Praktikum zum Programmierkurs (WI)

Aufgabe 3 (Vererbung, Abstrakte Klassen)

In diesem Praktikum entwickeln wir unsere "Lernkarten-App" weiter.

Hinweis zu Bonuspunkten: *Zur Vergabe der Bonuspunkte werden wir in regelmäßigen Abständen den von Ihnen produzierten Quellcode begutachten. Informationen darüber, wann wir welche Praktikumsaufgaben begutachten, wie viele Bonuspunkte es gibt und wie Sie Ihren Quellcodestand einreichen, werden rechtzeitig bekanntgegeben.*

Hinweis zum entstehenden Code: *Verwenden Sie für den entstehenden Code weiterhin das in GitLab angelegte Projekt.*

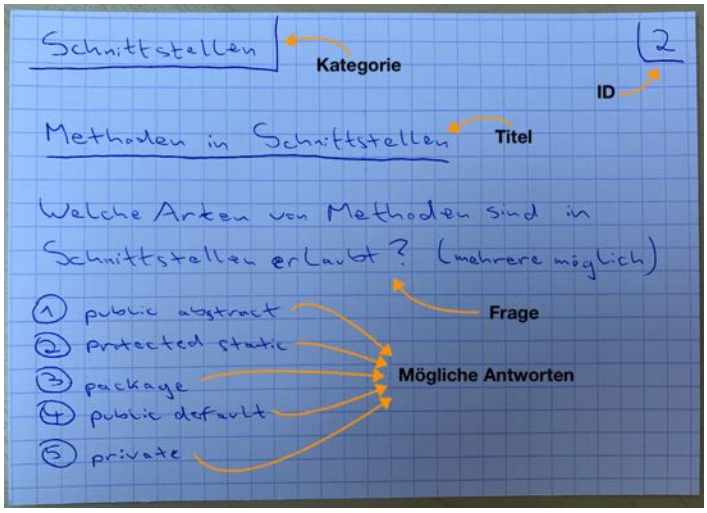
Fachliche Erweiterung: Kartentypen

Wir erweitern unsere Anwendung nun dahingehend, dass *zwei Typen von Lernkarten* unterstützt werden:

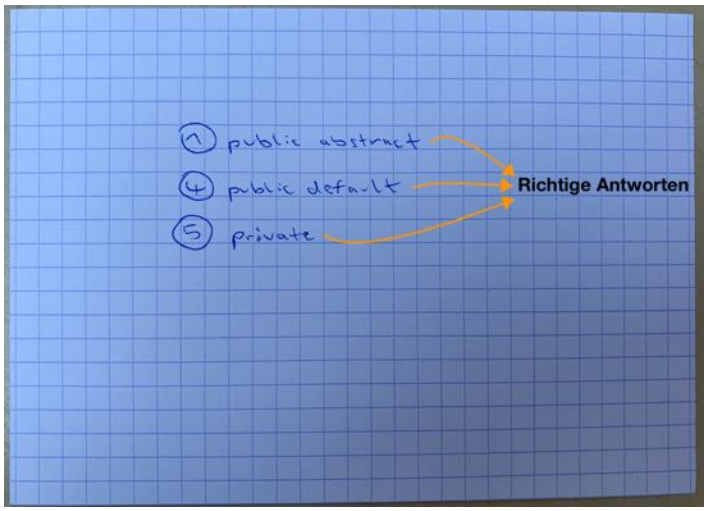
1. **Einzelantwortkarten:** Eine Einzelantwortkarte funktioniert analog zu der in Praktikum 2 vorgestellten Lernkarte: Auf der Vorderseite der Karte finden sich Kategorie, ID, Titel sowie eine Frage. Auf der Rückseite steht die Antwort auf die Frage.
2. **Mehrfachantwortkarten:** Eine Mehrfachantwortkarte enthält auf der Vorderseite ebenfalls Kategorie, ID, Titel sowie eine Frage. *Zusätzlich* stehen auf der Vorderseite mehrere (mindestens zwei) mögliche Antworten, aus denen ausgewählt werden kann. Auf der Rückseite stehen die Antworten, die korrekt sind (je nach Frage eine oder mehrere Antworten).

Hier ein Beispiel einer Mehrfachantwortkarte:

Vorderseite:

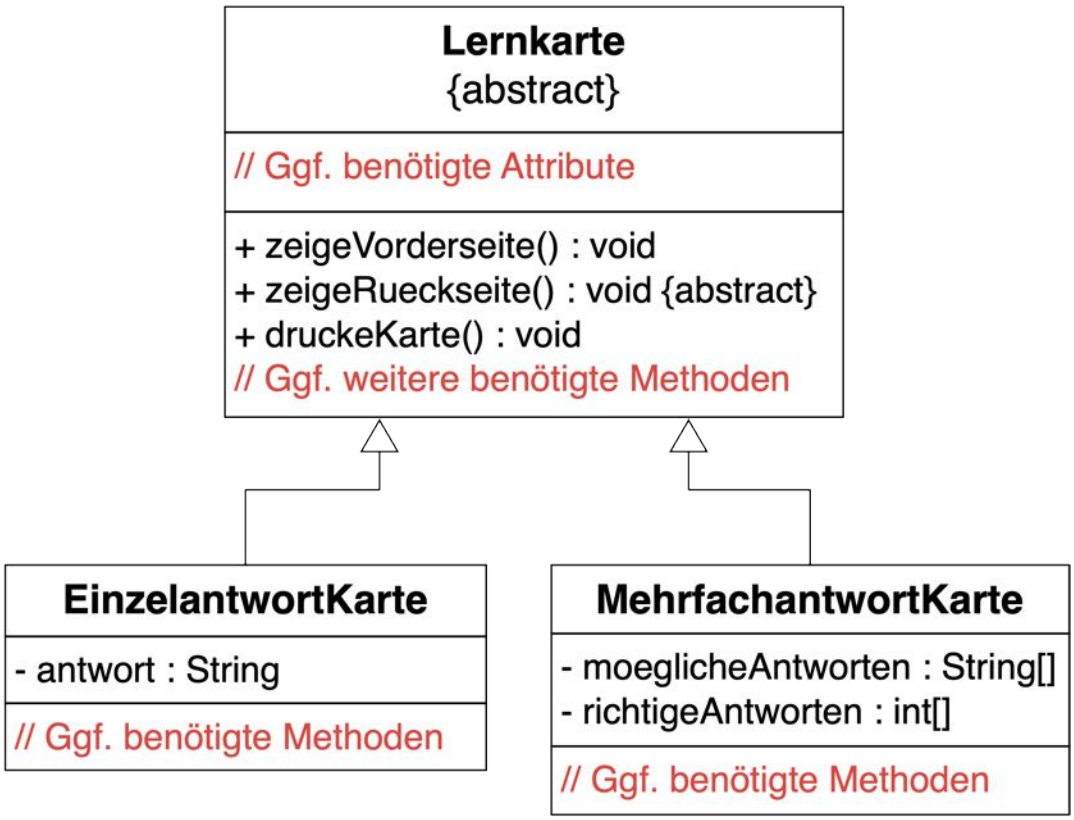


Rückseite:



1. Kartentypen implementieren

Ändern und erweitern Sie die Implementierung aus Praktikum 2 gemäß folgendem (unvollständigen!) Klassendiagramm:



Bitte berücksichtigen Sie bei der Implementierung die folgenden Punkte:

- Arbeiten Sie weiterhin im Package `pk.lkarten`.
- Die abstrakte Klasse `Lernkarte` bildet die Gemeinsamkeiten von Einzelantwortkarten und Mehrfachantwortkarten ab. *Vermeiden Sie doppelten Code!*
- Einzelantwortkarten werden durch die Klasse `EinzelantwortKarte` modelliert. Die zur Karte gehörende Antwort wird durch das Attribut `antwort` realisiert.
- Mehrfachantwortkarten werden durch die Klasse `MehrfachantwortKarte` modelliert. Diese enthält unter Anderem die Attribute:
  1. `moeglicheAntworten`: Enthält die möglichen Antworten, die auf der Vorderseite der Karte dargestellt werden.
  2. `richtigeAntworten`: Enthält die richtigen Antworten (für die Rückseite der Karte) als Zahlenwerte. Jeder Zahlenwert verweist auf einen Index im Array `moeglicheAntworten`.
- Die Methoden `zeigeVorderseite`, `zeigeRueckseite` und `druckeKarte`

sollen für Einzelantwortkarten genau so funktionieren, wie wir dies in Praktikum 2 implementiert haben.

- Für Mehrfachantwortkarten sollen die Methoden folgendermaßen funktionieren:

  1. `zeigeVorderseite` : Die Konsolenausgabe soll in folgender Form erfolgen:

     ```
     [ID, Kategorie] Titel:
     Frage
     Durchnummerierte Liste der möglichen Antworten
     Falls mehrere Antworten möglich: "(mehrere Antworten möglich)" au
     ```

     Beispiel:

     ```
     [2, Schnittstellen] Methoden in Schnittstellen:
     Welche Arten von Methoden sind in Schnittstellen erlaubt?
     1: public abstract
     2: protected static
     3: package
     4: public default
     5: private
     (mehrere Antworten möglich)
     ```

  2. `zeigeRueckseite` : Die Konsolenausgabe soll die richtigen Antworten in folgender Form darstellen:

     Beispiel:

     ```
     Die richtigen Antworten sind:
     1: public abstract
     4: public default
     5: private
     ```

  3. `druckeKarte` : Gibt die gesamte Karte (also Vorderseite und Rückseite) der Karte auf der Konsole aus - funktioniert analog zur Einzelantwortkarte.

Hinweis: Das Klassendiagramm ist unvollständig. Fügen Sie bei Bedarf weitere Attribute und Methoden hinzu.

2. Java-Anwendung

Ändern Sie Ihre ausführbare Java-Anwendung, so dass beim Ausführen der Java-Anwendung Folgendes geschieht:

1. Es sollen mindestens zwei Einzelantwortkarten und zwei Mehrfachantwortkarten

erzeugt und der Lernkartei hinzugefügt werden.

2. Rufen Sie weiterhin *alle Methoden* der Klasse `Lernkartei` auf und geben Sie das Ergebnis jeweils auf der Konsole aus.

3. Commit und Push

1. Schreiben Sie den entstandenden Code per Commit in Ihrem lokalen Repository fest. Verwenden Sie als Commit-Message "Aufgabe 3: Vererbung, Abstrakte Klassen".

2. Bringen Sie die Änderungen dann per Push auf den GitLab-Server. Kontrollieren Sie in GitLab, dass Ihre Änderungen angekommen sind.