Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Tel.: (0221) 8275 – 2489
Email: hans.nissen@th-koeln.de

Entwicklung komplexer Software-Systeme

Praktikumsblatt 2
Gruppe B
- Hausaufgaben -

Ziel: Anwendung Reflection, Annotationen und DI-Framework Guice

Abgabe der Lösungen: Bis zum 12.12., 08:00 Uhr morgens, im Master-Branch des Gitlab-Repositories P2EKS<IhreTeamnummer>. Abzugeben ist das vollständige IntelliJ-Projekt.
Hinweis: In Ihrem Gitlab-Repository habe ich Ihnen ein IntelliJ-Projekt zur Verfügung gestellt, welches Sie als Grundlage Ihrer Implementierung verwenden sollen – Sie sollen also den Ordner P2EKS<IhreTeamnummer> als IntelliJ-Projekt öffnen.

Aufgaben:

H2.1 Anwendung im DI-Framework Guice implementieren

Das DI-Framework Guice soll an einem (sehr) kleinen Beispiel angewendet werden: Es soll für die Artikel in unserem Online-Shop der Verkaufspreis berechnet werden. Der Verkaufspreis ergibt sich aus dem Grundpreis eines Artikels plus unserer Gewinnmarge plus der aktuellen Mehrwertsteuer. In unserer Anwendung werden der Gewinn und die Mehrwertsteuer für einen Artikel berechnet und mit bestimmten Anmerkungen auf der Console ausgegeben. Weil sich der Mehrwertsteuer-Satz derzeit oft ändert und wir auch unsere Gewinnmarge darauf anpassen müssen, setzen wir in dieser Anwendung Dependency Injection ein.
Zur Klärung wichtiger Begriffe in unserer Anwendung:

- Ein Artikel besitzt einen Grundpreis.
- Der Gewinn ist ein bestimmter Prozentsatz des Grundpreises.
- Der Nettopreis ist die Summe aus Grundpreis und Gewinn
- Die Mehrwertsteuer ist ein bestimmter Prozentsatz des Nettopreises.
- Der Bruttopreis ist die Summe aus Nettopreis und Mehrwertsteuer.
- Der Verkaufspreis ist der Bruttopreis.

.

Gegeben ist:

- Das Interface `PreisBerechner`, welches die Methoden
  - `gewinn(grundpreis : double):double` besitzt. Diese Methode liefert für den übergebenen `grundpreis` eines Artikels den gewünschten Gewinn.
  - `mwst(nettopreis : double):double` besitzt. Diese Methode liefert für den übergebenen `nettopreis` eines Artikels die zugehörige Mehrwertsteuer.
- Die Klasse `Artikel` mit den Attributen
  - `grundpreis : double`
  - `bezeichnung : String`
  - und zugehörigen Settern, Gettern und einem Konstruktor

Ihre Aufgaben:

Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Tel.: (0221) 8275 – 2489
Email: hans.nissen@th-koeln.de

- Alle diese Aufgaben sollen in dem gegebenen IntelliJ-Modul `aufgabe1` realisiert werden.
- Implementieren Sie die Klasse `PreisBerechnerNovember2022` als Implementierung des Interface `PreisBerechner`.
  - die Methode `gewinn(grundpreis)` liefert als Ergebnis genau 20% des übergebenen Grundpreises.
  - die Methode `mwst(nettopreis)` liefert als Ergebnis genau 7% des übergebenen Nettopreises.
- Implementieren Sie die Klasse `VerkaufspreisBerechner`, mit folgenden Eigenschaften:
  - Attribut vom Typ `String` für die Anmerkung bei der Ausgabe des berechneten Gewinns (dieses ist der später auf der Console auszugebene String zur Kennzeichnung des berechneten Gewinns),
  - Attribut vom Typ `String` für die Anmerkung bei der Ausgabe der berechneten Mehrwertsteuer (dieses ist der später auf der Console auszugebene String zur Kennzeichnung der berechneten Mehrwertsteuer),
  - Konstruktor, in dem die gültige Implementierung des Interface `PreisBerechner` gesetzt wird,
  - Methode `verkaufspreis(dieserArtikel : Artikel):void`, welche
    - zunächst die Bezeichnung des übergebenen Artikels ausgibt,
    - dann den Grundpreis des übergebenen Artikels mit vorangestellter Anmerkung „Grundpreis: „ auf der Console ausgibt,
    - dann den berechneten Gewinn zum Grundpreis des übergebenen Artikels mit der vorangestellten entsprechenden Anmerkung zum Gewinn auf der Console ausgibt
    - dann die berechnete Mehrwertsteuer zur Summe aus Grundpreis und Gewinn (=Nettopreis) mit der vorangestellten entsprechenden Anmerkung zur berechneten Mehrwertsteuer auf der Console ausgibt,
    - und schließlich den Verkaufspreis des Artikels (= Grundpreis + Gewinn + Mehrwertsteuer) mit vorangestellter Anmerkung „Verkaufspreis:„ auf der Console ausgibt.
  - Sie können bei Bedarf noch weiter Attribute hinzufügen.
  - Diese Klasse kennt natürlich die konkreten Anmerkungen und die konkrete Implementierung des Interface `PreisBerechner` nicht – diese werden erst durch Injektion hinzugefügt!
- Realisieren Sie alle Bindungen der Klasse `VerkaufspreisBerechner` unter Verwendung von Guice:
  - Die beiden Anmerkungen zum Gewinn und zur Mehrwertsteuer werden mittels Attribut-Injektion übergeben
  - Die konkrete Implementierung des Interface `PreisBerechner` wird mittels Konstruktor-Injektion übergeben
  - Verwenden Sie als Anmerkung für den Gewinn den String „Gewinn beträgt 20%“,
  - Verwenden Sie als Anmerkung für die Mehrwertsteuer den String „"MWSt beträgt 7%"“,
  - Verwenden Sie als konkrete Implementierung des Interface `PreisBerechner` die Klasse `PreisBerechnerNovember2022`
- Implementieren Sie eine Klasse `MainClass`, in der in einer `main`-Funktion die Methode `verkaufspreis()` an einer Instanz der Klasse `VerkaufspreisBerechner` aufgerufen wird für die beiden folgenden Artikel:

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Tel.: (0221) 8275 – 2489
Email: hans.nissen@th-koeln.de

- Artikel1 mit Grundpreis = 210.78 und Bezeichnung = „Artikel1“
- Artikel2 mit Grundpreis = 100.00 und Bezeichnung = „Artikel2“

– Man kann diese Funktionalität natürlich auch ohne Guice und DI realisieren. In dieser Aufgabe sollen Sie aber DI und Guice verwenden!

H2.2 Anwendung mit Reflection und Annotationen implementieren

In dieser Aufgabe soll das Entwurfsmuster Schablonenmethode unter Verwendung von Annotationen und Reflection umgesetzt werden.
Im Entwurfsmuster Schablonenmethode existiert ja eine Methode, die einige lokal definierte Methoden aufruft und zusätzlich abstrakte, noch nicht definierte Methoden aufruft. Typischerweise werden diese abstrakten Methoden in einer Unterklasse definiert. Eine Umsetzung dieses Musters unter Verwendung der Vererbung ist in dieser Aufgabe nicht erwünscht.
Zur Motivation: Der Vorteil dieser Lösungsvariante des Musters Schablonenmethode ist, dass die variablen Methoden auf mehrere Klassen aufgeteilt werden können. Dieses ist in der Umsetzung des Musters mit Unterklassen nicht möglich.

Gegeben sind folgende Klassen:

- `SchablonenKlasse`
  - mit einem vordefinierten Konstruktor,
  - mit Schablonen-Methode `schablonenMethode()`:
    - diese ruft die lokal definierten Methoden `findePrimzahl()`, `ausgabe()` und `restMod5()` auf - diese Methoden sind also bekannt und fertig implementiert und müssen nicht erweitert oder verändert werden.
    - diese ruft die nicht vollständig implementierten Methoden `bearbeite1()`, `bearbeite2()` und `bearbeite3()` auf - die konkrete Implementierung dieser Methoden soll flexibel und variabel gehalten werden. Deshalb sollen für diese Methoden implementierte Methoden aus anderen Klassen aufgerufen werden.
- `KlasseMitMethoden1` mit mehreren vollständig implementierten Methoden.
- `KlasseMitMethoden2` mit mehreren vollständig implementierten Methoden.
- `KlasseMitMethoden3` mit mehreren vollständig implementierten Methoden.
- `Starter` mit der die Beispielanwendung gestartet wird.

Idee:
Die Idee zur Ermittlung der konkreten Methoden für die drei nicht vollständig implementierten Methoden `bearbeite1()`, `bearbeite2()` und `bearbeite3()` ist die Folgende:

- Für jede der drei nicht vollständig implementierten Methoden `bearbeite1()`, `bearbeite2()` und `bearbeite3`() existiert eine eindeutige Annotation. Diese Annotationen sollen Sie selbst festlegen und erstellen.
- Im Konstruktor der Klasse `SchablonenKlasse` werden mehrere Klassen (vom Typ Class<?>) in einem Array übergeben. Im Beispiel unten sind es drei Klassen. Es könnten aber auch zwei oder mehr als drei Klassen sein. Über diese Klassen wissen wir:
  - An jeder Klasse ist genau ein Konstruktor deklariert und dieser Konstruktor ist parameterlos.
  - Genau eine Methode der Methoden aller Klassen besitzt die eindeutige Annotation für die Methode `bearbeite1()`. Diese Methode ist die konkrete Methode für die Methode `bearbeite1()`.

Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Tel.: (0221) 8275 – 2489
Email: hans.nissen@th-koeln.de

  - Genau eine Methode der Methoden aller Klassen besitzt die eindeutige Annotation für die Methode `bearbeite2()`. Diese Methode ist die konkrete Methode für die Methode `bearbeite2()`.
  - Genau eine Methode der Methoden aller Klassen besitzt die eindeutige Annotation für die Methode `bearbeite3()`. Diese Methode ist die konkrete Methode für die Methode `bearbeite3()`.
- In der Methode `bearbeite1(int zahl1)` soll die Methode mit der zugehörigen Annotation gesucht, dann mit dem korrekten Parameter aufgerufen und das Ergebnis dieses Methodenaufrufes als Ergebnis von `bearbeite1()` zurückgegeben werden.
- Identisch ist für `bearbeite2(int zahl2)` zu verfahren, dann jedoch mit der zu dieser Methode gehörigen (selbst definierten) Annotation.
- Identisch ist für `bearbeite3(int zahl3)` zu verfahren, dann jedoch mit der zu dieser Methode gehörigen (selbst definierten) Annotation.
- Die `SchablonenKlasse` weiß nicht, um welche konkreten Klassen es sich handelt.
- Die `SchablonenKlasse` weiß nicht, welche Methode welcher Klasse aufgerufen werden muss.
- Die `SchablonenKlasse` weiss nicht, ob die Methoden `private` oder `public` definiert sind. Sichtbarkeiten der Methoden dürfen in der Klasse nicht verändert werden - nur über Reflection.

Ihre Aufgaben:

- Alle diese Aufgaben sollen in dem gegebenen IntelliJ-Modul `aufgabe2` realisiert werden.
- Sie müssen die drei Annotationen erstellen und im Beispiel anwenden. Erstellen Sie diese Annotationen im Paket `annotationen`.
- Sie sollen die drei bisher nicht vollständig implementierten Methoden `bearbeite1()`, `bearbeite2()` und `bearbeite3()` wie oben beschrieben vervollständigen.
  .
- Für unser konkretes Beispiel soll:
  - Methode `rechne()` an Klasse `KlasseMitMethoden1` für `bearbeite1()` verwendet werden.
  - Methode `macheWasMitZahl()` an Klasse `KlasseMitMethoden2` für `bearbeite2()` verwendet werden.
  - Methode `rechne()` an Klasse `KlasseMitMethoden3` für `bearbeite3()` verwendet werden.
  - Dieses ist aber nur ein Beispiel und darf natürlich nicht in der Implementierung der `SchablonenKlasse` verwendet werden.
  - Die `SchablonenKlasse` muss für beliebige Klassen verwendbar sein.
  - Es kann auch sein, dass eine Klasse mehrere benötigte Methoden besitzt (also bspw. zur Ausführung von `bearbeite2()` und `bearbeite3()`).

- Nun können Sie die main-Methode in der Klasse Starter aufrufen.