Abstrakte Klassen und Interfaces II

Lernziele

- Sie setzen Interfaces bei der Lösung von Problemstellungen im Umfang von einigen Klassen gezielt und korrekt ein.
- Sie kennen das Einsatzgebiet des Strategy-Pattern und können dieses bei einfachen Problemstellungen gezielt einsetzten.

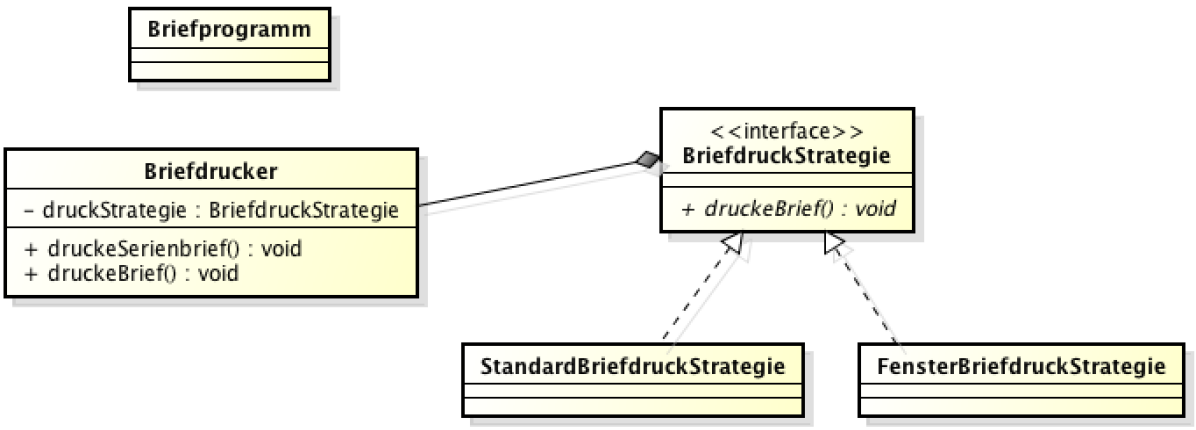
Aufgabe 1

Forken Sie für diese Aufgabe das (leere) Projekt https://github.engineering.zhaw.ch/prog1-kurs/10_Praktikum-2_Brief. Nutzen Sie Eclipse um die eigene Projektkopie auf Ihren Computer zu holen und zu bearbeiten. Gegeben sind die kompletten Klassen `Brief`, `Adresse` und `Inhalt` und Skelette für zwei weitere Klassen. Sie sollen nun eine Applikation schreiben, die folgende Funktionalität realisiert:

- Drucken von Briefen und Serienbriefen. Serienbriefe gehen an verschiedene Empfänger, wobei die Briefe bis auf den jeweiligen Empfänger alle identisch sind. Sie können die Briefe einfach auf der Konsole ausgeben, dabei soll aber sämtliche Information eines Briefs (Datum, Sender, Empfänger, Titel, Anrede, Text) ausgegeben werden.
- Die Briefe können zudem in zwei Formaten ausgegeben werden:
    - Standard: Alles linksbündig
    - Fenster: Einrücken der Empfängeradresse

Verwenden Sie in dieser Aufgabe das Strategy-Pattern. Dadurch soll es möglich sein, beliebige weitere „Druckformate“ hinzuzufügen ohne dass irgendwelche Änderungen an bestehenden Klassen vorgenommen werden müssen.

Nachfolgend ist beschrieben, wie das Programm funktionieren soll:



- `Briefprogramm` ist das Hauptprogramm, das den Inhalt resp. die Adresse der Briefe erzeugt und den `Briefdrucker` anweist, diese zu drucken. D.h. die Briefe werden durch den Briefdrucker erzeugt.

- `Briefdruckstrategie` ist ein Interface, welches die Briefdruck-Strategie vorgibt (die Methode `druckeBrief`).
- `StandardBriefdruckStrategie` und `FensterBriefdruckStrategie` implementieren dieses Interface und bieten dadurch konkrete Druckfunktionalität.
- `Briefdrucker` druckt die Briefe. Beim Erzeugen des Objekts erhält der Konstruktor eines der Objekte, die `BriefdruckStrategie` implementieren, welches dann zum Drucken verwendet wird. Der Briefdrucker bietet zudem passende Methoden, um einen einzelnen Brief oder einen Serienbrief zu drucken. Überlegen Sie sich, was die jeweiligen Methoden für Parameter haben sollten, um diese Funktionalität implementieren zu können.

Wie oben erwähnt ist es damit nun möglich, weitere Implementierungen von `BriefdruckStrategie` zu erzeugen, ohne dass `Briefdrucker` verändert werden muss.

Aufgabe 2

Eine Unschönheit des Ansatzes in Aufgabe 1 ist, dass die Strategieklassen den Brief formatieren *und* diesen drucken, obwohl dies eigentlich zwei verschiedene Aufgaben sind.

Passen Sie ihr Programm an, um diese beiden Aufgaben zu trennen. Verwenden Sie dazu zwei Strategien: Eine um einen Brief zu formatieren (dabei soll ein String zurückgegeben werden) und eine um den formatierten String zu drucken. Entsprechend wird `Briefdrucker` dann konkrete Implementierungen beider Strategien benötigen.

Bezüglich Formatierungen sollen wiederum die beiden oben definierten Formate unterstützt werden: Standard und Fenster. Verwenden Sie die Klasse `java.lang.StringBuilder` für das Zusammenstellen des formatierten Strings. Beim Drucken genügt es, wenn die Ausgabe auf der Konsole („Druck auf die Konsole“) implementiert wird. Es sind aber natürlich weitere Ausgaben denkbar: auf dem Drucker, in ein File etc.

Beachten Sie die Namensgebung. Der Interface-Name `BriefdruckStrategie` aus Aufgabe 1 macht nun z.B. keinen Sinn mehr, denn die beiden neuen Strategien sollen einerseits einen *Brief formatieren* und andererseits einen beliebigen *String drucken* – berücksichtigen Sie dies bei der Namensgebung.