PLF in Programmieren und Software Engineering

> Klasse: 5AAIF
> Datum: DI, 16. Dezember 2025
> Arbeitszeit: 3 UE

Wichtiger Hinweis vor Arbeitsbeginn

Auf dem Laufwerk *Z* finden Sie die Datei *Plf5aaif_20251216.7z*. Klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie *Weitere Optionen - 7-Zip* und *Extract here*. Gehen Sie dann in den Ordner *Plf5aaif* und starten Sie die Datei *start_solution.cmd*. Diese Datei lädt zuerst alle Dependencies aus dem Internet und startet dann die sln Datei in diesem Ordner. Sie müssen das Programm nicht abgeben, denn Sie arbeiten direkt am Netzlaufwerk.

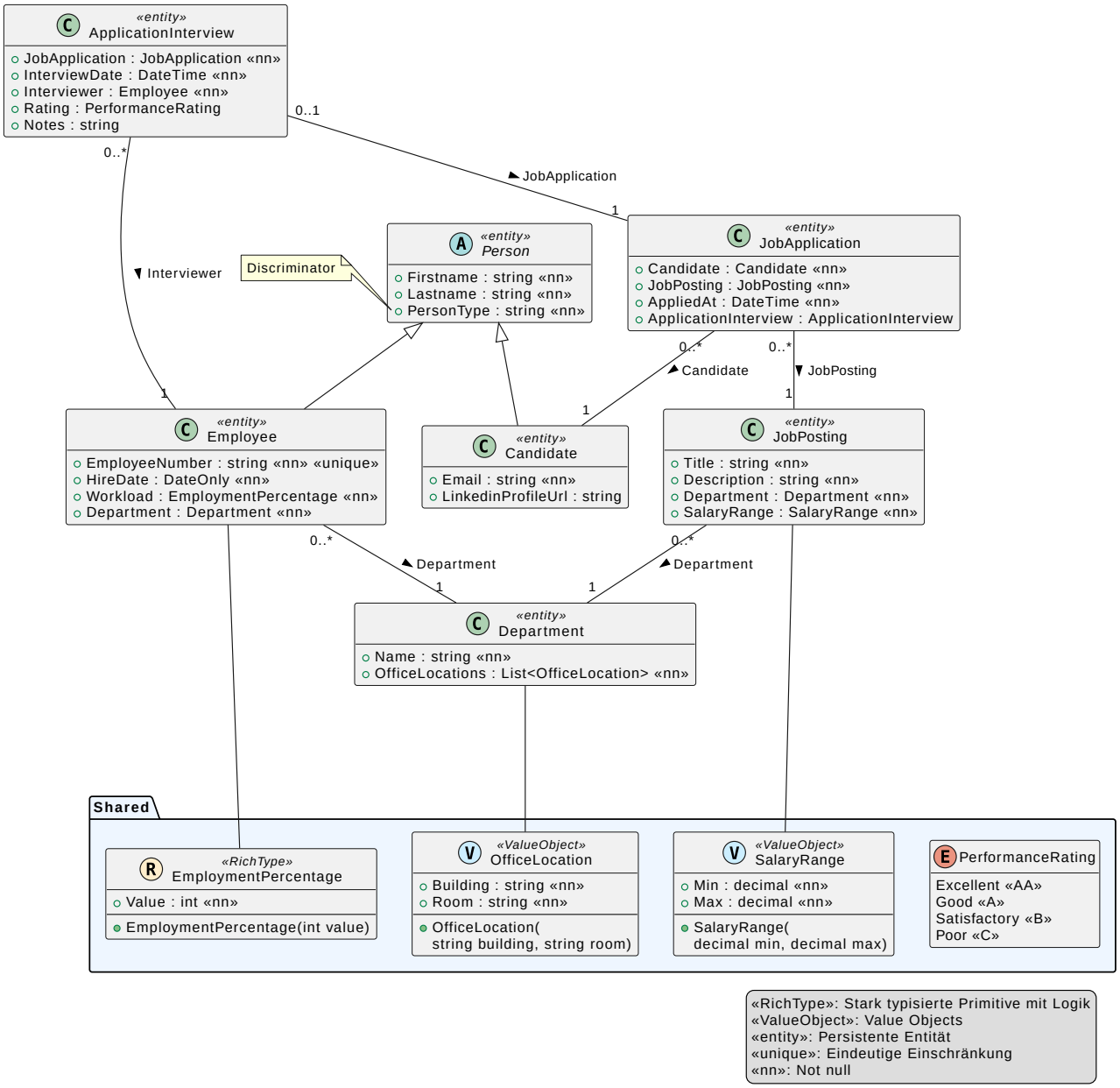
Füllen Sie die Datei *README.md* in *Plf5aaif/README.md* mit Ihren Daten (Klasse, Name und Accountname) aus. Sie sehen die Datei in Visual Studio unter *Solution Items* nach dem Öffnen der Solution. **Falls Sie dies nicht machen, kann Ihre Arbeit nicht zugeordnet und daher nicht bewertet werden!**

Während der Prüfung ist der Internetzugriff gesperrt. Führen Sie daher niemals *Build - Rebuild Solution* aus, denn dadurch werden die lokalen Pakete gelöscht. Arbeiten Sie immer mit *Build - Build Solution* (F6).

Aufgabe: Object Relation Mapping

Ein Unternehmen möchte eine Plattform für Bewerberinnen und Bewerber entwickeln. Das System verwaltet die Daten der Mitarbeiter (*Employee*), die Job Interviews durchführen. Die Mitarbeiter können für die jeweilige Abteilung (*Department*) Jobangebote (*JobPosting*) erstellen. Ein Bewerber (*Candidate*) kann sich für dieses Jobangebot bewerben. Dadurch entsteht eine *JobApplication* (Bewerbung). Der Mitarbeiter kann nun für die Bewerbung ein *ApplicationInterview* durchführen und bewerten.

Das Domain Model sieht so aus:



Arbeitsauftrag

Erstellung der Modelklassen

Implementieren Sie das dargestellte Diagramm Modelklassen für den OR Mapper.

Im Projekt *Plf5aaif.Application* befinden sich leere Klassen sowie die Klasse *JobApplicationContext*, die Sie nutzen sollen.

Beachten Sie bei der Umsetzung folgende Punkte:

Allgemeine Anforderungen

- Legen Sie nötige Konstruktoren an. Ein *public* Konstruktor soll alle im Modell enthaltenen Properties initialisieren. Ergänzen Sie bei Bedarf die für den OR Mapper nötigen Konstrukto-

ren.

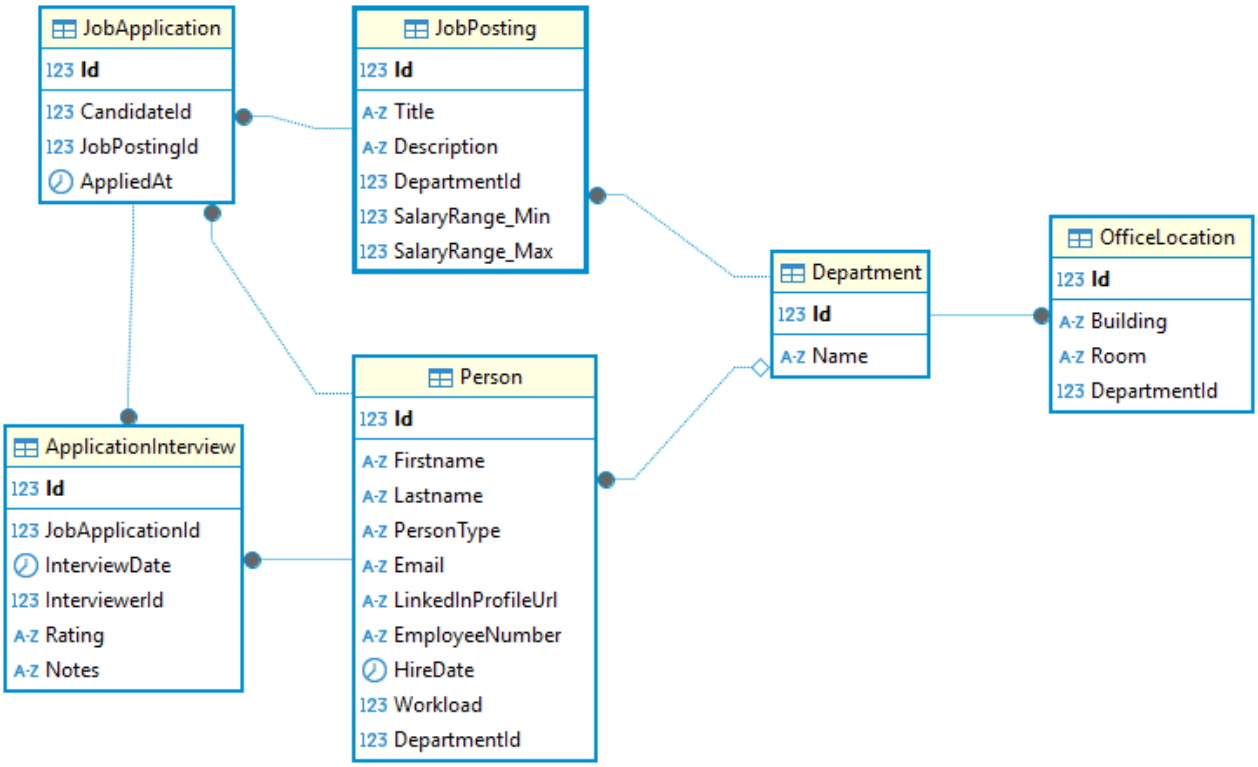
- Beachten Sie Attribute Constraints wie *not null* (`<<nn>>`).
- Verwenden sie eigene primary keys mit dem Namen *Id* (autoincrement), außer im Modell ist mit *PK* explizit ein Schlüssel mit `<<PK>>` angegeben.
- Die Foreign Keys werden nach der Convention Propertyname + Name des PK generiert. Dies ist z. B. bei der Zuweisung des FKs der dependent Entities wichtig.

Fachliche Anforderungen

- Die generierten Tabellennamen sollen in Einzahl erzeugt werden (*Person*, *Department*, ...).
- Für eine *JobApplication* kann es maximal ein *ApplicationInterview* geben. Implementieren Sie die *1:0..1* Beziehung entsprechend.
- Das Attribut *Employee.EmployeeNumber* soll ein *unique constraint* besitzen.
- Es dürfen nicht 2 JobApplications mit den gleichen Werten in *Candidate* und *JobPosting* angelegt werden können. Stellen Sie dies durch ein unique constraint über diese 2 Spalten (bzw. die darunterliegenden Fremdschlüssel) sicher.
- Das Attribut *JobPosting.SalaryRange* soll als *value object* definiert werden. Die erzeugten Spalten sollen *SalaryRange_Max* und *SalaryRange_Min* in der Datenbank heißen (Standardverhalten in EF Core).
- Das Attribut *Department.OfficeLocations* soll als Liste von *value objects* definiert werden. Stellen Sie durch Konfiguration sicher, dass der primary key der erzeugten Tabelle *Id* heißt.
- Das Attribut *Employee.Workload* soll als *rich type* definiert und als *int* gespeichert werden.
- Das Attribut *Person.PersonType* soll als discriminator definiert werden.
- Die Enum in *ApplicationInterview.Rating* soll als String mit den folgenden Werten in der Datenbank gespeichert werden: *AA* für Excellent, *A* für Good, *B* für Satisfactory, *C* für Poor. Beachten Sie, dass der Wert auch *null* sein kann.
- Implementieren Sie die Klasse *EmploymentPercentage* so, dass im Konstruktor der Wert geprüft wird. Ist der Wert nicht im Bereich von 0 - 100 (inklusive), so ist eine *ArgumentException* zu werfen.
- Implementieren Sie die Klasse *SalaryRange* so, dass im Konstruktor der Wert geprüft wird. Ist der Wert von *min* größer als der Wert von *max*, so ist eine *ArgumentException* zu werfen.

Falls Sie Probleme mit rich types oder enums haben, können Sie diese auch als einfaches Stringfeld implementieren.

Das durch den OR Mapper erzeugte Datenbankschema soll so aussehen:



Der vorgegebene Test *T00_CanCreateDatabaseTest* in *GradingTests* prüft, ob mit Ihrer Implementierung überhaupt eine Datenbank erzeugt werden kann. **Läuft dieser Test nicht durch, können Sie keine positive Note erhalten.**

Der vorgegebene Test *T00_SchemaTest* in *GradingTests* verwenden, um das Gesamtmodell zu prüfen. Läuft dieser Test erfolgreich durch, sind Sie positiv.

Verfassen von Tests

In der Klasse *JobApplicationContextTests* im Projekt *Plf5aaif.Test* sollen Testmethoden verfasst werden, die die Richtigkeit der Konfiguration des OR Mappers beweisen sollen.

- *T01_InsertJobPostingWithApplicationInterviewTest* zeigt, dass Sie eine *JobApplication* für ein *JobPosting* samt *ApplicationInterview* speichern können.
- *T02_JobApplicationCandidateAndJobPostintIsUniqueTest* zeigt, dass beim Einfügen einer zweiten *JobApplication* für den selben Kandidaten und das selbe Job Posting eine *DbUpdateException* geworfen wird.
- *T03_AddDepartmentWithOfficeLocations* zeigt, dass Sie eine Abteilung (*Department*) mit mehreren *OfficeLocations* speichern klnnen. Achten Sie im Assert darauf, dass Sie ohne *Include* beim Laden des Departments auch auf die Liste der *OfficeLocations* zugreifen können (Anzahl prüfen).
- *T04_SalaryRangeThrowsExceptionWhenMinMaxIsInvalidTest* zeigt, dass der Konstruktor der Klasse *SalaryRange* bei ungültigen Argumenten eine *ArgumentException* wirft.

Bewertung

Um eine positive Beurteilung erreichen zu können, muss das Programm kompilieren und die implementierten Modelklassen müssen ein Erzeugen der Datenbank über den OR Mapper ermöglichen.

Aufgabe (36 Punkte in Summe)	**Ges**
Das Entity Department wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity Person wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity Employee wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity Candidate wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity UserJobPosting wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity JobApplication wird mit den definierten Feldern Datenbank abgebildet.	1
Das Entity ApplicationInterview wird mit den definierten Feldern Datenbank abgebildet.	1
Die 1:1 Beziehung zwischen JobApplication und ApplicationInterview ist korrekt konfiguriert.	2
Employee.EmployeeNumber ist unique	1
Das Tupel JobApplication.Candidate und JobPosting ist unique.	2
JobPosting.SalaryRange ist als value object definiert.	2
Depertment.OfficeLocations ist als Liste von value objects definiert.	2
Employee.Workload ist als rich type mit converter definiert.	2
Person.Type ist als Discriminator definiert.	1
ApplicationInterview.Rating wird mit den korrekten Werten (AA, A, B und C) gespeichert.	3
Die Klasse SalaryRange wirft eine ArgumentException bei ungültigen Argumenten.	1
Die Klasse EmploymentPercentage wirft eine ArgumentException bei ungültigen Argumenten.	1
Der Test T01_InsertJobPostingWithApplicationInterviewTest hat den richtigen Aufbau.	2
Der Test T01_InsertJobPostingWithApplicationInterviewTest läuft durch.	1
Der Test T02_JobApplicationCandidateAndJobPostintIsUniqueTest hat den richtigen Aufbau.	2
Der Test T02_JobApplicationCandidateAndJobPostintIsUniqueTest läuft durch.	1
Der Test T03_AddDepartmentWithOfficeLocations hat den richtigen Aufbau.	2
Der Test T03_AddDepartmentWithOfficeLocations läuft durch.	1
Der Test T04_SalaryRangeThrowsExceptionWhenMinMaxIsInvalidTest hat den richtigen Aufbau.	2
Der Test T04_SalaryRangeThrowsExceptionWhenMinMaxIsInvalidTest läuft durch.	1

36 - 32 Punkte: Sehr gut, 31 - 28 Punkte: Gut, 27 - 23 Punkte: Befriedigend, 22 - 19 Punkte: Genügend, 18 - 0 Punkte: Nicht genügend.