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 C
- Hausaufgaben -

Ziel: Anwendung Reflection, Annotationen und DI-Framework Guice

Abgabe der Lösungen: Bis zum 20.12., 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 einen Sensor ein Dashboard erstellt werden. Dieses Dashboard gibt 10 Werte des Sensors aus und markiert Sensorwerte, die über einer gegebenen oberen Alarmgrenze liegen bzw. unter einer gegebenen unteren Alarmgrenze liegen.
Damit wir dieses Dashboard in einer sicheren Umgebung testen können, wollen wir das Dashboard zunächst nur auf einen Dummy-Sensor anwenden.

Gegeben ist:
- Das Interface `ISensor`, welches die Methode `gibWert():Integer` besitzt. Diese Methode liefert immer den aktuellen Wert eines Sensors.

Ihre Aufgaben:

- Alle diese Aufgaben sollen in dem gegebenen IntelliJ-Modul `aufgabe1` realisiert werden.
- Implementieren Sie die Klasse `DummySensor` als Implementierung des Interface `ISensor`. In der Methode `gibWert()` liefert dieser immer eine Zufallszahl zwischen 1 und 10 (siehe Java-Klasse Random).
- Implementieren Sie die Klasse `SensorDashboard`, mit folgenden Eigenschaften:
  - Attribut vom Typ `Integer` für die untere Alarmgrenze
  - Attribut vom Typ `Integer` für die obere Alarmgrenze
  - Konstruktor, in dem der betrachtete Sensor gesetzt wird (aber nicht die Alarmgrenzen!)
  - Methode `ausgabeSensorWerte():void`, welche in einer Schleife 10-Mal die Methode `gibWert()` des betrachteten Sensors aufruft, diesen aktuellen Wert auf der Console ausgibt und
    - falls dieser aktuelle Wert gleich oder niedriger als die untere Alarmgrenze ist, die Meldung „Achtung: aktueller Wert niedriger als untere Alarmgrenze!“ direkt hinter dem aktuellen Wert ausgibt,

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

        - falls dieser aktuelle Wert gleich oder größer als die obere Alarmgrenze ist, die Meldung „Achtung: aktueller Wert höher als obere Alarmgrenze!“ direkt hinter dem aktuellen Wert ausgibt.
    - Sie können bei Bedarf noch weiter Attribute hinzufügen.
    - Diese Klasse kennt natürlich die konkreten Alarmgrenzen und die konkrete Implementierung des Interface `ISensor` nicht – diese werden erst durch Injektion hinzugefügt!
- Realisieren Sie alle Bindungen der Klasse `SensorDashboard` unter Verwendung von Guice:
    - Die obere und untere Alarmgrenze werden mittels Attribut-Injektion übergeben
    - Der konkrete Sensor (in unserem Fall der Dummy-Sensor) wird mittels Konstruktor-Injektion übergeben
    - Verwenden Sie als untere Alarmgrenze den Wert 3 und als obere Alarmgrenze den Wert 8.
    - Verwenden Sie als konkrete Implementierung des Interface `ISensor` die Klasse `DummySensor`
- Implementieren Sie eine Klasse `MainClass`, in der in einer `main`-Funktion die Methode `ausgabeSensorWerte()` an einer Instanz der Klasse `SensorDashboard` aufgerufen wird.
- 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

Geschichte zur Aufgabe:
Unsere Firma expandiert derzeit erheblich. Wir haben gerade Firmen in Spanien und England übernommen. Das Problem hierbei ist die IT-Landschaft. Die Mitarbeiter-Klassen aus Spanien und England sind in der jeweiligen Landessprache erstellt und passen deshalb nicht mit unseren deutschen Mitarbeiter-Klassen zusammen. Trotzdem muss das Gehalt aller Mitarbeiter einheitlich berechnet werden. Weil wir demnächst auch noch Firmen in anderen Ländern übernehmen wollen, benötigen wir ein allgemeines Konzept zur Berechnung der Gehälter. Wir haben 2 Typen von Mitarbeitern: normale Angestellte und Abteilungsleiter.
Es existiert bereits zur Berechnung der Gehälter:

- Die Klasse `GehaltAbteilungsleiter` mit der Methode `berechneGehalt(grundgehalt, anzahlMitarbeiter, bonus)`. Diese berechnet das Monatsgehalt eines Abteilungsleiters auf Basis seines Grundgehalts, der Anzahl der Mitarbeiter in seiner Abteilung und seinem Bonus.
- Die Klasse `GehaltAngestellter` mit der Methode `berechneGehalt(grundgehalt, überstunden, einstellungsjahr)`. Diese berechnet das Monatsgehalt eines Angestellten auf Basis seines Grundgehalts, der Anzahl seiner Überstunden und seinem Einstellungsjahr.
- Diese Klassen und Methoden sind fertig programmiert und sollen nicht mehr verändert werden.

Wir haben insgesamt 6 Mitarbeiter-Klassen:

- `Angestellter` – für einen Angestellten in Deutschland mit Attributen
    - `angName`: der Name des Angestellten
    - `grundgehalt`: das Grundgehalt des Angestellten
    - `anzahlUeberstunden`: die Anzahl an Überstunden in einem Monat
    - `einstellungsJahr`: das Jahr seiner Einstellung
- `Abteilungsleiter` – für einen Abteilungsleiter in Deutschland mit Attributen:

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

  - `seinName`: der Name des Abteilungsleiters
  - `grundgehalt`: das Grundgehalt des Abteilungsleiters
  - `anzahlMitarbeiter`: Anzahl der Mitarbeiter in der Abteilung des Abteilungsleiters
  - `bonus`: der vereinbarte Bonus des Abteilungsleiters
- `Gerente` – für einen Abteilungsleiter in Spanien mit Attributen
  - `numeroEmpleados`: entspricht `anzahlMitarbeiter`
  - `salarioBase`: entspricht `grundgehalt`
  - `prima`: entspricht `bonus`
  - `gerenteNombre`: entspricht `seinName`
- `Empleado` – für einen Angestellten in Spanien mit Attributen
  - `horasExtraordinarias`: entspricht `anzahlUeberstunden`
  - `salarioBase`: entspricht `grundgehalt`
  - `anoReclutamiento`: entspricht `einstellungsJahr`
  - `emplNombre`: entspricht `angName`
- `Manager` – für einen Abteilungsleiter in England mit Attributen
  - `addition`: entspricht `bonus`
  - `numberEmployees`: entspricht `anzahlMitarbeiter`
  - `baseSalary`: entspricht `grundgehalt`
  - `managerName`: entspricht `seinName`
- `Employee` – für einen Angestellten in England mit Attributen
  - `recruitmentYear`: entspricht `einstellungsJahr`
  - `numberOvertimeHours`: entspricht `anzahlUeberstunden`
  - `baseSalary`: entspricht `grundgehalt`
  - `emplName`: entspricht `angName`
- Alle diese Klassen sind auch schon vollständig realisiert. Aber alle Klassen haben unterschiedliche Attributnamen für eigentlich die gleichen Informationen.

Es wurde auch schon die Klasse `MainClass` erstellt, in der alle derzeit vorhandenen Mitarbeiter unserer Firma definiert sind. Dort wird auch schon die von Ihnen zu erstellende Methode `berechneGehaelter()` aufgerufen.

Ihre Aufgaben:

- Alle diese Aufgaben sollen in dem gegebenen IntelliJ-Modul `aufgabe2` realisiert werden.
- Die Gehaltsberechnung für alle Mitarbeiter soll ohne Kenntnis der Mitarbeiter-Klassen funktionieren – es können noch weitere Mitarbeiter-Klassen mit einer anderen Struktur hinzukommen und die Gehaltsberechnung muss dann auch mit diesen funktionieren.
- Erstellen Sie geeignete Annotationen, um an den Mitarbeiter-Klassen zu kennzeichnen,
  - ob es sich um einen Angestellten oder einen Abteilungsleiter handelt, und
  - um welche Art von Information es sich bei einer Getter-Methode handelt.
- Diese Annotation sind im Paket `annotationen` zu erstellen und dann an den Mitarbeiter-Klassen anzuwenden.
- Vervollständigen Sie an der Klasse `GehaltFirma` die Methode `berechneGehaelter()`. Diese nimmt ein Array von Objekten (= Objekte zu unseren Mitarbeiter-Klassen) als Eingabe und soll für jedes Objekt dann in Abhängigkeit vom Typ des Mitarbeiters (Angestellter oder Abteilungsleiter) die Methode

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

`berechneGehalt()` an der Klasse `GehaltAbteilungsleiter` für einen Abteilungsleiter bzw. die Methode `berechneGehalt()` an der Klasse `GehaltAngestellter` für einen Angestellten mit den korrekten Parameterbelegungen aufrufen.

- Für einen Abteilungsleiter soll diese Methode auf der Console ausgeben: „Gehalt vom Abteilungsleiter <Name des Abteilungsleiters> : <berechnetes Gehalt des Abteilungsleiters>“
- Für einen Angestellten soll diese Methode auf der Console ausgeben: „Gehalt vom Angestellten <Name des Angestellten> : <berechnetes Gehalt des Angestellten>“
- Die Methode `berechneGehaelter()` muss also auf Basis von Annotationen die richtigen Getter aufrufen, um an die benötigten Informationen zu gelangen.