Übungsaufgabe 6: HTML-Notenspiegel

Als Voraussetzung für diese Aufgabe müssen Sie die Vorlesungsunterlagen bis Seite 5-31 nachbereitet und die zugehörigen Programmbeispiele nachvollzogen haben.
Ihr Unterverzeichnis `prog1-beispiele\Programme5\` *sollte danach mindestens folgende Programme enthalten:* `vererbung\Datum.java, vererbung\Termin.java, vererbung\OrtsTermin.java vererbung\TerminTest.java, local\IntList.java, local\ListVar.java.`

Nach dem Download und dem Entpacken von Aufgabe 6 sollten in Ihrem Arbeitsverzeichnis folgende Dateien hinzugekommen sein:

`BennoBeispiel.html`	Beispiel für einen Notenspiegel
`aufgabe6\package-info.java` **`aufgabe6\HtmlNotenspiegel.java`** **`aufgabe6\Leistung.java`** **`aufgabe6\LeistungsListe.java`**	Unterverzeichnis für Java-Quellcode, darin die Paketdokumentation und das noch unvollständige Java-Programm

Schritt 1: Vorbereitung

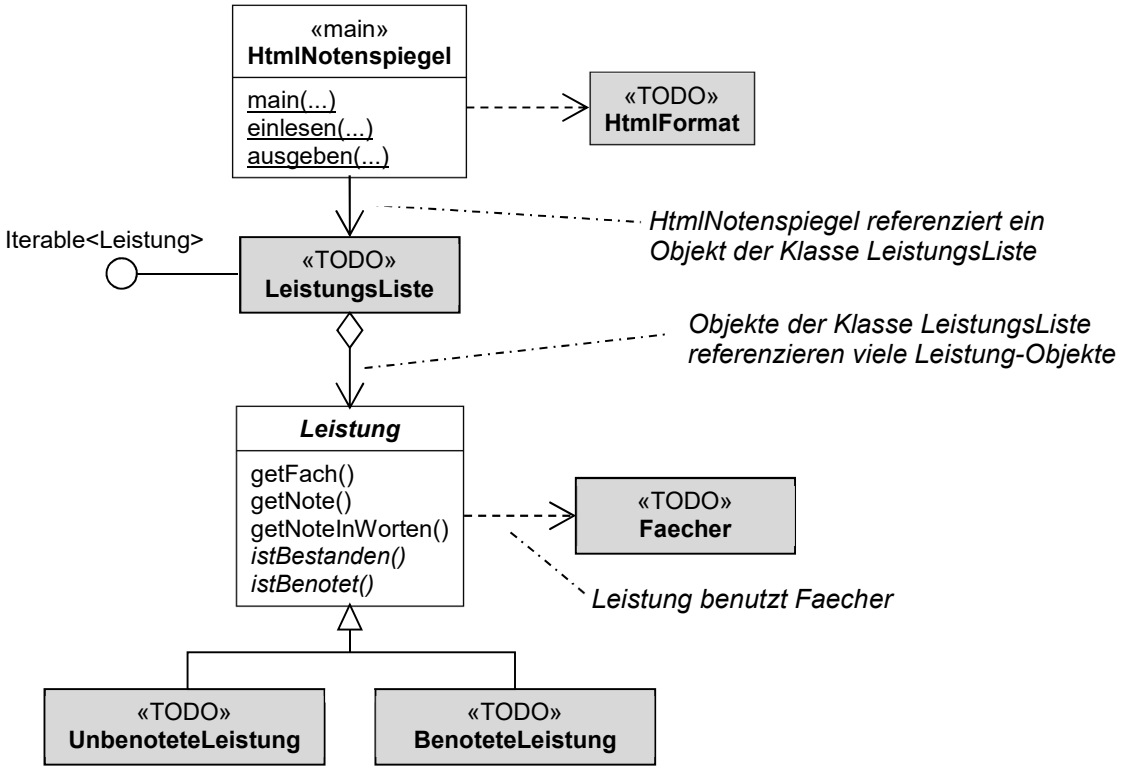
- Aktualisieren Sie Ihre Javadoc-Seiten:
  `ant doc`

  Beim Paket aufgabe6 kommen einige Fehlermeldungen, weil sie dieses Paket in Schritt 2 erst noch vervollständigen müssen. Die Meldungen können Sie ignorieren.

  Öffnen Sie im Browser alle Lesezeichen aus `Lesezeichen.html`.

  Lesen Sie die Spezifikation des zu erstellenden Programms auf der Javadoc-Seite des Pakets `aufgabe6`. Die Javadoc-Seite finden Sie im Browser-Tab "*Overview (Programmiertechnik 1)*" oder indem Sie die Datei `doc\index.html` mit dem Browser öffnen.

- Das folgende UML-Diagramm gibt Ihnen einen Überblick über die in Schritt 2 zu erstellenden Klassen:



Schritt 2: Programmierung

Erstellen Sie eine Utility-Klasse `aufgabe6.Faecher` mit folgenden Komponenten:

- einer privaten konstanten Klassenvariablen `FAECHER` vom Typ Array von Strings, die alle laut AIN-Studienplan des ersten Semesters erlaubten Fachnamen enthält
- einer öffentlichen Klassenmethode `istZulaessig()`, die prüft, ob ein gegebenes Fach im Studienplan vorgesehen ist (zur Signatur siehe den Aufruf der Methode in der Klasse `aufgabe6.Leistung`)

Erstellen Sie eine instanziierbare Klasse `aufgabe6.UnbenoteteLeistung` für Entitäten als Unterklasse von `aufgabe6.Leistung`:

- die Klasse soll eine konstante private Instanzvariable haben, in der gespeichert ist, ob die Leistung bestanden ist oder nicht
- die Klasse soll einen öffentlichen Konstruktor haben (die erforderliche Signatur können Sie dem `new`-Ausdruck in der Klasse `aufgabe6.HtmlNotenspiegel` entnehmen)
- die Klasse muss geerbte Methoden überschreiben. Welche das sind, können Sie der in Schritt 1 erstellten Javadoc-Beschreibung der Oberklasse `aufgabe6.Leistung` entnehmen.

Erstellen Sie eine instanziierbare Klasse `aufgabe6.BenoteteLeistung` für Entitäten als Unterklasse von `aufgabe6.Leistung`:

- die Klasse soll eine konstante private Instanzvariable vom Typ `aufgabe5.Note` zum Speichern der Note haben.
- die Klasse soll einen öffentlichen Konstruktor haben (die erforderliche Signatur können Sie dem `new`-Ausdruck in der Klasse `aufgabe6.HtmlNotenspiegel` entnehmen)
- die Klasse muss geerbte Methoden überschreiben. Welche das sind, können Sie der in Schritt 1 erstellten Javadoc-Beschreibung der Oberklasse `aufgabe6.Leistung` entnehmen.

  Verwenden Sie die entsprechende Instanzmethode der Wert-Klasse `aufgabe5.Note`, um zu entscheiden, ob die gekapselte Note als bestanden gilt.

  Halten Sie sich außerdem an die folgenden Notennamen der HTWG:

`"sehr gut"`	für Noten `1,0` bis `1,5` einschließlich
`"gut"`	für Noten `1,6` bis `2,5` einschließlich
`"befriedigend"`	für Noten `2,6` bis `3,5` einschließlich
`"ausreichend"`	für Noten `3,6` bis `4,0` einschließlich
`"nicht ausreichend"`	für Noten ab `4,1`

Erstellen Sie eine instanziierbare Klasse `aufgabe6.LeistungsListe` für Entitäten:

- Verwenden Sie das Vorlesungsbeispiel `local.IntList` aus Teil 5 als Vorlage. Passen Sie darin den Paket- und Klassennamen an, implementieren Sie die Schnittstelle `java.util.Iterator<Leistung>` statt `java.util.Iterator<Integer>` und stellen Sie den Typ der gespeicherten Werte von `int` auf `Leistung` um. Das Boxing in der Implementierung der `next`-Methode kann entfallen.

Erstellen Sie eine Utility-Klasse `aufgabe6.HtmlFormat` mit einer öffentlichen Klassenmethode `ausgeben`:

- Leiten Sie die Signatur der Klassenmethode aus dem Aufruf in `aufgabe6.HtmlNotenspiegel.ausgeben` ab.
- Leiten Sie die Implementierung der Klassenmethode aus der Beispielausgabe in der Datei `BennoBeispiel.html` ab. Sie können sich den Inhalt von `BennoBeispiel.html` mit einem Texteditor ansehen. Wenn Sie `BennoBeispiel.html` im Browser öffnen, sehen Sie die formatierte Darstellung des Inhalts. Verwenden Sie eine for-each-Schleife zum Ablaufen der übergebenen Leistungsliste und geben Sie den HTML-Text mit `printf` und `println` aus.

  Wenn Sie den Inhalt von `BennoBeispiel.html` etwas tiefer verstehen wollen, sind die folgenden Webseiten eine gute Hilfe:

  ```
  https://wiki.selfhtml.org/
  https://www.w3schools.com/html/default.asp
  ```

Schritt 3: Test und Qualitätssicherung

- Übersetzen Sie das Programm:

  `ant -Dpackage=aufgabe6 compile`

  Andere Möglichkeiten zum Aufruf von **`javac`** sind bei Aufgabe 5 beschrieben.
- Testen Sie das Programm mit verschiedenen Eingaben.

  Erstellen Sie dazu wieder Dateien mit Testeingaben im Stil von Aufgabe 3 und 4 (`aufgabe6-in1.txt` usw.) und betrachten Sie die Ausgabedateien mit dem Browser Firefox:

  `java -ea aufgabe6.HtmlNotenspiegel` ***`Vorname Nachname`***

  Die Option `-ea` sorgt dafür, dass die `assert`-Regeln in `Leistung` ausgewertet werden.

  Verwenden Sie im Firefox auch `rechte Maustaste->Quelltext anzeigen`, um Ihre HTML-Ausgabe zu prüfen.
- Prüfen Sie mit checkstyle, ob Sie Stilregeln verletzt haben. Bessern Sie nach, solange das Werkzeug Fehler meldet. Rufen Sie checkstyle im Arbeitsverzeichnis über ant auf:

  `ant -Dpackage=aufgabe6 style`
- Erstellen Sie Ihre endgültigen Javadoc-Seiten. Bessern Sie Ihre Javadoc-Kommentare nach, solange javadoc noch Fehler meldet:

  `ant doc`
- Prüfen Sie mit spotbugs, ob Ihr Programm problematischen Code enthält. Bessern Sie nach, solange das Werkzeug Probleme meldet. Rufen Sie spotbugs wie folgt über ant auf:

  `ant -Dpackage=aufgabe6 clean bugs`

  spotbugs schreibt seine Meldungen nicht auf den Bildschirm, sondern in eine Datei *`bugs.html`*, die Sie mit einem Webbrowser anschauen müssen.

Schritt 4: Abgabe

Den spätesten Abgabetermin finden Sie auf der Webseite der Lehrveranstaltung.

Geben Sie bitte erst ab, nachdem Sie Schritt 3 erledigt haben!
Die Werkzeuge checkstyle und spotbugs dürfen keine Fehler mehr melden!

- Führen Sie Ihre in Schritt 3 erstellten Testfälle `aufgabe6-in1.txt` usw. vor und zeigen Sie die Notenspiegel im Browser.
- Zeigen Sie ihre Javadoc-Seiten mit dem verlinkten Quellcode der Klassen.

Ergänzende Übungen für Schnellprogrammierer *(freiwillig, pro Spiegelpunkt 1 Bonuspunkt)*

- Seit Java 15 gibt es zusätzlich zu den String-Literalen `"..."` auch Textblöcke, die in dreifache Anführungszeichen `"""` eingeschlossen werden und über mehrere Zeilen gehen können.
  Lesen Sie dazu `https://docs.oracle.com/en/java/javase/15/text-blocks/`
  und nutzen Sie dann in Ihrer Klasse **`HtmlFormat`** Textblöcke, um die Ausgabe des HTML-Texts einfacher und übersichtlicher zu gestalten.

- Erstellen Sie eine Utility-Klasse **`TextFormat`** analog zu `HtmlFormat`, die einen Notenspiegel im tabellarischen Format nach dem Vorbild von Aufgabe 5 in einer Datei mit Endung `.txt` ausgibt.

  Bauen Sie die Verwendung der Klasse zusätzlich zu `HtmlFormat` in die Methode `aufgabe6.HtmlNotenspiegel.ausgeben` ein.

- Sorgen Sie dafür, dass in der verketteten Liste der Fachnoten keine Leistung doppelt vorkommt.

  Ändern Sie dazu `LeistungsListe.insert` so ab, dass nach dem Einfügen der neuen Leistung am Listenanfang in der restlichen Liste nach einer Leistung gesucht wird, bei der `getFach` und `istBenotet` denselben Wert wie bei der neuen Leistung liefern. Wird eine solche Leistung gefunden, soll Sie aus der Liste entfernt und mit `return` zurückgeliefert werden. Dazu müssen Sie den Rückgabetyp von `LeistungsListe` auf `Leistung` ändern.

  An der Aufrufstelle in `HtmlNotenspiegel.eingeben` soll für jede Leistung, die aus der Liste entfernt wurde, eine Warnung ausgegeben werden.