# Programmierkurs Java

**Dr. Dietrich Boles**

## Aufgaben zu UE33-AbstrakteKlassen
**(Stand 22.02.2010)**

### Aufgabe 1:

In der Unterrichtseinheit wurde ein Java-Programm vorgestellt, mit dem eine Prüfung simuliert werden kann. An Fragetypen unterstützt dieses Programm Wahr/Falsch- und MultipleChoice-Fragen. Hierzu wurden entsprechende Klassen von einer abstrakten Klasse `Frage` abgeleitet und die daraus resultierende Polymorphie ausgenutzt.

Erweitern Sie das Programm so, dass als weiterer Fragetyp die **Mehrfachauswahl** unterstützt wird, d.h. zu einer Frage werden mehrere Antworten gegeben, von denen keine, eine oder auch mehrere korrekt sind. Leiten Sie eine entsprechende Klasse von der Klasse `Frage` ab und erweitern Sie die Methode `frageErzeugen` der Klasse `Pruefung`, so dass auch Fragen des Typs "Mehrfachauswahl" gestellt werden können.

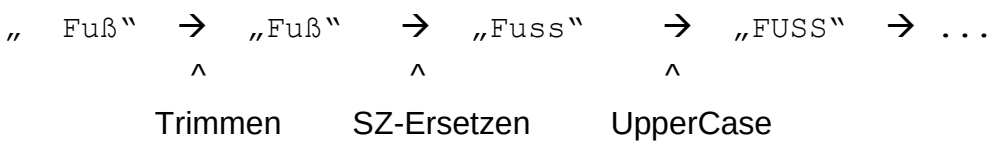
### Aufgabe 2:

In der Unterrichtseinheit wurde ein Java-Programm vorgestellt, mit dem eine Prüfung simuliert werden kann. An Fragetypen unterstützt dieses Programm Wahr/Falsch- und MultipleChoice-Fragen. Hierzu wurden entsprechende Klassen von einer abstrakten Klasse `Frage` abgeleitet und die daraus resultierende Polymorphie ausgenutzt.

Erweitern Sie das Programm so, dass als weiterer Fragetyp der **Lückenfrage** unterstützt wird, d.h. in einer Aussage fehlt ein Wort, das ein Prüfling zu ergänzen hat. Leiten Sie eine entsprechende Klasse von der Klasse `Frage` ab und erweitern Sie die Methode `frageErzeugen` der Klasse `Pruefung`, so dass auch Fragen des Typs "Lückenfrage" gestellt werden können.

### Aufgabe 3:

In dieser Aufgabe geht es um den Aufbau von Pipelines für die Ausführung von Filter-Operationen auf Strings, wie Streichen von führenden und abschließenden Leerzeichen (Trimmen), Umwandeln von Klein- in Großbuchstaben, Ersetzen von ‚ß' („sz") durch „ss", Verschlüsseln oder ähnliches; Beispiel:




Die String-Operationen der Pipeline sollen dabei beliebig erweiterbar sein, die Reihenfolge ihrer Ausführung soll beliebig angeordnet werden können und die Pipeline soll auch beliebig lang werden können.

Realisiert werden kann eine solche Pipeline dadurch, dass es für jede String-Filter-Operation eine Klasse gibt, die in einer entsprechenden Methode `verarbeiten` die Operation auf einen als Parameter übergebenen String durchführt, Beispiel:

```
class Trimmer {
  public String verarbeiten(String str) {
      // liefere getrimmten str
  }
}
```

Allerdings sind zum Aufbau einer solchen Pipeline weitere Dinge notwendig, über die Sie sich Gedanken machen sollen. Letztendlich soll es möglich sein, mit den von Ihnen zu entwickelnden Klassen Programme folgender Gestalt zu schreiben:

```
Filter pipeline = new ToUpperCaser(new SZErsetzer(new
Trimmer()));
while (true) {
     String eingabe = IO.readString("String: ");
     System.out.println(pipeline.verarbeiten(eingabe));
}
```

In diesem Beispiel wird (bei der Definition entsprechender Klassen `ToUpperCaser`, `SZErsetzer` und `Trimmer`) die obige Pipeline aufgebaut und beim Aufruf von pipeline.verarbeiten aktiviert.

Andere mögliche Pipelines wären (bei entsprechenden Klassen):

```
Filter pipeline = new Trimmer(new ToUpperCaser(new Caesar(1)));
Filter pipeline =
    new Caesar(2, new ToLowerCaser(new Trimmer(new UmlautErsetzer())));
```

Aufgabe**:**

- Überlegen Sie aufbauend auf den Konzepten der Polymorphie und des dynamischen Bindens ein Konzept zur Realisierung obiger Pipelines.
- Implementieren Sie konkret die Klassen `ToUpperCaser`, `SZErsetzer` und `Trimmer`, sodass sich das Beispielprogramm übersetzen lässt und die gewünschte Pipeline realisiert.

**Aufgabe 4:**

In der Unterrichtseinheit wurde ein Java-Programm vorgestellt, mit dem eine Prüfung simuliert werden kann. An Fragetypen unterstützt dieses Programm Wahr/Falsch- und MultipleChoice-Fragen. Hierzu wurden entsprechende Klassen von einer abstrakten Klasse `Frage` abgeleitet und die daraus resultierende Polymorphie ausgenutzt.

Erweitern Sie das Programm so, dass als weiterer Fragetyp **Zahlenfolgen-erweiterungen** unterstützt werden, d.h. es werden *n* Zahlen einer Zahlenfolge bekannt gegeben und der Prüfling muss daraus die zugrunde liegende Zahlenfolge identifizieren und die *n+1*-te Zahl der Zahlenfolge eingeben. Leiten Sie eine entsprechende Klasse von der Klasse `Frage` ab und erweitern Sie die Methode `frageErzeugen` der Klasse `Pruefung`, so dass auch Fragen des Typs " **Zahlenfolgenerweiterungen** " gestellt werden können.

Beispiel**:**

```
Frage 1 (Zahlenfolgen ergaenzen)
Anzahl an vorgegebenen Zahlen: 4
Zahl 0: 1
Zahl 1: 2
Zahl 2: 3
Zahl 3: 4
Korrekte Folgezahl: 5
Erreichbare Punkte: 10

...

Identifizieren Sie die Zahlenfolge und geben Sie die nächste Zahl der Folge
ein!
1 2 3 4 5
Richtige Antwort: 10 Punkte
```