INF10

Arbeitsblatt: INF2

Name: **Kurznamen:**

Das Datenzugriffsobject (DAO) und Entwurfsmuster

Wir haben in einem früheren Praktikum das GUI für die Rangliste mittels dem GUI Builder entwickelt. Das funktioniert zwar sehr gut, der von NetBeans generierte Code ist jedoch recht komplex. In diesem Praktikum entwickeln wird das GUI nochmals mittels den Klassen und Layout-Manager, die wir in den letzten Vorlesungen kennengelernt haben. Gleichzeitig passen wir den Zugriff auf die Rangliste so an, dass diese als sogenanntes DAO umgesetzt wird.

Data Access Object (DAO, englisch für Datenzugriffsobjekt) ist ein Entwurfsmuster, das den Zugriff auf unterschiedliche Datenquellen (z. B. Datenbanken, Dateisystem) so kapselt, dass die Datenquelle ausgetauscht werden können, ohne dass der aufrufende Code gross geändert werden muss. Dadurch wird die eigentliche Programmlogik von technischen Details der Datenspeicherung befreit und flexibler einsetzbar.

Ein DAO setzt mindestens die 4 sog. CRUD Operationen um:

- **C**reate: füge neuen Datensatz hinzu
- **R**ead: lese Datensätze: i.d.R. Methoden *read* (für alle) und *find* spezifische Datensätze
- **U**pdate: speichere geänderte Datensätze
- **D**elete: lösche Datensatz

Daneben:

- Connect: definiert zusätzliche Daten für die Speicherung (z.B. Filename)
- Load: lädt die Daten
- Save: Speichert die Daten

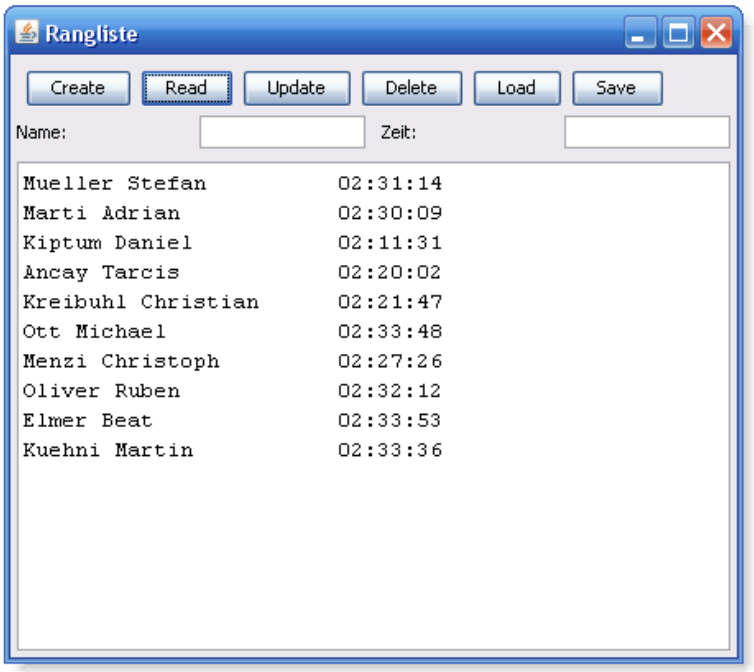
Entwurfsmuster

Ein DAO ist ein sogenanntes Entwurfsmuster. Das sind bewährte *Lösungsschablonen* für wiederkehrende Entwurfsprobleme sowohl in der Programmierung als auch in der Softwarearchitektur. Sie stellen wiederverwendbare Vorlagen zur Problemlösungen dar, die in einembestimmten Zusammenhang einsetzbar sind. (Wikipedia). Es gibt Hunderte solcher Entwurfsmuster, von denen jedoch nur ca. 30 wirklich wichtig sind. Das DOA ist eines davon.

Das beigefügte RanglisteDAO ist so ausgelegt, dass es die (Demo-)Daten im Hauptspeicher verwaltet. Wir werden es aber in einem späteren Praktikum so anpassen, dass die Daten vom Dateisystem geladen werden.

Aufgabe 1: GUI

Das GUI selber ist aus mehreren Panels mit unterschiedlichen Layout Managern aufgebaut. Versuchen Sie den beigefügten Code zu verstehen und ergänzen Sie ihn so, dass es so wie unten dargestellt aussieht.



Die einzelnen Knöpfe sollen die entsprechenden Methoden des DOAs aufrufen.

Aufgabe 2: Laufzeittests: Exception und Anzeige

Falls Fehler auftauchen, so soll das im DOA überprüft werden. Folgende Fehler sollen abgefangen werden.

- Create eines Teilnehmers der schon existiert
- Update mit fehlendem Namen oder keiner gültigen Zeit
- Delete mit fehlendem oder falschem Namen

Das soll nun im DAO überprüft werden und im Fehlerfall soll eine DAOException geworfen werden.

```
public class DAOException extends Exception {
      public DAOException(String reason) {
            super(reason);
      }
}
```

Alle *public* Methoden des DAOs müssen dafür um den entsprechenden DAOException ergänzt werden.

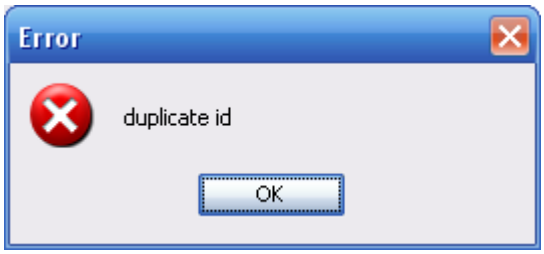
```
public void create(Teilnehmer teilnemer) throws DAOException {
       ...
      max++;
}
```

Hinweis:

Überprüfung von create

```
public void create(Teilnehmer teilnehmer) throws DAOException {
    if (find(teilnehmer.getName()) != null)
         throw new DAOException("Teilnehmer schon vorhanden");
    rangliste[max] = teilnehmer;
    max++;
}
```

Im GUI soll beim auslösen einer Exception dann eine Fehlermeldung anzeigen. Hinweis: es reicht ein catch am Schluss der actionListener Methode.




```
try {
  ...
}
catch (DAOException ex) {
     JOptionPane.showMessageDialog(this,ex.getMessage(),"Error",
    JOptionPane.ERROR_MESSAGE);
}
```

Abgabe

Praktikum: INF10.1

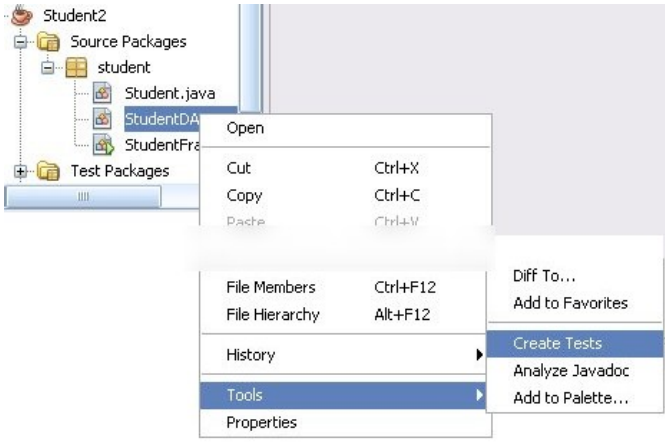
Filename: RanglisteDAO.java

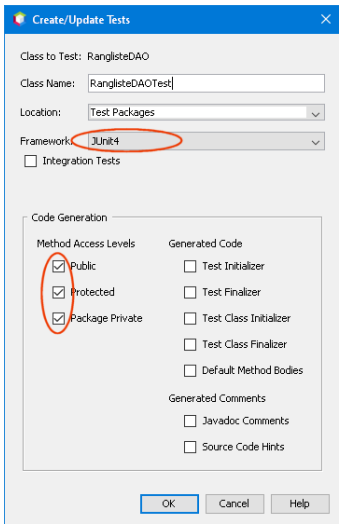
Aufgabe 3: Entwicklertests: Testfälle mit JUnit (optional)

Ob sich ein Programm korrekt verhält, kann mit JUnit überprüft werden JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist.

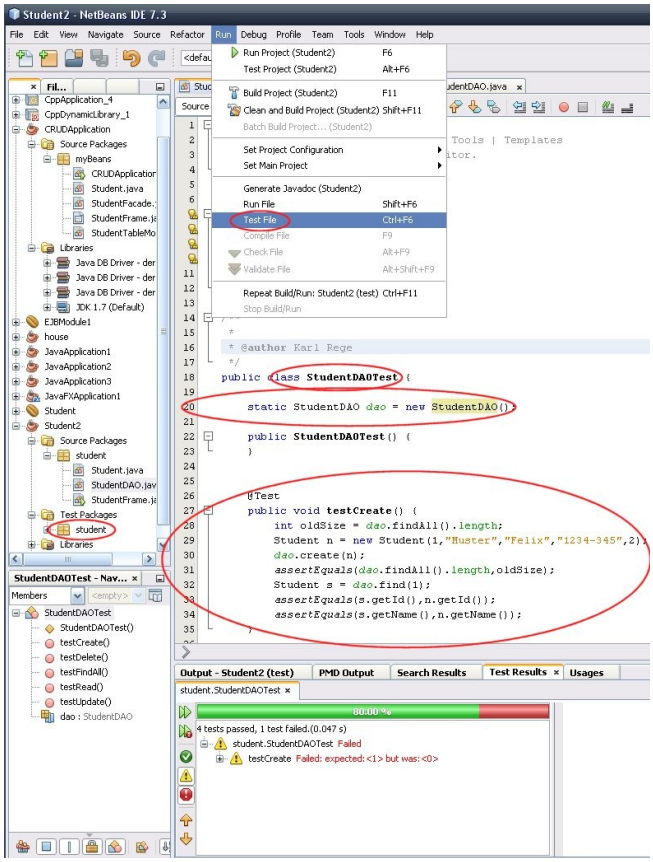
Dabei schreibt ein Programmierer zuerst einen automatisch wiederholbaren (JUnit-)Test und dann den zu testenden Code. Der Test ist selbst ein Stück Software und wird ebenso wie der zu testende Code programmiert. Wenn zu einem späteren Zeitpunkt ein anderer Programmierer den so entstandenen Code ändern möchte, so ruft er zuerst alle JUnit-Tests auf, um sich zu vergewissern, dass der Code vor seiner Änderung fehlerfrei ist. Dann führt er die Änderung durch und ruft die JUnit-Tests erneut auf. Misslingen diese, so weiss er, dass er selbst einen Fehler eingebaut hat, und muss ihn korrigieren. NetBeans unterstützt Sie bei der Entwicklung solcher Test.

Rechtsklick auf RanglisteDAO→ Tools→Create Tests





Es wird eine RanglisteDAOTest Klasse erzeugt und ein Test Package erstellt. Wählen Sie die Codegenerierung wie oben angegeben an; **wichtig: JUnit4 auswählen**. Im generierten Testklassengerüst (RanglisteDAOTest) muss jetzt noch der eigentlicheTestcode ergänzt weden. Zu jeder Methode in der ursprünglichen, zu testenden Klasse wird eine entsprechende *test*Methode generiert, z.B. create → testCreate. Nachdem der Test implementiert ist, kann er mit Run → Test File ausgeführt werden. Unten erscheint die Auswertung der Tests als rot-grüner Balken und weiteren Informationen im Fehlerfall.



Hinweis:

falls ***fail*** erreicht wird, dann ist der Test schiefgelaufen. Das kann ausgenutzt werden, um zu überprüfen, ob die Exceptions wie erwartet geworfen werden.

```
@Test
 public void testCreate() {
     RanglisteDAO dao = new RanglisteDAO();
     // neuer Name, sollte klappen
     Teilnehmer tn = new Teilnehmer("Felix Muster", 2, 31, 14);
     try {
         dao.create(tn);
     } catch (DAOException e) {
         fail("Exception"+e);
     }
     // name existiert schon; sollte Exception werfen
     tn = new Teilnehmer("Mueller Stefan", 2, 31, 14);
     try {
         dao.create(tn);
         fail("Exception erwartet");
     } catch (DAOException e2) {
     }
 }
```

Aufgabe 4 (optional)

Grössere Datenmengen werden typischerweise in Datenbanken gespeichert. Die haben ein Gerüst vorgegeben, das den Zugriff auf eine SQLite Datenbank implementiert. Auch hier wird das DAO Pattern verwendet, so dass die obige Klasse einfach durch die neue, erweiterte ersetzt werden kann.

Hinweise:

https://www.javatpoint.com/java-jdbc
https://www.baeldung.com/java-jdbc