menu.setItemEnabled("SSAVE",true);
    menu.setItemEnabled("THELP",false);
    menu.setItemEnabled("TEND",false);
    menu.setItemEnabled("SAVE",true);
    menu.setItemEnabled("LOAD",true);
    menu.setItemEnabled("CONF",true);

    //question.setDisabled(true);
    question.setQuestion(null,false);
  }

  public void setQuestion( Question q, boolean e ){ question.setQuestion(q,e); }
  public Question getQuestion(){ return question.getQuestion(); }
  public void testCorrect(){ question.testCorrect(); }
  public void setQuestionInfoText(int pos, int max){ leisteoben.setInfoText(pos+"/"+max); }
  public void showQuestionInfoList(Question[] q)
  { leisteoben.showInfoList(q); }

  public void disableNext(boolean b)
  {
    b=!b;
    leisteoben.setButtonEnabled("QNEXT", b);
    leisteoben.setButtonEnabled("QLAST", b);
    leisteunten.setButtonEnabled("QNEXT", b);
  }

  public void disablePrevious(boolean b)
  {
    b=!b;
    leisteoben.setButtonEnabled("QFIRST", b);
    leisteoben.setButtonEnabled("QBEVORE", b);
  }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
  private void actionExit()
  {
    this.setVisible(false);
    this.dispose();
    doAction("EXIT", this);
  }

  private void actionCommand(String cmd, Object o)
  {
    if(cmd.equals("EXIT")) actionExit();
    else doAction(cmd, o);
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : Menu.java

Menu.java

```
package gui;

import java.awt.event.*;
import java.util.ArrayList;

import javax.swing.*;

/**
 *
 * @author ToPeG
 * The Menu of the MainWindow
 *
 */
public class Menu extends JMenuBar{
  private static final long serialVersionUID = -5617155576631422259L;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();
  private ArrayList<JMenuItem> list = new ArrayList<JMenuItem>();

  private class ActionL implements ActionListener{
    String cmd="";
    Object o=null;
    public ActionL(String s,Object o){cmd=s; this.o=o;}
    public void actionPerformed(ActionEvent e){doAction(cmd,o);}
  }

  public Menu()
  {
    JMenu file = createMenu("Einstellungen",KeyEvent.VK_L);
    file.add(crateMenuItem("OEffnen",       "open",        "LOAD",      KeyEvent.VK_O, KeyEvent.VK_O));
    file.add(crateMenuItem("Speichern",     "save",        "SAVE",      KeyEvent.VK_S, KeyEvent.VK_S));
    file.addSeparator();
    file.add(crateMenuItem("Bearbeiten",    "preferences", "CONF",      KeyEvent.VK_B));
    file.addSeparator();
    file.add(crateMenuItem("Exit",          "quit",        "EXIT",      KeyEvent.VK_X, KeyEvent.VK_Q));

    JMenu db = createMenu("DataBase",       KeyEvent.VK_D);
    db.add(crateMenuItem("Auswählen",       "network",     "DBSEL",     KeyEvent.VK_W, KeyEvent.VK_W));
    db.add(crateMenuItem("Konfiguration",   "properties",  "DBCONF",    KeyEvent.VK_K, KeyEvent.VK_K));

    JMenu test = createMenu("Test",         KeyEvent.VK_T);
    test.add(crateMenuItem("UEbung",        "new",          "TPROBE",   KeyEvent.VK_U, KeyEvent.VK_U));
    test.add(crateMenuItem("Test",          "index",        "TSTART",   KeyEvent.VK_T, KeyEvent.VK_T));
    test.add(crateMenuItem("Test Beenden",  "stop",         "TEND",     KeyEvent.VK_T, KeyEvent.VK_T));
    //test.add(crateMenuItem("Konfiguration", "properties",   "TCONF",    KeyEvent.VK_G));
    test.add(crateMenuItem("Hilfe",         "help",         "THELP",    KeyEvent.VK_H));
    test.add(crateMenuItem("Ergebnisse",    "spell-check",  "SSHOW",    KeyEvent.VK_A,KeyEvent.VK_A));
  }
```

```
  private JMenu createMenu(String text, int key)
  {
    JMenu menu = new JMenu(text);
    if(key>=0) menu.setMnemonic(key);
    add(menu);
    return menu;
  }

  private JMenuItem crateMenuItem(String name, String icon, String action, int key){ return crateMenuItem(name,icon,action,key,-1); }
  private JMenuItem crateMenuItem(String name, String icon, String action, int key, int key2)
  {
    JMenuItem item = new JMenuItem(name,LoadIcon.load(icon));
    if(key2 >=0) item.setAccelerator(KeyStroke.getKeyStroke(key2, ActionEvent.CTRL_MASK ));
    if(key >=0) item.setMnemonic(key);
    item.addActionListener(new ActionL(action,item));
    item.setActionCommand(action);
    list.add(item);
    return item;
  }

  public void setItemEnabled(String cmd, boolean b)
  {
    if(cmd != null)
    {
      for(JMenuItem btn: list)
      { if(btn != null && btn.getActionCommand().equals(cmd)) btn.setEnabled(b); }
    }
  }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : LeisteOben.java

LeisteOben.java

```
package gui;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import database.Question;

/**
 *
 * @author ToPeG
 * MainWindow Navigation ButtonPannel above
 *
 */
public class LeisteOben extends Leiste{
  private static final long serialVersionUID = -1867292702969193620L;

  private MenuButton info;

  LeisteOben()
  {
    setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.gridwidth=1;
    c.gridheight=1;
    c.fill = GridBagConstraints.NONE;
    c.anchor = GridBagConstraints.CENTER;
    c.insets = new Insets(0,2,0,2);
    c.weightx=0.0;
    c.weighty=0.0;
    c.gridy=0;

    c.gridx=0;
    add(createButton("goto-first-ltr","QFIRST"),c);
    c.gridx=1;
    add(createButton("go-back-ltr",   "QBEVORE"),c);
    c.gridx=2;
    add(createButton("stop",          "TEND"),c);
    c.gridx=3;
    add(createButton("spell-check",   "QTEST"),c);
    c.gridx=4;
    add(createButton("help",          "THELP"),c);
    c.gridx=5;
    add(createButton("go-forward-ltr","QNEXT"),c);
    c.gridx=6;
    add(createButton("goto-last-ltr", "QLAST"),c);
```

```
    c.gridx=7;
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.CENTER;
    c.weightx=1.0;
    c.weighty=0.0;
    add(new JLabel(""),c);

    c.gridx=8;
    c.fill = GridBagConstraints.NONE;
    c.anchor = GridBagConstraints.CENTER;
    c.weightx=0.0;
    c.weighty=0.0;
    info=new MenuButton("0/0" ,LoadIcon.scale(LoadIcon.load("about"),dim), "QINFO");
    info.addActionListener(this);
    add(info,c,"QINFO");
  }

  public void actionPerformed(ActionEvent e)
  {
    if(e.getActionCommand().equals("QJUMP"))
    {
      //System.out.println(e.getSource());
      doAction(e.getActionCommand(),e.getSource());
    }
    else
      super.actionPerformed(e);
  }

  public void setInfoText(String t)
  { info.setText(t); }

  public void showInfoList(Question[] q)
  { info.showList(q); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : MenuButton.java

MenuButton.java

```java
package gui;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JList;
import javax.swing.JPopupMenu;
import javax.swing.JScrollPane;

import database.Question;

/**
 *
 * @author ToPeG
 * button With an Menu on it, to select an Question
 *
 */
public class MenuButton extends JButton implements ActionListener{
  private static final long serialVersionUID = 8288320049066554427L;
  private JPopupMenu popup = new JPopupMenu();

  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  public MenuButton(String text ,ImageIcon icon, String cmd, int key)
  {
    super(text, icon);
    init(cmd, key);
  }

  public MenuButton(String text ,ImageIcon icon, String cmd)
  {
    super(text, icon);
    init(cmd, 0);
  }

  public MenuButton(String text ,String cmd, int key)
  {
    super(text);
    init(cmd, key);
  }

  public MenuButton(String text ,String cmd)
  {
    super(text);
    init(cmd, 0);
  }
```

```
  private void init(String cmd, int key)
  {
    setActionCommand(cmd);
    if(key >= 0) setMnemonic(key);
    super.addActionListener(this);
  }

  public void showList(Question[] q)
  {
    //System.out.println("OK "+q);
    popup.removeAll();

    JList QuestionList = new JList(q);
    QuestionCellRenderer qcr = new QuestionCellRenderer();
    qcr.addActionListener(this);
    QuestionList.setCellRenderer(qcr);

    JScrollPane scroll = new JScrollPane(QuestionList);
    scroll.setVerticalScrollBarPolicy( JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    scroll.setPreferredSize(new Dimension(100, 200));
    scroll.setMinimumSize(new Dimension(10, 10));
    popup.add(scroll);
    popup.show(this,0,getHeight());
  }


  public void actionPerformed(ActionEvent e)
  {
    //System.out.println(e.getActionCommand());
    String cmd=e.getActionCommand();

    if(getActionCommand().equals(cmd) && popup.isVisible())
    { popup.menuSelectionChanged(false); }
    else
    {
      doAction(cmd,e.getSource());
      if(cmd.equals("QJUMP")) popup.menuSelectionChanged(false);
    }
  }


  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }

}
```

Projekt: FragenTester
Paket: gui
Klasse : QuestionCellRenderer.java

QuestionCellRenderer.java

```
package gui;

import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.ArrayList;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.ListCellRenderer;

import database.Question;

/**
 *
 * @author ToPeG
 * ListCellRenderer for the menu in MenuButton
 *
 */
public class QuestionCellRenderer extends JLabel implements ListCellRenderer{
  private static final long serialVersionUID = -3790288593771361 97L;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private Icon iconA = loadImage("ask");
  private Icon iconB = loadImage("about");
  private Icon iconC = loadImage("yes");

  private Question last=null;

  QuestionCellRenderer()
  {
    setOpaque(true);
    setIconTextGap(12);
  }

  public Component getListCellRendererComponent(JList list, Object o, int pos, boolean selected, boolean focused)
  {
    Question q=(Question)o;

    Icon icon = iconA;
    if(q.isAnswered()) icon = iconB;
    if(q.isActive()) icon = iconC;
    setIcon(icon);

    setText( q.getId() );
```

```
    if (selected) {
      setBackground( list.getSelectionBackground() );
      setForeground( list.getSelectionForeground() );
      if(last!=q)
      {
        doAction("QJUMP",o);
        last=q;
      }
    }
    else {
      setBackground( list.getBackground() );
      setForeground( list.getForeground() );
      setIcon(icon);
    }
    return this;
  }

  public static ImageIcon loadImage(String name)
  {
    try {
      return new ImageIcon(new File("imgs/gtk-"+name+".png").toURI().toURL(),name);
    } catch (Exception e) {
      return null;
    }
  }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ViewQuestion.java

ViewQuestion.java

```
package gui;

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

import database.Question;

/**
 *
 * @author ToPeG
 * Display an Question
 *
 */
public class ViewQuestion extends JPanel{
  private static final long serialVersionUID = -2620930762197556811L;

  private JLabel question;
  private ShowAnswers answers;
  private JLabel ltest;
  private JLabel lid;
  private JButton btest;

  private Question frage;

  ViewQuestion()
  {
    super();
    setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.gridwidth=1;
    c.gridheight=1;
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    c.insets = new Insets(10,10,10,10);
    c.weightx=1.0;
    c.weighty=0.0;
    c.gridx=0;

    c.gridy=0;
    lid=new JLabel();
    add(lid,c);

    c.gridy=1;
    c.weighty=1.0;
    c.fill = GridBagConstraints.BOTH;
    question = new ShowQuestion("");
    JPanel in = new JPanel(new BorderLayout());
    in.add(question,BorderLayout.NORTH);
```

```
    JScrollPane scroll = new JScrollPane(in);
    scroll.setVerticalScrollBarPolicy( JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    scroll.setHorizontalScrollBarPolicy( JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    scroll.setBorder(null);
    add(scroll,c);

    c.fill = GridBagConstraints.HORIZONTAL;
    c.gridy=2;
    c.weighty=0.0;
    answers=new ShowAnswers();
    add(answers,c);

    c.gridy=3;
    c.weighty=0.0;
    c.fill = GridBagConstraints.HORIZONTAL;
    btest =new JButton("Antwort Püfen",LoadIcon.scale(LoadIcon.load("spell-check"),new Dimension(20,20)));
    btest.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e) { testCorrect(); }});
    add(btest,c);

    c.gridy=4;
    ltest =new JLabel("",JLabel.CENTER);
    add(ltest,c);

    cleanup();
  }

  private void cleanup()
  {
    setId("");
    setQuestion(null);
    answers.show(null);
    btest.setEnabled(false);
    resetCorrect();
  }

  public void setQuestion(Question q,boolean enable)
  {
    frage=q;
    if(q != null)
    {
      setId(frage.getId());
      setQuestion(frage.getQuestionText());
      btest.setEnabled(enable);
      answers.show(frage.getAnswers());
      resetCorrect();
    }
    else
    { cleanup(); }
  }

  public Question getQuestion()
  { return frage; }

  private void setId(String id)
  { lid.setText("Frage Nr.: "+id); }
```

```
  private void setQuestion(String q)
  { question.setText(q); }

  public void setTestButton()
  { btest.setEnabled(true); }

  private void resetCorrect()
  { ltest.setText("<html><font color=green size=+4> </font></html>"); }

  public void testCorrect()
  {
    if(frage.isAnswered())
    {
      if(frage.testUserAnswers())
      { ltest.setText("<html><font color=green size=+4>KORREKT</font></html>"); }
      else
      { ltest.setText("<html><font color=red size=+4>FALSCH</font></html>"); }
    }
    else
    { resetCorrect(); }
  }

  public void setDisabled(boolean b)
  {
    btest.setEnabled(!b);
    if(b)
    {
      setQuestion("");
      answers.show(null);
      resetCorrect();
      setId("");
    }
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ShowQuestion.java

ShowQuestion.java

```java
package gui;

import javax.swing.JLabel;

/**
 *
 * @author ToPeG
 * Formated Output for the Question
 *
 */
public class ShowQuestion extends JLabel{
  private static final long serialVersionUID = 1778660560794681275L;

  ShowQuestion(String text)
  {
    super();
    setText(text);
    setVerticalTextPosition(JLabel.NORTH);
    setHorizontalTextPosition(JLabel.RIGHT);
  }

  public void setText(String txt)
  {
    if(txt == null || txt.equals("")) super.setText(null);
    else super.setText(Tools.preformatQuestion(txt));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ShowAnswers.java

ShowAnswers.java

```
package gui;

import java.awt.Component;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputMethodEvent;
import java.awt.event.InputMethodListener;

import javax.swing.ButtonGroup;
import javax.swing.JCheckBox;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTextField;
import javax.swing.event.CaretEvent;
import javax.swing.event.CaretListener;

import database.Answer;
import database.Question.Type;


/**
 *
 * @author ToPeG
 * Create an List of the Answers
 *
 */
public class ShowAnswers extends JPanel implements ActionListener,InputMethodListener,CaretListener{
  private static final long serialVersionUID = 4415408140549640400L;
  private Answer[] answers;
  private ButtonGroup group = new ButtonGroup();

  ShowAnswers()
  {
    super();
    delete();
  }

  void delete()
  {
    setLayout(null);
  }

  void show(Answer[] aa)
  {
    this.answers=aa;
    for(Component c : this.getComponents()) remove(c);
    if(answers != null)
    {
```

```
      setLayout(new GridLayout(answers.length,1));
      for(int i=0; i<answers.length; i++)
      {
        Answer a=answers[i];
        if(a.getType()==Type.single)
        {
          add(createRadioButton(a.getText(),""+i,a.getUserSelection()));
        }
        else if(a.getType()==Type.multi)
        {
          add(createCheckBox(a.getText(),""+i,a.getUserSelection()));
        }
        else
        {
          add(createTextField(a.getUserText(),"txt"));
        }
      }
    }
    else
    { delete(); }
  }



  private JRadioButton createRadioButton(String txt, String cmd, boolean sel)
  {
    JRadioButton rb = new JRadioButton(Tools.preformatQuestion(txt));
    rb.setActionCommand(cmd);
    rb.addActionListener(this);
    group.add(rb);
    rb.setSelected(sel);
    return rb;
  }

  private JCheckBox createCheckBox(String txt, String cmd, boolean sel)
  {
    JCheckBox cb = new JCheckBox(Tools.preformatQuestion(txt));
    cb.setActionCommand(cmd);
    cb.addActionListener(this);
    cb.setSelected(sel);
    return cb;
  }



  private JTextField createTextField(String txt, String cmd)
  {
    JTextField text = new JTextField(10);
    if(txt != null) text.setText(txt);
    text.addInputMethodListener(this);
    text.addCaretListener(this);
    text.setActionCommand(cmd);
    text.addActionListener(this);
    return text;
  }
```

```
  public void actionPerformed(ActionEvent e)
  {
    String cmd=e.getActionCommand();
    if(answers != null)
    {
      Type type= answers[0].getType();
      if(type == Type.single)
      {
        for(int i=0; i<answers.length; i++)
        {
          answers[i].setUserSelection(false);
          if(cmd.equals(""+i))
          { answers[i].setUserSelection(true); }
        }
      }
      else if(type == Type.multi)
      {
        int i=Integer.parseInt(cmd);
        answers[i].setUserSelection(!answers[i].getUserSelection());
      }
      else
      { answers[0].setUserText(((JTextField)e.getSource()).getText()); }
    }
  }

  public void caretPositionChanged(InputMethodEvent ime) {}
  public void inputMethodTextChanged(InputMethodEvent ime)
  {
    //System.out.println(((JTextField)ime.getSource()).getText());
    actionPerformed(new ActionEvent(ime.getSource(),0,"txt"));
  }

  public void caretUpdate(CaretEvent e) {
    //System.out.println(((JTextField)e.getSource()).getText());
    actionPerformed(new ActionEvent(e.getSource(),0,"txt"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : LeisteUnten.java

LeisteUnten.java

```
package gui;

import java.awt.FlowLayout;

/**
 *
 * @author ToPeG
 * MainWindow navigation ButtonPanel bottom
 *
 */
public class LeisteUnten extends Leiste{
  private static final long serialVersionUID = 7087570819992062O9L;

  public LeisteUnten()
  {
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    add(createButton("Test Beenden", "stop",          "TEND"));
    add(createButton("HILFE",        "help",          "THELP"));
    add(createButton("Nächste",      "go-forward-ltr","QNEXT"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ConfigWindow.java

ConfigWindow.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;

import javax.swing.*;

import main.Configuration;


/**
 *
 * @author ToPeG
 * Configure the Application
 *
 */
public class ConfigWindow extends JDialog{
  private static final long serialVersionUID = -2518570718935913631;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private Leiste leiste = new ConfLeiste();
  private ConfPanel panel = null;

  public ConfigWindow(Frame owner, Configuration conf)
  {
    super(owner,"Konfiguration",true);
    setSize(500,450);
    //setResizable(false);
    setIconImage(LoadIcon.loadImage("tester"));
    addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e){ setVisible(false); dispose();}});
    setLayout(new BorderLayout());

    if(conf != null)
    {
      panel=new ConfPanel(conf.clone());
      add(panel,BorderLayout.CENTER);
    }

    leiste.addActionListener(new ActionListener(){public void actionPerformed(ActionEvent e){ actionCommand(e.getActionCommand(),e.getSource());}});
    add(leiste,BorderLayout.SOUTH);

    setVisible(true);
  }
```

```
  private void actionCommand(String cmd, Object source)
  {
    if(cmd.equals("CWCLOSE")){setVisible(false);dispose();}
    else if(cmd.equals("CWACEPT"))
    {
      sync();
    }
    else if(cmd.equals("CWOK"))
    {
      sync();
      setVisible(false);dispose();
    }
    else
    doAction(cmd,source);
  }

  private void sync()
  { if(panel!= null) doAction("CWSYNC",panel.getConf()); }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ConfLeiste.java

ConfLeiste.java

```
package gui;

import java.awt.FlowLayout;

/**
 *
 * @author ToPeG
 *
 * Panel for the ConfigWindow
 *
 */
public class ConfLeiste extends Leiste{
  private static final long serialVersionUID = -6969878578414949900L;

  public ConfLeiste()
  {
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    add(createButton("Übernehmen", "ok", "CWACEPT"));
    add(createButton("Abbruch", "cancel", "CWCLOSE"));
    add(createButton("OK", "apply", "CWOK"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : ConfPanel.java

ConfPanel.java

```
package gui;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Arrays;

import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import main.Configuration;

/**
 *
 * @author ToPeG
 * Inherits of the ConfigWindow
 *
 */
public class ConfPanel extends JPanel implements ActionListener,ListSelectionListener{
  private static final long serialVersionUID = 80269991187271256 42L;
  private Configuration conf = null;

  private JComboBox group;
  private DefaultListModel listmodel = new DefaultListModel();
  private JList list = new JList(listmodel);
  private JTextField value = new JTextField(20);
  private JTextField key = new JTextField(20);


  public ConfPanel(Configuration c)
  {
    conf=c;
    if(c!= null) init();
  }

  private void init()
  {
    setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.gridwidth=2;
    c.gridheight=1;
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    c.insets = new Insets(10,0,10,0);
    c.weightx=1.0;
    c.weighty=0.0;
    c.gridx=0;
    c.gridy=0;
```

```
    {
      JPanel panel= new JPanel();
      add(panel,c);
      panel.setLayout(new GridBagLayout());

      GridBagConstraints pc = new GridBagConstraints();
      pc.gridwidth=1;
      pc.gridheight=1;
      pc.fill = GridBagConstraints.HORIZONTAL;
      pc.anchor = GridBagConstraints.FIRST_LINE_START;
      pc.insets = new Insets(10,10,10,10);
      pc.weightx=0.0;
      pc.weighty=0.0;
      pc.gridx=0;
      pc.gridy=0;

      panel.add(new JLabel("Gruppe:"),pc);

      pc.weightx=1.0;
      pc.gridx=1;

      group = new JComboBox(conf.getAsStringArray("global","sort"));
      group.setSelectedIndex(0);
      group.setEditable(false);
      group.setActionCommand("UPDATELIST");
      group.addActionListener( this );
      panel.add(group,pc);
    }


    c.weightx=1.0;
    c.gridx=0;
    c.gridy=1;
    c.gridwidth=2;
    c.anchor = GridBagConstraints.SOUTHWEST;
    add(new JLabel("Keys:"),c);

    c.gridx=0;
    c.gridy=2;
    c.gridwidth=1;
    c.weightx=1.0;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    c.fill = GridBagConstraints.BOTH;
    list.addListSelectionListener( this );
    add(list,c);

    c.gridx=1;
    c.gridwidth=1;
    c.gridwidth=1;
    c.weightx=0.0;
    c.weighty=1.0;
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    {
      JPanel panel= new JPanel();
      add(panel,c);
```

```
        panel.setLayout(new GridBagLayout());

        GridBagConstraints pc = new GridBagConstraints();
        pc.gridwidth=2;
        pc.gridheight=1;
        pc.fill = GridBagConstraints.HORIZONTAL;
        pc.anchor = GridBagConstraints.FIRST_LINE_START;
        pc.insets = new Insets(10,10,10,10);
        pc.weightx=1.0;
        pc.weighty=0.0;
        pc.gridx=0;
        pc.gridy=0;

        pc.insets = new Insets(10,10,0,10);
        panel.add(new JLabel("Key:"),pc);

        pc.insets = new Insets(0,10,10,10);
        pc.gridy=1;
        key.setColumns(1);
        panel.add(key,pc);

        pc.insets = new Insets(10,10,0,10);
        pc.gridy=2;
        panel.add(new JLabel("Value:"), pc);

        pc.insets = new Insets(0,10,10,10);
        pc.gridy=3;
        value.setColumns(1);
        panel.add(value,pc);

        pc.insets = new Insets(10,10,10,10);
        pc.gridy=4;
        JButton change= new JButton("Übernehmen");
        change.setActionCommand("SETVALUE");
        change.addActionListener( this );
        panel.add(change,pc);

        pc.gridwidth=1;
        pc.gridy=5;
        pc.gridx=0;
        JButton delete= new JButton("Löschen");
        delete.setActionCommand("DELVALUE");
        delete.addActionListener( this );
        panel.add(delete,pc);

        pc.gridx=1;
        JButton add= new JButton("Hinzufügen");
        add.setActionCommand("ADDVALUE");
        add.addActionListener( this );
        panel.add(add,pc);

        pc.gridwidth=2;
        pc.gridx=0;
        pc.gridy=6;
        pc.fill = GridBagConstraints.BOTH;
        pc.anchor = GridBagConstraints.CENTER;
```

```
      pc.insets = new Insets(0,0,0,0);
      panel.add(new Label(), pc);
    }

    updateList();
  }

  private void updateList()
  {
    if(group.getSelectedIndex() != -1)
    {
      listmodel.clear();
      String[] keys=conf.getKeys((String)group.getSelectedItem());
      Arrays.sort(keys);
      for(String key : keys)
      { listmodel.addElement(key); }
      updateValue();
    }
  }

  private void updateValue()
  {
    if(group.getSelectedIndex() != -1 && list.getSelectedIndex() != -1)
    {
      String grp =(String)(group.getSelectedItem());
      String k = (String)(list.getSelectedValue());
      key.setText(k);
      value.setText(conf.get(grp,k));
    }
    else
    {
      value.setText("");
      key.setText("");
    }
  }

  private void changeValue()
  {
    if(group.getSelectedIndex() != -1)
    {
      String grp =(String)(group.getSelectedItem());
      String val = value.getText();
      String k=(String)list.getSelectedValue();
      if(k != null && grp != null)
      { conf.set(grp,k,val); }
    }
  }

  private void addKey()
  {
    if(group.getSelectedIndex() != -1)
    {
      String grp =(String)(group.getSelectedItem());
      String val = value.getText();
      String k= key.getText();
      if(k != null && !k.equals("") && grp != null)
```

```
      {
        listmodel.addElement(k);
        conf.set(grp,k,val);
      }
    }
  }

  private void removeKey()
  {
    if(group.getSelectedIndex() != -1)
    {
      String grp =(String)(group.getSelectedItem());
      String k=(String)list.getSelectedValue();
      if(k != null && grp != null)
      {
        listmodel.removeElementAt(list.getSelectedIndex());
        conf.remove(grp,k);
      }
    }
  }

  public Configuration getConf(){ return conf; }

  public void valueChanged(ListSelectionEvent e){actionPerformed(new ActionEvent(e.getSource(),0,"UPDATEVALUE"));}
  public void actionPerformed(ActionEvent e)
  {
    String txt=e.getActionCommand();
    if(txt.equals("UPDATELIST"))
    {
      updateList();
    }
    else if(txt.equals("UPDATEVALUE"))
    {
      updateValue();
    }
    else if(txt.equals("SETVALUE"))
    {
      changeValue();
    }
    else if(txt.equals("DELVALUE"))
    {
      removeKey();
    }
    else if(txt.equals("ADDVALUE"))
    {
      addKey();
    }
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : DBConfWindow.java

DBConfWindow.java

```java
package gui;

import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;

import javax.swing.*;

import database.DataBase;

/**
 * @author ToPeG
 * The DatabaseConfigWindow
 */
public class DBConfWindow extends JDialog{
  private static final long serialVersionUID = 8743621564739338168L;
  private Leiste leiste = new DBConfLeiste();
  private DBConfPanel panel;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  public DBConfWindow(Frame owner, DataBase db)
  {
    super(owner,"DatenbankEinrichten",true);
    setSize(500,450);
    setResizable(false);
    setIconImage(LoadIcon.loadImage("tester"));
    addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e){ setVisible(false); dispose();}});
    setLayout(new BorderLayout());

    ActionListener acl=new ActionListener(){public void actionPerformed(ActionEvent e){ actionCommand(e.getActionCommand(),e.getSource());}};
    add(leiste,BorderLayout.SOUTH);
    leiste.addActionListener(acl);

    if(db!=null)
    {
      panel=new DBConfPanel(db);
      add(panel,BorderLayout.CENTER);
    }
    else
    {
      leiste.setButtonEnabled("DBCWACEPT", false);
      leiste.setButtonEnabled("DBCWOK", false);
    }

    setVisible(true);
  }
```

```
  private void actionCommand(String cmd, Object source)
  {
    if(cmd.equals("DBCWCLOSE")){setVisible(false);dispose();}
    else if(cmd.equals("DBCWACEPT")){panel.updateDB();}
    else if(cmd.equals("DBCWOK")){panel.updateDB();setVisible(false);dispose();}
    else
    doAction(cmd,source);
  }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: gui
Klasse : DBConfLeiste.java

DBConfLeiste.java

```java
package gui;

import java.awt.FlowLayout;

/**
 *
 * @author ToPeG
 * Button Panel of the DatabaseConfigWindow
 *
 */
public class DBConfLeiste extends Leiste{
  private static final long serialVersionUID = -6969878578414949900L;

  public DBConfLeiste()
  {
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    add(createButton("Übernehmen", "ok", "DBCWACEPT"));
    add(createButton("Abbruch", "cancel", "DBCWCLOSE"));
    add(createButton("OK", "apply", "DBCWOK"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : DBConfPanel.java

DBConfPanel.java

```
package gui;

import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import database.DataBase;

/**
 *
 * @author ToPeG
 * Inherits of the DatabaseConfigWindow
 *
 */
public class DBConfPanel extends JPanel implements ActionListener{
  private static final long serialVersionUID = -8978737207601237018L;
  private DataBase db=null;

  private JCheckBox selsingle;
  private JCheckBox selmulti;
  private JCheckBox selregexp;
  private JCheckBox selrandq;
  private JCheckBox selranda;

  private JSpinner spinnmin;
  private JSpinner spinnmax;
  private JLabel anz =new JLabel("");
  private JLabel summsingle =new JLabel("");
  private JLabel summmulti =new JLabel("");
  private JLabel summregexp =new JLabel("");
  private JLabel summ =new JLabel("");

  public DBConfPanel(DataBase d)
  {
    db=d;
    if(db!= null) init();
  }

  private void  init()
  {
    setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.gridwidth=1;
    c.gridheight=1;
```

```
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    c.insets = new Insets(10,0,10,0);
    c.weightx=1.0;
    c.weighty=0.0;
    c.gridx=0;
    c.gridy=0;

    add(new JLabel("Gesammtzahl der Fragen im Datensatz: "+db.size()),c);

    c.gridy=1;
    add(new JLabel("Fragenbereich Auswählen:"),c);

    c.gridy=2;
    {
      JPanel jp= new JPanel();
      add(jp,c);
      jp.setLayout(new GridBagLayout());
      GridBagConstraints jpc = new GridBagConstraints();
      jpc.gridwidth=1;
      jpc.gridheight=1;
      jpc.fill = GridBagConstraints.HORIZONTAL;
      jpc.anchor = GridBagConstraints.FIRST_LINE_START;
      jpc.insets = new Insets(10,10,10,10);
      jpc.weightx=0.0;
      jpc.weighty=0.0;
      jpc.gridx=0;
      jpc.gridy=0;

      jp.add(new JLabel(" von:"),jpc);

      jpc.weightx=1.0;
      jpc.gridx=1;
      spinnmin=createSpinner("CWSPINNMIN",db.getMin()+1,1,db.size());
      jp.add(spinnmin,jpc);

      jpc.weightx=0.0;
      jpc.gridx=2;
      jp.add(new JLabel(" bis:"),jpc);

      jpc.weightx=1.0;
      jpc.gridx=3;
      spinnmax=createSpinner("CWSPINNMAX",db.getMax()+1,2,db.size());
      jp.add(spinnmax,jpc);

      jpc.weightx=0.0;
      jpc.gridx=4;
      jp.add(new JLabel("   Anzahl:"),jpc);

      jpc.weightx=1.0;
      jpc.gridx=5;
      jp.add(anz,jpc);
    }

    c.gridy=3;
    {
```

```
      JPanel jp= new JPanel();
      add(jp,c);
      jp.setLayout(new GridBagLayout());
      GridBagConstraints jpc = new GridBagConstraints();
      jpc.gridwidth=1;
      jpc.gridheight=1;
      jpc.fill = GridBagConstraints.HORIZONTAL;
      jpc.anchor = GridBagConstraints.LINE_START;
      jpc.insets = new Insets(0,0,0,0);
      jpc.weighty=0.0;


      jpc.gridy=0;
      jpc.gridx=0;
      jpc.weightx=0.0;
      selsingle=createCheckBox("Single Choice Fragen","CWSELSINGLE",!db.isSingleExcuded());
      jp.add(selsingle,jpc);

      jpc.weightx=1.0;
      jpc.gridx=1;
      jp.add(summsingle,jpc);


      jpc.gridy=1;
      jpc.gridx=0;
      jpc.weightx=0.0;
      selmulti=createCheckBox("Multiple Choice Fragen","CWSELMULTI",!db.isMultiExcuded());
      jp.add(selmulti,jpc);

      jpc.gridx=1;
      jpc.weightx=1.0;
      jp.add(summmulti,jpc);


      jpc.gridy=2;
      jpc.gridx=0;
      jpc.weightx=0.0;
      selregexp=createCheckBox("Text Fragen","CWSELREGEXP",!db.isRegexpExcuded());
      jp.add(selregexp,jpc);

      jpc.gridx=1;
      jpc.weightx=1.0;
      jp.add(summregexp,jpc);
    }

    c.gridy=4;
    add(new JLabel(" "),c);

    {
      JPanel jp= new JPanel();
      add(jp,c);
      jp.setLayout(new FlowLayout(FlowLayout.LEFT));
      jp.add(new JLabel("Anzahl der Ausgewählten Fragen: "));
      jp.add(summ);
    }
```

```
    c.gridy=5;
    selrandq=createCheckBox("Fragen Sortiert","CWSELRANDQ",db.isSortedQuestions());
    add(selrandq,c);

    c.gridy=6;
    selranda=createCheckBox("Antworten Sortiert","CWSELRANDQ",db.isSortedAnswers());
    add(selranda,c);

    c.gridy=7;
    c.weighty=1.0;
    c.weightx=1.0;
    add(new JLabel(),c);
    db.sortQuestions();
    db.sortAnswers();
    update_summ();
  }

  private void update_summ()
  {
    if(db != null)
    {
      boolean si=selsingle.isSelected();
      boolean sm=selmulti.isSelected();
      boolean sr=selregexp.isSelected();
      int min=((Integer)spinnmin.getValue()).intValue()-1;
      int max=((Integer)spinnmax.getValue()).intValue()-1;

      if(min>=max)
      {
        spinnmin.setValue(spinnmax.getValue());
        spinnmax.setValue(new Integer(((Integer)spinnmax.getValue()).intValue()+1));
      }
      else if(max <= min)
      {
        spinnmax.setValue(spinnmin.getValue());
        spinnmin.setValue(new Integer(((Integer)spinnmin.getValue()).intValue()-1));
      }


      anz.setText(""+db.countAllPossibleQuestions(min,max,true,true,true));
      summsingle.setText(""+db.countAllPossibleQuestions(min,max,true,false,false));
      summmulti.setText(""+db.countAllPossibleQuestions(min,max,false,true,false));
      summregexp.setText(""+db.countAllPossibleQuestions(min,max,false,false,true));

      summ.setText(""+db.countAllPossibleQuestions(min,max,si,sm,sr));
    }
  }

  private JCheckBox createCheckBox(String txt, String cmd, boolean sel)
  {
    JCheckBox cb = new JCheckBox(txt);
    cb.setActionCommand(cmd);
    cb.addActionListener(this);
    cb.setSelected(sel);
    return cb;
  }
```

```
  private JSpinner createSpinner(final String cmd, int now, int min, int max)
  {
    JSpinner sb = new JSpinner();
    sb.setModel(new SpinnerNumberModel(now, min, max, 1));
    sb.addChangeListener(new ChangeListener(){ public void stateChanged(ChangeEvent e) { actionPerformed(new ActionEvent(e.getSource(),0,cmd));}});
    return sb;
  }


  public void updateDB()
  {
    db.excludeSingle(!selsingle.isSelected());
    db.excludeMulti(!selmulti.isSelected());
    db.excludeRegexp(!selregexp.isSelected());
    int min=((Integer)spinnmin.getValue()).intValue()-1;
    int max=((Integer)spinnmax.getValue()).intValue()-1;
    db.setRange(min,max);

    if(selrandq.isSelected()) db.sortQuestions();
    else db.randomQuestions();

    if(selranda.isSelected()) db.sortAnswers();
    else db.randomAnswers();
  }

  public void actionPerformed(ActionEvent e)
  { update_summ(); }

}
```

Projekt: FragenTester
Paket: gui
Klasse : HelpWindow.java

HelpWindow.java

```
package gui;

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;
import database.Question;

/**
 * @author ToPeG
 * HelpWindow
 * Shows Information and Descriptions about an Question
 */
public class HelpWindow extends JFrame{
  private static final long serialVersionUID = -2474120333070019286L;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private JLabel textId;
  private JLabel textQuestion;
  private JTextPane textAnswer;
  private JTextPane textDescription;
  private Question question=null;

  public HelpWindow(){
    super("HILFE");
    setSize(300,400);
    setIconImage(LoadIcon.loadImage("tester"));
    addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e){ actionClose(); }});
    setLayout(new BorderLayout());

    HelpLeiste hl=new HelpLeiste();
    hl.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){ performAction(e.getActionCommand()); }});
    add(hl,BorderLayout.SOUTH);

    JPanel panelQuestion = new JPanel();
    panelQuestion.setLayout(new GridBagLayout());
    add(panelQuestion,BorderLayout.NORTH);

    GridBagConstraints c = new GridBagConstraints();
    c.fill = GridBagConstraints.HORIZONTAL;
    c.anchor = GridBagConstraints.FIRST_LINE_START;
    c.insets = new Insets(5,5,5,5);
    c.weightx=1.0;
    c.weighty=0.0;
    c.gridx=0;
    c.gridy=0;

    textId = new JLabel();
    panelQuestion.add(textId,c);
```

```
    c.gridy=1;
    textQuestion = new JLabel();
    panelQuestion.add(textQuestion,c);

    c.gridy=2;
    panelQuestion.add(new JSeparator(SwingConstants.HORIZONTAL),c);

    c.gridy=3;
    panelQuestion.add(new JLabel("Korrekte Antwort:"),c);

    c.gridy=4;
    textAnswer = new JTextPane();
    panelQuestion.add(textAnswer,c);

    c.gridy=5;
    panelQuestion.add(new JSeparator(SwingConstants.HORIZONTAL),c);

    c.gridy=6;
    panelQuestion.add(new JLabel("Erklährung:"),c);

    textDescription = new JTextPane();
    add(textDescription,BorderLayout.CENTER);
  }

  public void setQuestion(Question q)
  {
    question = q;
    if(question != null)
    {
      String txt=Tools.preformatQuestion(q.getQuestionText());
      txt=txt.replaceFirst("<html>","<html><font color=gray>Frage Nr.: "+q.getId()+"</font><br>");
      textQuestion.setText(txt);
      textAnswer.setText(q.getTrueAnswersAsString());
      textDescription.setText(q.getDescriptionText());
    }
    else
    {
      textQuestion.setText("");
      textQuestion.setText("");
      textAnswer.setText("");
      textDescription.setText("");
    }
  }

  private void performAction(String cmd){ if(cmd.equals("HWCLOSE")) actionClose(); }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name){ for(ActionListener l:action) l.actionPerformed(new ActionEvent(this, 0, name)); }

  private void actionClose()
  {
    this.setVisible(false);
    doAction("HWCLOSE");
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : HelpLeiste.java

HelpLeiste.java

```
package gui;

import java.awt.FlowLayout;

/**
 *
 * @author ToPeG
 * Button Panel for the HelpWindow
 */
public class HelpLeiste extends Leiste{
  private static final long serialVersionUID = 7087570819992062O9L;

  public HelpLeiste()
  {
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    add(createButton("Schließen", "cancel", "HWCLOSE"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : SummaryWindow.java

SummaryWindow.java

```
package gui;

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

import database.DataBase;

/**
 *
 * @author ToPeG
 * the Question Summary Window
 * Displays all questions which are tested
 *
 */
public class SummaryWindow extends JFrame{
  private static final long serialVersionUID = -2975223359246555595L;
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private SummaryList list;
  private JLabel score = new JLabel();

  SummaryWindow()
  {
    super("Zusammenfassung");
    setSize(300,400);
    setIconImage(LoadIcon.loadImage("tester"));
    addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e){ actionClose(); }});
    setLayout(new BorderLayout());

    add(score,BorderLayout.NORTH);

    list= new SummaryList();
    list.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){ performAction(e.getActionCommand(), e.getSource()); }});

    JScrollPane scroll = new JScrollPane(list);
    scroll.setVerticalScrollBarPolicy( JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    scroll.setPreferredSize(new Dimension(250, 145));
    scroll.setMinimumSize(new Dimension(10, 10));
    add(scroll,BorderLayout.CENTER);

    SummaryLeiste hl=new SummaryLeiste();
    hl.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){ performAction(e.getActionCommand(), e.getSource()); }});
    add(hl,BorderLayout.SOUTH);
    setVisible(false);
  }
```

```
  public void setDataBase(DataBase db)
  {
    if(db!= null) score.setText("Erreichte Punkte "+db.getUserScore()+" von maximal "+db.getScore());
    else score.setText("");

    list.setDataBase(db);
  }

  private void performAction(String cmd, Object o)
  {
    if(cmd.equals("SWCLOSE"))     actionClose();
    else if(cmd.equals("SWHELP")) action(cmd,o);
  }


  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void action(String name, Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }

  private void actionClose()
  {
    this.setVisible(false);
    list.setDataBase(null);
    //this.dispose();
    action("SWCLOSE", this);
  }

}
```

Projekt: FragenTester
Paket: gui
Klasse : SummaryLeiste.java

SummaryLeiste.java

```
package gui;

import java.awt.FlowLayout;

/**
 *
 * @author ToPeG
 * ButtonPanel for the QuestionSummaryWindow
 *
 */
public class SummaryLeiste extends Leiste{
  private static final long serialVersionUID = -6299627089646002539L;

  public SummaryLeiste()
  {
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    add(createButton("Schließen", "cancel", "SWCLOSE"));
  }
}
```

Projekt: FragenTester
Paket: gui
Klasse : SummaryList.java

SummaryList.java

```
package gui;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.*;

import database.*;
import database.Question.Type;

/**
 *
 * @author ToPeG
 * Summary of The Questions witch are tested
 */
public class SummaryList extends JPanel{
  private static final long serialVersionUID = -2506391788865523138L;
  private GridBagConstraints gbc = new GridBagConstraints();
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();

  private class myAL implements ActionListener
  {
    String cmd="";
    Question quest;
    myAL(String s, Question q){cmd =s; quest=q;}
    public void actionPerformed(ActionEvent e){ doAction(cmd,quest); }
  }

  SummaryList()
  {
    setLayout( new GridBagLayout() );
    cleanList();
    gbc.weighty=1.0;
    gbc.gridy++;
    add(new JLabel(),gbc);
  }
```

```
  private void cleanList()
  {
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.insets = new Insets(5,5,5,5);
    gbc.gridwidth=0;
    gbc.weightx=1.0;
    gbc.weighty=0.0;
    gbc.gridx=0;
    gbc.gridy=0;

    for(Component c : this.getComponents()) remove(c);
    addDesc();
  }

  public void setDataBase(DataBase db)
  {
    cleanList();
    int pos=1;
    if(db != null)
    {
      for(Question q: db.getAllPossibleQuestions())
      {
        addQuestion(q,pos);
        pos++;
      }
    }
    gbc.weightx=1.0;
    gbc.weighty=1.0;
    gbc.gridx=0;
    gbc.gridy=pos*2;
    gbc.gridwidth=5;
    add(new JLabel(),gbc);
  }

  private void addQuestion(Question q, int pos)
  {
    pos=pos*2;
    JLabel sid = new JLabel(q.getId());
    JLabel icon = new JLabel(LoadIcon.load("ask"));
    if(q.isAnswered())
    {
      if(q.testUserAnswers()) icon = new JLabel(LoadIcon.load("apply"));
      else icon = new JLabel(LoadIcon.load("cancel"));
    }
```

```
    JLabel quest = new JLabel(Tools.preformatQuestion(q.getQuestionText()));

    JLabel typ = new JLabel(LoadIcon.load("string"));

    if(q.getType() == Type.single) typ = new JLabel(LoadIcon.load("single"));
    else if(q.getType() == Type.multi) typ = new JLabel(LoadIcon.load("multi"));

    JButton help = new JButton(LoadIcon.load("help"));
    help.addActionListener( new myAL("SWHELP",q) );
    help.setBorderPainted(false);
    help.setContentAreaFilled(false);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.weighty=0.0;
    gbc.gridy=pos;
    gbc.gridx=0;
    gbc.gridwidth=1;

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=0;
    add(typ, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=1;
    add(sid, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=2;
    add(icon, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=3;
    add(help, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=4;
    add(quest, gbc);
```

```
    gbc.weightx=1.0;
    gbc.gridx=0;
    gbc.gridy++;
    gbc.gridwidth=5;
    add(new JSeparator(SwingConstants.HORIZONTAL),gbc);
  }

  private void addDesc()
  {
    JLabel typ = new JLabel("Typ");
    JLabel sid = new JLabel("ID");
    JLabel icon = new JLabel("Antw.");
    JLabel help = new JLabel("Hilfe");
    JLabel quest = new JLabel("Frage");

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.weighty=0.0;
    gbc.gridx=0;
    gbc.gridy=0;
    gbc.gridwidth=1;

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=0;
    add(typ, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=1;
    add(sid, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=0.0;
    gbc.gridx=2;
    add(icon, gbc);

    gbc.anchor = GridBagConstraints.CENTER;
    gbc.weightx=1.0;
    gbc.gridx=3;
    add(help, gbc);

    gbc.anchor = GridBagConstraints.FIRST_LINE_START;
    gbc.weightx=1.0;
    gbc.gridx=4;
```

```
    add(quest, gbc);

    gbc.weightx=1.0;
    gbc.gridx=0;
    gbc.gridy=1;
    gbc.gridwidth=5;
    add(new JSeparator(SwingConstants.HORIZONTAL),gbc);
  }

  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name, Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }
}
```

Projekt: FragenTester
Paket: database
Klasse : AllQuestionData.java

AllQuestionData.java

```
package database;

/**
 *
 * A Bag of all Values of an Question
 * Useful for Load/Save Questions
 *
 */
public class AllQuestionData{
  public int pos=0;
  public String id="";
  public String question = "";
  public String description = "";
  public String[] answers = new String[0];
  public boolean[] correct = new boolean[0];
  public int points=0;
  public String type="";
  AllQuestionData(){};
  public AllQuestionData(int po, String i, String f, String[] a, boolean[] c, String d, int p, String t)
  { pos=po; id=i; question=f; answers=a; correct=c; description=d; points=p; type=t; }
}
```

Projekt: FragenTester
Paket: database
Klasse : DataBase.java

DataBase.java

```
package database;

import java.awt.event.ActionListener;
import java.util.Map;
/**
 *
 * @author ToPeG
 *
 * Die Datenbank enthält Die nötigen Werte,
 * um Fragen stellen zu können,
 * die richten Antworten darauf zu kennen und
 * sich Benutzerdaten Merken zu können.
 *
 * Über setSource wird die Datenbank initalisiert
 * Die übergebenen wertepaaar sind Implementationsspezifisch
 * und müssen in der Klasse erklährt,
 * werden die das Interface implementiert
 *
 */
public interface DataBase {
  //#########################################################

  /**
   *
   * @return type of the database
   */
  public String getType();

  /**
   * return a map of configurtion of the datbase
   * @return
   */
  public Map<String,?> getSource();

  /**
   *
   * Configure the database
   * Set file or network source etc.
   *
   * @param dbs
   * @throws Exception
   */
  public void setSource(Map<String,?> dbs) throws Exception;

  //#########################################################

  /**
   * @param b set true to exclude all single questions
   */
  public void excludeSingle(boolean b);
```

```
/**
 * @param b set true to exclude all multi questions
 */
public void excludeMulti(boolean b);

/**
 * @param b set true to exclude all regular expression questions
 */
public void excludeRegexp(boolean b);

/**
 *
 * @return true if single Questions are excluded
 */
public boolean isSingleExcuded();

/**
 *
 * @return true if multi Questions are excluded
 */
public boolean isMultiExcuded();

/**
 *
 * @return true if regexp Questions are excluded
 */
public boolean isRegexpExcuded();

//#########################################################

/**
 * set the range of the selected question for the next test
 * @param min minimal range (>=0 and <=Questions)
 * @param max maximal range (>=0 and <=Questions)
 */
public void setRange(int mn, int mx);

/**
 * get the minimal value of the selected Questions
 * @return minimal value
 */
public int getMin();

/**
 * get the maximal value of the selected Questions
 * @return maximal value
 */
public int getMax();

/**
 * set the minimal value of the selected Questions
 * ignored if min>max or min<0 or min>questions
 * @param min
 */
public void setMin(int min);
```

```
/**
 * set the maximal value of the selected Questions
 * ignored if min>max or max<0 or max>questions
 * @param min
 */
public void setMax(int max);

//#########################################################

/**
 * @return absolute position of the active Question
 */
public int getPosition();

/**
 * @return relative (getPosition()-min) position of the active Question
 */
public int getRelativePosition();

/**
 * @param pos set the absolute position of the active Question
 */
public void setPosition(int pos);

//#########################################################

/**
 * delete History
 */
public void deleteHistory();

/**
 * reset Question Position
 * set the Postition of the active Question to the minimal value
 */
public void resetPosition();

/**
 * Cleanup Database
 * Remove all User Answers,
 * set the Question status to not answered
 */
public void deleteUserAnswers();

//#########################################################

/**
 * @return Size of the Database
 */
public int size();

//#########################################################
```

```java
/**
 * @param id id of a Question
 * @return true if the Question with this id exists
 */
public boolean QuestionExists(String id);

/**
 *
 * @param pos absolute Position in the Database
 * @return true if ther is an question at this position
 */
public boolean QuestionExists(int pos);

/**
 *
 * @param id ID of an Question
 * @return absolute Position of the Question or -1 if there is no Question with this id
 */
public int getQuestionPosition(String id);

//############################################################

/**
 * Add an Question to the Database
 * @param question
 */
public void addQuestion(Question question);

/**
 *
 * @return the active question
 */
public Question getQuestion();

/**
 *
 * @param pos
 * @return Question at the position or null if the Position not exists
 */
public Question getQuestion(int pos);

/**
 *
 * @param id
 * @return Question with this id or null if there is no Question with this id
 */
public Question getQuestionById(String id);

/**
 *
 * @return List of all Questions in the Database
 */
public Question[] getAllQuestions();
```

```
    /**
     * @return List of all questions between min and max and are not excluded
     */
    public Question[] getAllPossibleQuestions();

    /**
     * Sort Question by ID
     */
    public void sortQuestions();

    /**
     *
     * @return true if the Questions ar sorted
     */
    public boolean isSortedQuestions();

    /**
     *
     * @return true if the Answers are Sorted
     */
    public boolean isSortedAnswers();

    /**
     * Sort Answers of all Questions
     */
    public void sortAnswers();

    /**
     * Random Questions
     * That change the Order of the Questions
     * Random should only done between min and max
     */
    public void randomQuestions();

    /**
     * Random Answers
     * That change the order of the Answers
     */
    public void randomAnswers();

    //#########################################################

    /**
     * is the Position of the question between min and max and not excluded
     */
    public boolean isPossibleQuestion(int pos);

    /**
     * is the position of the Question between min and max and match the selected types
     * @param pos Position of the question
     * @param single Could be an Single Answered Question
     * @param multi  Could be an Multi Answered Question
     * @param regexp Could be an regexp Question
     * @return true if all matched
     */
    public boolean isPossibleQuestion(int pos, boolean single, boolean multi, boolean regexp);
```

```java
/**
 * count all Questions that match the Parameter?
 * @param min Absolute position in DB
 * @param max Absolute position in DB
 * @param single could be single
 * @param multi could be multi
 * @param regexp could be regexp
 * @return count of the Questions
 */
public int countAllPossibleQuestions(int min, int max, boolean single, boolean multi, boolean regexp);

/**
 * count all questions that match the predefined parameter excluding min and max
 * @param min Absolute position in DB
 * @param max Absolute position in DB
 * @return count of the Questions
 */
public int countAllPossibleQuestions(int min, int max);

/**
 * count all questions that match the predefined parameter
 * @return count of the Questions
 */
public int countAllPossibleQuestions();
/**
 * see above
 * @return
 */
public int countPossibleQuestion();

//#########################################################


/**
 * Start an Test
 */
public void start();

/**
 * next question
 */
public void next();

/**
 * previous question
 */
public void previous();

/**
 * First Question
 */
public void first();
```

```
  /**
   * Last question
   */
  public void last();

  //#########################################################

  /**
   * count all user Selections
   */
  public int getUserScore();

  /**
   * maximal possible Score
   */
  public int getScore();

  public void addActionListener(ActionListener l);
  public void removeActionListener(ActionListener l);
}
```

Projekt: FragenTester
Paket: database
Klasse : Question.java

Question.java

```
package database;

/**
 *
 * @author ToPeG
 *
 * Interface für eine Frage.
 * Sie enthält als kinder die mölichen antworten.
 *
 * single Type Question:
 * Es gibt nur eine wahre Antwort auf die Frage
 *
 * multi Type Question
 * Es gibt mehrer wahre Antworten auf die Frage.
 * Est wenn alle korrekt ausgewählt sind ist wie Frage richtig beantwortet
 *
 * regexp type Question
 * Sie antwort mus vom Nutzer eingegeben werden
 * und wird gegen einen Regulähren Ausdruck geprüft.
 */
public interface Question extends Comparable<Question>{

  public enum Type {single, multi, regexp};

  /**
   *
   * @return true if this Question is the active one
   */
  public boolean isActive();

  /**
   * Set the question active
   * @param active
   */
  public void setActive(boolean active);

  /**
   *
   * @return the Question Text
   */
  public String getQuestionText();

  /**
   *
   * @return the Description Text or "" if no description is set.
   */
  public String getDescriptionText();
```

```
/**
 *
 * @return The type of the Question
 */
public Type getType();

/**
 *
 * @return returns the Questiontype as String "single" or "multi" or "regexp"
 */
public String getTypeAsString();

/**
 *
 * @return a list of the possible Answers of this Question
 */
public Answer[] getAnswers();

/**
 *
 * @return the Answertexts separated by "\n"
 */
public String[] getAnswersAsString();

/**
 * If an answer is correct then true else false
 *
 * @return List of true/false
 */
public boolean[] getTrueAnswers();

/**
 *
 * @return the texts of true Answers separated by "\n"
 */
public String getTrueAnswersAsString();

/**
 *
 * @return the points of this question
 */
public int getPoints();

/**
 *
 * @return QuestionID
 */
public String getId();

/**
 * @see AllQuestionData
 * @return
 */
public AllQuestionData getAll();
```

```
    /**
     *
     * @return true if Question is answered
     */
    public boolean isAnswered();

    //#############################################################

    /**
     * Sort answers
     * The Sequence is the same as in the database
     */
    public void sortAnswers();

    /**
     * random the Answers for this question
     */
    public void randomAnswers();

    /**
     *
     * @return true if the answers ar sorted
     */
    public boolean isSortedAnswers();

    //#############################################################

    /**
     * @return  getPoints() if User answered correct or 0 if not
     */
    public int getUserPoints();

    /**
     *
     * @param useranswers List of useranswers
     */
    public void setUserAnswers(boolean[] useranswers);

    /**
     *
     * Only user for Question type "regexp"
     * @param useranswer Set the user typed text
     */
    public void setUserAnswerString(String useranswer);

    /**
     *
     * @return list of the answers the user set.
     */
    public boolean[] getUserAnswers();

    /**
     * Only user for Question type "regexp"
     * @return the user typed text
     */
    public String getUserAnswerString();
```

```
  /**
   * Delete all user set Values
   */
  public void deleteUserAnswers();

  /**
   *
   * @return true if user answers are correct
   */
  public boolean testUserAnswers();

  //#############################################################

  public int compareTo(Question q);

  /**
   *
   * @return position of the question
   */
  public int getPosition();
}
```

Projekt: FragenTester
Paket: database
Klasse : Answer.java

Answer.java

```
package database;

/**
 *
 * @author ToPeG
 *
 * Eine Antwort kann richtig oder falsch sein.
 * Es gibt nur einen Unterschied zwischen selktiebaren Antworten
 * und Antworten, bei denen man Text eingeben muss.
 */
public interface Answer extends Comparable<Answer>{

  public String getText();
  public database.Question.Type getType();

  //#################################################################

  /**
   * Is this Answer correct for the question
   * @return true if the answer is correct
   */
  public Boolean isOK();

  /**
   *
   * @return true if user answered the question
   */
  public boolean isAnswered();

  /**
   * reset the UserSelections in this answer
   */
  public void resetUser();

  //#################################################################

  /**
   * This Method is not used when the question type is "regexp"
   * then it should return "false"
   * @return true if user has answered the Question and selected this Answer
   */
  public boolean getUserSelection();

  /**
   * This Method is not used when the question type is "regexp"
   * @param selection The User select (true) or select not (false) this Answer
   */
  public void setUserSelection(boolean selection);
```

```
  /**
   * This Method is only used when the question type is "regexp"
   * else it should return "";
   * @param text The user set this text for the Question
   */
  public void setUserText(String text);

  /**
   * Returns "" if no text is set else returns the User Text
   * This Method is only used when the question type is "regexp"
   * @return The text set by User
   */
  public String getUserText();

  /**
   * Has the user select the correct answer?
   *
   * if the QuestionType is "regexp",
   * it will test the AnswerText as Regular expression
   * on the text set by user.
   *
   * @return true if the User set the Answer Correct
   */
  public boolean isUserOK();

  //#############################################################

  /**
   * @return The position off the Question
   */
  public int getPosition();

  public int compareTo(Answer a);
}
```

Projekt: FragenTester
Paket: databasefile
Klasse : DataBase.java

DataBase.java

```
package databasefile;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import database.Question;
import database.Question.Type;

/**
 *
 * @author ToPeG
 *
 * Implementaion einer Datei basiert Datenbank
 * es können xml formatierte Daten:
 *
 * ****************************************************************
 * <?xm version="1.0" encoding="UTF-8" standalone="yes" ?>
 * <!DOCTYPE questions [
 *   <!ELEMENT questions   (question+)>
 *   <!ELEMENT question    (nr, text, answers, explanation? )>
 *   <!ELEMENT nr          (#PCDATA)>
 *   <!ELEMENT text        (#PCDATA)>
 *   <!ELEMENT answers     (answer+)>
 *   <!ELEMENT answer      (#PCDATA)>
 *   <!ELEMENT explanation (#PCDATA)>
 *   <!ATTLIST question points CDATA                      #REQUIRED>
 *   <!ATTLIST answer   status ( correct | wrong | regexp) #REQUIRED>
 *   <!ATTLIST answers  type   ( single | multi | string ) #IMPLIED>
 * ]>
 * <questions>
 *   <question points="5">
 *     <nr>1</nr>
 *     <text><![CDATA[Ist das eine Frage?]]></text>
 *     <answers type="string">
```

```
 *       <answer status="singe">
 *         <answer status="correct"><![CDATA[ja]]></answer>
 *         <answer status="wrong"><![CDATA[nein]]></answer>
 *       </answer>
 *     </answers>
 *     <explanation><![CDATA[Das ist eine Unsinnige Erkährung für eine unsinnige Frage]]></explanation>
 *   </question>
 *   <question points="5">
 *     <nr>2</nr>
 *     <text><![CDATA[gib einen Text ein]]></text>
 *     <answers type="string">
 *       <answer status="regexp"><![CDATA[.]]></answer>
 *     </answers>
 *     <explanation><![CDATA[Es wird auf einen Regulähren ausdrk geprüft
 * in diesem fall wird einfach geschaut ob ein Zeichen in der Antwort ist]]></explanation>
 *   </question>
 * </questions>
 * ********************************************************************
 *
 * oder auch PlainText formatierte Fragen:
 * ********************************************************************
 * QUESTION 1:
 * Ist das eine Frage?
 * A.  ja
 * B.  nein
 *
 * Answer: A
 * Points: 1
 *
 * Explanation:
 * Das ist eine Unsinnige Erkährung für eine unsinnige Frage
 *
 * QUESTION 2:
 * gib einen Text ein
 * Answer: .
 * Points: 4
 *
 * Explanation:
 * Es wird auf einen Regulähren Ausdruck geprüft
 * in diesem fall wird einfach geschaut ob ein Zeichen in der Antwort ist
 * ********************************************************************
 * gelesen werden.
 */
public class DataBase implements database.DataBase{
  private ArrayList<Question> questions = new ArrayList<Question>();
  private ArrayList<Integer> history = new ArrayList<Integer>();
  private int position=0;
  private int min=0;
  private int max=1;
  private boolean excludesingle=false;
  private boolean excludemulti=false;
  private boolean excluderegexp=false;
  private boolean sorted=true;
  private String file="";
  private ArrayList<ActionListener> action = new ArrayList<ActionListener>();
```

```
  public DataBase(String f, String type)
  {
    file=f;
    try{
      loadFile(f,type);
    }
    catch(IOException e){}
  }

  public DataBase(){}

  public void addQuestion(Question q)
  { questions.add(q); }

  //#########################################################

  public void excludeSingle(boolean b){ excludesingle=b; }
  public void excludeMulti(boolean b){ excludemulti=b; }
  public void excludeRegexp(boolean b){ excluderegexp=b; }
  public boolean isSingleExcuded(){ return excludesingle; }
  public boolean isMultiExcuded(){ return excludemulti; }
  public boolean isRegexpExcuded(){ return excluderegexp; }

  //#########################################################

  public void setRange(int mn, int mx)
  {
    if(mn>-1 && mn<size()) min=mn;
    if(mx>0 && mx<size()) max=mx;
    if(min>max){ int m =min; min=max; max=m; }
  }

  public int getMin(){ return min;}
  public int getMax(){ return max;}
  public void setMin(int m){ if(m>-1 && m<size() && m < max) min=m;}
  public void setMax(int m){ if(m>0 && m<size() && m > min) max=m;}

  //#########################################################

  public int getPosition(){ return position; }
  public int getRelativePosition(){ return countAllPossibleQuestions(min,position); }
  public void setPosition(int pos)
  {
    if(isPossibleQuestion(pos))
    {
      if(getNextQuestionPositionTo(pos) >= max || pos>=max)
        doAction("DBQLAST",this);
      else if(getPreviousQuestionPositionTo(pos+1) <= min || pos<=min)
        doAction("DBQFIRST",this);
      else doAction("DBQMIDDLE",this);
      history.add(position);
      getQuestion().setActive(false);
      getQuestion(pos).setActive(true);
      position=pos;
    }
  }
```

```
public String getType(){ return "file"; }
public Map<String, ?> getSource()
{
  HashMap<String,String> map = new HashMap<String,String>();
  map.put("file",file);
  return map;
}

//#########################################################

public void deleteHistory(){ history = new ArrayList<Integer>(); }
public void resetPosition()
{
  for(Question q:questions) q.setActive(false);
  setPosition(getNextQuestionPositionTo(min));
}
public void deleteUserAnswers(){ for(Question q: questions) q.deleteUserAnswers(); }

//#########################################################
public int size(){ return questions.size();}
//#########################################################

public boolean QuestionExists(String id)
{ return getQuestionPosition(id)>-1? true: false;}

public boolean QuestionExists(int pos)
{ return (pos >-1 && pos<questions.size())? true: false; }

public int getQuestionPosition(String id)
{
  for(int i=0; i<questions.size();i++)
    if(questions.get(i).getId().equals(id)) return i;
  return -1;
}

//#########################################################

public Question getQuestion()
{ return getQuestion(position); }

public Question getQuestion(int pos)
{ return QuestionExists(pos)? questions.get(pos): null; }

public Question getQuestionById(String id)
{ return getQuestion(getQuestionPosition(id)); }

public Question[] getAllQuestions()
{
  ArrayList<Question> q=new ArrayList<Question>();
  for(int i=min; i<=max; i++) q.add(getQuestion(i));
  Collections.sort(q);
  return q.toArray(new Question[q.size()]);
}
```

```
  public Question[] getAllPossibleQuestions()
  {
    ArrayList<Question> q=new ArrayList<Question>();
    for(int i=min; i<=max; i++) if(isPossibleQuestion(i)) q.add(getQuestion(i));
    //Collections.sort(q);
    return q.toArray(new Question[q.size()]);
  }

  public void sortQuestions()
  {
    Collections.sort(questions);
    sorted=true;
  }

  public boolean isSortedQuestions(){return sorted; }
  public boolean isSortedAnswers()
  {
    boolean s=true;
    for(Question q:questions) if(!q.isSortedAnswers()) s=false;
    return s;
  }

  public void sortAnswers()
  { for(Question q:questions) q.sortAnswers(); }

  public void randomQuestions()
  {
    Random rn = new Random();
    for(int p1=min; p1<max; p1++)
    {
      int p2=(int)(rn.nextDouble()*(max-min))+min;
      Question a=questions.get(p1);
      questions.set(p1,questions.get(p2));
      questions.set(p2,a);
    }
    sorted=false;
  }

  public void randomAnswers()
  { for(Question q:questions) q.randomAnswers(); }

  //#########################################################
  public boolean isPossibleQuestion(int pos)
  { return isPossibleQuestion(pos,!excludesingle,!excludemulti,!excluderegexp); }

  public boolean isPossibleQuestion(int pos, boolean single, boolean multi, boolean regexp)
  {
    if( QuestionExists(pos) &&
        (
          (single && questions.get(pos).getType()==Type.single) ||
          (multi  && questions.get(pos).getType()==Type.multi) ||
          (regexp && questions.get(pos).getType()==Type.regexp)
        )
      )return true;
    return false;
  }
```

```
public int countAllPossibleQuestions(int min, int max, boolean single, boolean multi, boolean regexp)
{
  int ret=0;
  for(int i=min; i<=max; i++)
  { if(isPossibleQuestion(i,single,multi,regexp)) ret++; }
  return ret;
}

public int countAllPossibleQuestions(int min, int max)
{ return countAllPossibleQuestions(min, max, !excludesingle,!excludemulti,!excluderegexp); }

public int countAllPossibleQuestions()
{ return countAllPossibleQuestions(min, max, !excludesingle,!excludemulti,!excluderegexp); }

public int countPossibleQuestion()
{ return countAllPossibleQuestions(min, position, !excludesingle,!excludemulti,!excluderegexp); }

private int getNextQuestionPositionTo(int pos)
{
  for(; pos<questions.size(); pos++) if(isPossibleQuestion(pos)) break;
  return pos;
}

private int getPreviousQuestionPositionTo(int pos)
{
  for(int ret=pos-1; ret>min; ret--) if(isPossibleQuestion(ret)) return ret;
  if(isPossibleQuestion(min)) return min;
  return pos;
}

//#########################################################
public void start()
{
  deleteHistory();
  deleteUserAnswers();
  resetPosition();
  setPosition(getNextQuestionPositionTo(position));
}

public void next()
{
  //System.out.println("HISTORY A: "+history);
  int pos=getNextQuestionPositionTo(position+1);
  if(pos <= max) setPosition(pos);
  //System.out.println("HISTORY NEXT: "+history);
}

public void previous()
{
  //System.out.println("HISTORY C: "+history);
  int pos=getPreviousQuestionPositionTo(position);
  if(pos >= min) setPosition(pos);

  //System.out.println("HISTORY PREV: "+history);
}
```

```
public void first()
{
  setPosition(getNextQuestionPositionTo(min));
  //doAction("DBQFIRST",this);
  //System.out.println("HISTORY FIRST: "+history);
}

public void last()
{
  //System.out.println("HISTORY G: "+history);
  setPosition(getPreviousQuestionPositionTo(max+1));
  //doAction("DBQLAST",this);
  //System.out.println("HISTORY LAST: "+history);
}


public int getUserScore()
{
  int score=0;
  for(int i=min; i<=max; i++) score+=questions.get(i).getUserPoints();
  return score;
}

public int getScore()
{
  int score=0;
  for(int i=min; i<=max; i++) score+=questions.get(i).getPoints();
  return score;
}

public void loadDB(String f) throws IOException
{
  questions.clear();
  String s=main.Tools.slurpFile(f);
  if(s==null) throw new IOException("No Data");
  //System.out.println(s);

  String[] data=s.split("(QUESTION|Question|question)\\s*");

  //System.out.println(data[1]);

  int pos=0;
  for(String q: data)
  {
    if(!q.matches("\\A[\r\n\\s]*\\z"))
    {
      String[] l=q.split("\\s*:\\s*?\n?",2);
      int score=-1;

      Type type=null;

      String id=l[0];
      //System.out.println("Parse Frage: "+id);
      q=l[1];
```

```
        l=q.split("(Answer|ANSWER|answer)\\s*:\\s*",2);

        String qtext="";
        boolean noregexp=false;
        for(String ll: l[0].split("\n"))
        {
          //System.out.println("-->"+ll);
          if(!ll.matches("\\A\\p{Upper}\\..+"))
          {
            //System.out.println("---->"+ll);
            qtext+=ll+"\n";
          }
          else
          { noregexp=true; }
        }
        //System.out.println(qtext);
        q=l[1];

        HashMap<Integer,Integer> answers_unique = new HashMap<Integer,Integer>();
        ArrayList<String> answers = new ArrayList<String>();
        if(noregexp)
        {
          int i=-1;
          for(String ll: l[0].split("\n"))
          {
            //System.out.println("-->"+ll);
            if(ll.matches("\\A\\p{Upper}\\..+"))
            {
              int key=ll.charAt(0);
              if(answers_unique.get(key) == null)
              {
                i++;
                //System.out.println("ID: "+id+" => CHANGE: "+i+" -> "+(char)key);
                answers_unique.put(key,i);
              }
              while(i>=answers.size()) answers.add("");
              if( answers.get(i) == null ) answers.add(i,"");

              String[] a=ll.split("\\.\\s",2);
              if(a.length>1) answers.set(i,answers.get(i)+a[1]+"\n");
            }
          }
        }
        else type=Type.regexp;
        l=null;

        String[] ll=q.split("[\r\n\\s]*Points\\s*:?\\s*");
        if(ll.length>1)
        {
          l=ll[1].split("[\r\n\\s]*Explanation\\s*:?\\s*");
          score=Integer.parseInt(l[0]);
          //System.out.println("ID: "+id+" => READ POINTS:("+score+")");
          l[0]=ll[0];
        }
        else
        {
```

```
          //System.out.println("ID: "+id+" => NO POINTS, USING:("+score+")");
          l=q.split("[\r\n\\s]*Explanation\\s*:?\\s*");
        }
        boolean[] correct=new boolean[answers.size()];
        if(type == null)
        {
          String[] c=l[0].split("\\s*,\\s*");
          if(score<0)
          {
            score=c.length;
            //System.out.println("ID: "+id+" => GENRATE POINTS:("+score+")");
          }
          if(c.length>1) type=Type.multi;
          //if(type == Type.multi) System.out.println(id+" ==> "+l[0]);
          else if(c.length == 1) type=Type.single;
          for(int i: answers_unique.keySet())
          {
            correct[answers_unique.get(i)]=false;
            for(String cc: c)
            {
              if(i == cc.charAt(0))
              {
                correct[answers_unique.get(i)]=true;
                break;
              }
            }
          }
        }
        else
        {
          answers = new ArrayList<String>();
          answers.add(l[0]);
          correct = new boolean[]{false};
          score=5;
        }
        String desc=(l.length>1)?l[1]:"";


        addQuestion(new databasefile.Question(pos,id,score,qtext,answers.toArray(new String[answers.size()]),correct,type,desc));
        pos++;
      }
    }
    if(size()==0) throw new IOException("No Questions in File");
    setRange(0,size()-1);
    setPosition(0);
    deleteHistory();
  }

  public void loadXDB(File f) throws IOException
  {
    questions.clear();
    Document tree=null;
    try
    { tree = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(f); }
    catch (Exception e)
    { throw new IOException("Could not parse XMLFile"); }
```

```
    if(tree != null)
    {
      NodeList fragen = tree.getElementsByTagName("question");
      for(int i=0; i<fragen.getLength();i++)
      {
        Element frage = (Element)(fragen.item(i));

        // Typ der Frage bestimmen
        String stype=frage.getElementsByTagName("answers").item(0).getAttributes().getNamedItem("type").getNodeValue();
        Type typ = Type.regexp;
        if(stype.contains("single"))
        {typ = Type.single; }
        else if(stype.contains("multi"))
        { typ = Type.multi; }
        //System.out.println("IN-TYPE: "+stype+ " --> "+typ);

        // "Name" der Frage
        String id = frage.getElementsByTagName("nr").item(0).getTextContent();

        // Anzahl der möglichen Punkte, bei richtiger Antwort
        int punkte = Integer.parseInt(fragen.item(i).getAttributes().getNamedItem("points").getNodeValue());

        // Fragenblock lesen
        String frag = frage.getElementsByTagName("text").item(0).getTextContent();

        // Mögliche Antworten lesen
        NodeList antworten = ((Element)(frage.getElementsByTagName("answers").item(0))).getElementsByTagName("answer");
        database.Answer[] antw = new databasefile.Answer[antworten.getLength()];
        for(int ii=0; ii<antworten.getLength();ii++)
        {
          Node an=antworten.item(ii);
          String txt=an.getTextContent();
          boolean wahr=false;
          NamedNodeMap attr = an.getAttributes();
          Node stat=attr.getNamedItem("status");
          if(stat.getNodeValue().contains("correct")) wahr=true;
          if(typ == Type.regexp) wahr=true;
          antw[ii]=new databasefile.Answer(ii,txt,wahr,typ);
        }

        // Erklährungsblock lesen

        String expl="";

        if(frage.getElementsByTagName("explanation") != null && frage.getElementsByTagName("explanation").item(0) != null)
        { expl = frage.getElementsByTagName("explanation").item(0).getTextContent(); }

        addQuestion(new databasefile.Question(i,id,punkte,frag,antw,typ,expl));
      }
    }
    if(size()==0) throw new IOException("No Questions in File");
    setRange(0,size()-1);
    setPosition(0);
    deleteHistory();
  }
```

```
  public void addActionListener(ActionListener l){action.add(l);}
  public void removeActionListener(ActionListener l){action.remove(l);}
  private void doAction(String name,Object o)
  { for(ActionListener l:action) l.actionPerformed(new ActionEvent(o, 0, name)); }

  private void loadFile(String file,String type) throws IOException
  {
    if(type.equals("xdb"))
      loadXDB(new File(file));
    else if(type.equals("db"))
      loadDB(file);
    else new Exception("Could not verify Filetype! ("+type+")");
  }

  /**
   * Possible paramter
   *
   * file => File in witch the Questions are stored
   * type => Type of the file
   *         "db" if the file is palintext
   *         "dbx" if the file ist xml
   *
   */
  public void setSource(Map<String, ?> dbs) throws Exception
  {
    if(dbs.get("file")!=null && !dbs.get("file").equals(""))
    {
      String file = (String)dbs.get("file");
      if(dbs.get("type") != null && !dbs.get("type").equals(""))
        loadFile(file,(String)dbs.get("type"));
      else
      {
        if(((String)dbs.get("file")).matches(".+\\.xdb$"))
          loadFile(file,"xdb");
        else if(((String)dbs.get("file")).matches(".+\\.db$"))
          loadFile(file,"db");
        else
          throw new Exception("Could not identify Filetype!");
      }
    }
    else
      throw new Exception("No File Specified!");
  }
}
```

Projekt: FragenTester
Paket: databasefile
Klasse : Question.java

Question.java

```
package databasefile;

import java.util.Arrays;
import java.util.Random;

import database.AllQuestionData;

/**
 * @author ToPeG
 */
public class Question implements database.Question{
  private String id;
  private String question;
  private String description;
  private database.Answer[] answers;
  private int ponints;
  private Type type;
  private int position=-1;
  private boolean sorted=true;
  private boolean active=false;

  public Question (int pos, String inid, Integer p, String f, database.Answer[] a, Type t, String b)
  {
    position=pos;
    id=inid;
    ponints=p;
    question= f;
    description = b;
    answers = a;
    type = t;
  }

  public Question (int pos,String inid, int p, String f, String[] a, boolean[] b, Type t, String d)
  { this(pos, inid, p, f, StringArraytoAnswer(a,b,t), t, d); }

  public Question (int pos, String id, int p, String f, String[] a, boolean[] b, String t, String d)
  { this(pos, id,p,f,a,b,StringToType(t),d);}

  public Question(AllQuestionData d)
  { this(d.pos, d.id, d.points, d.question, StringArraytoAnswer(d.answers,d.correct,StringToType(d.type)), StringToType(d.type), d.description); }

  public boolean isActive(){return active;}
  public void setActive(boolean a){active=a;}

  private static Type StringToType(String s)
  {
    if(s.equals("single")) return Type.single;
    else if(s.equals("multi")) return Type.multi;
    else return Type.regexp;
  }
```

```
  private static Answer[] StringArraytoAnswer(String[] a, boolean[] b, Type type)
  {
    Answer[] answers = new Answer[a.length];
    if( type != Type.regexp)
    {
      for(int i=0; i<answers.length; i++)
      {
        boolean bb=false;
        if(b!= null && b.length>i) bb=b[i];
        answers[i]= new Answer(i,a[i],bb,type);
      }
    }
    else
    {
      for(int i=0; i<answers.length; i++)
        answers[i]=new Answer(i,a[i],true,type);
    }
    return answers;
  }

  public String getQuestionText()
  { return this.question; }

  public String getDescriptionText()
  { return this.description; }

  public Type getType()
  { return this.type; }

  public boolean isAnswered()
  {
    for(database.Answer a: answers)  if(a.isAnswered()) return true;
    return false;
  }

  public void sortAnswers()
  {
    Arrays.sort(answers);
    sorted=true;
  }

  public void randomAnswers()
  {
    Random rn = new Random();
    for(int p1=0; p1<answers.length; p1++)
    {
      int p2=(int)(rn.nextDouble()*answers.length);
      database.Answer a=answers[p1];
      answers[p1]=answers[p2];
      answers[p2]=a;
    }
    sorted=false;
  }

  public boolean isSortedAnswers(){return sorted;}
```

```
public String getTypeAsString()
{ return this.type.toString(); }

public database.Answer[] getAnswers()
{ return this.answers; }

public String[] getAnswersAsString()
{
  String[] out = new String[this.answers.length];
  for( int i=0; i<this.answers.length; i++)
  { out[i]=this.answers[i].getText(); }
  return out;
}

public boolean[] getTrueAnswers()
{
  boolean[] out = new boolean[this.answers.length];
  for( int i=0; i<this.answers.length; i++)
  { out[i]=this.answers[i].isOK(); }
  return out;
}

public String getTrueAnswersAsString()
{
  String out = "";
  for( int i=0; i<this.answers.length; i++)
  { if(this.answers[i].isOK()) out+=this.answers[i].getText()+"\n"; }
  return out;
}

public int getPoints()
{ return this.ponints; }

public String getId()
{ return this.id; }

public AllQuestionData getAll()
{ return new AllQuestionData(position,id,question,getAnswersAsString(),getTrueAnswers(),description,ponints,getTypeAsString()); }

//############################################################
public int getUserPoints()
{
  if(testUserAnswers()) return getPoints();
  return 0;
}

public void setUserAnswers(boolean[] a)
{
  System.out.println("USER "+a);
  if(getType()!=Type.regexp)
  {
    for(int i=0; i<a.length; i++)
    { answers[i].setUserSelection(a[i]); }
  }
}
```

```
  public void setUserAnswerString(String a)
  {
    System.out.println("USER "+a);
    if(getType()==Type.regexp && answers != null)
    { answers[0].setUserText(a); }
  }

  public boolean[] getUserAnswers()
  {
    if(answers != null && isAnswered())
    {
      boolean[] ba=new boolean[answers.length];
      for(int i=0; i<answers.length; i++)
      { ba[i]=answers[i].getUserSelection();}
      return ba;
    }
    return null;
  }

  public String getUserAnswerString()
  { return (answers != null && isAnswered())? answers[0].getUserText(): ""; }

  public void deleteUserAnswers()
  {
    if(answers!=null)
    { for(database.Answer a: answers) a.resetUser(); }
  }

  public boolean testUserAnswers()
  {
    if(!isAnswered() ) return false;
    boolean is_ok = true;
    for(int i = 0; i < answers.length; i++)
    { if(!answers[i].isUserOK())is_ok=false; }
    return is_ok;
  }

  public int compareTo(database.Question q) { return position-q.getPosition(); }
  public int getPosition() {return position;}
}
```

Projekt: FragenTester
Paket: databasefile
Klasse : Answer.java

Answer.java

```
package databasefile;

import database.Question.Type;

/**
 *
 * @author ToPeG
 *
 */
public class Answer implements database.Answer{

  private String text;
  private boolean wahr;
  private Type type;
  private boolean usersel = false;
  private String usertxt = "";
  private boolean answered =false;
  private int position=-1;

  //#############################################################
  public Answer(int p, String a, Boolean w)
  {
    position=p;
    text=a;
    wahr=w;
    type=Type.regexp;
  }

  public Answer(int p, String a, Type r)
  {
    position=p;
    text=a;
    wahr=false;
    type=r;
  }

  public Answer(int p, String a, Boolean w, Type r)
  {
    position=p;
    text=a;
    wahr=w;
    type=r;
  }
  //#############################################################

  //#############################################################
  public String getText(){ return text; }
  public Type getType(){ return type;}
  public boolean getUserSelection(){ return usersel;}
  public void setUserSelection(boolean b){ usersel=b; answered=true;}
```

```
  public void setUserText(String t){usertxt=t; answered=true;}
  public String getUserText(){return usertxt;}
  public boolean isAnswered(){ return answered; }
  public void resetUser(){usersel=false; usertxt=""; answered=false;}
  //#############################################################

  public boolean isUserOK()
  {
    if(type != Type.regexp)
    { return usersel==wahr; }
    else
    { return usertxt.matches(text); }
  }

  public Boolean isOK()
  { return this.wahr; }

  public int compareTo(database.Answer a) {
    return position-a.getPosition();
  }
  public int getPosition() {return position; }
}
```