Java Persistence API & Related Patterns

Domain Logic, Data Source, Object Relational

www.cs.uoi.gr/~zarras/http://www.cs.uoi.gr/~zarras/se.htm

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

https://www.vogella.com/tutorials/JavaPersistenceAPI/article.html/

https://www.baeldung.com/spring-data-derived-queries

1

What is JPA?

Mapping objects to **database table rows** and vice versa is called **Object-Relational Mapping (ORM)**.

The **Java Persistence API (JPA)** is an **approach to ORM** for Java objects.

Via JPA the developer can map, store, update and retrieve data from relational databases to Java objects and vice versa.

JPA is a **specification** from Oracle
https://download.oracle.com/otn-pub/jcp/persistence-2_1-fr-eval-spec/JavaPersistence.pdf?AuthParam=1647415880_273dae2fb01296223 1cd5758a30e83d3

It can have multiple compliant implementations Hibernate, EclipseLink and Apache OpenJPA.

2

How do we organize Domain layer logic?

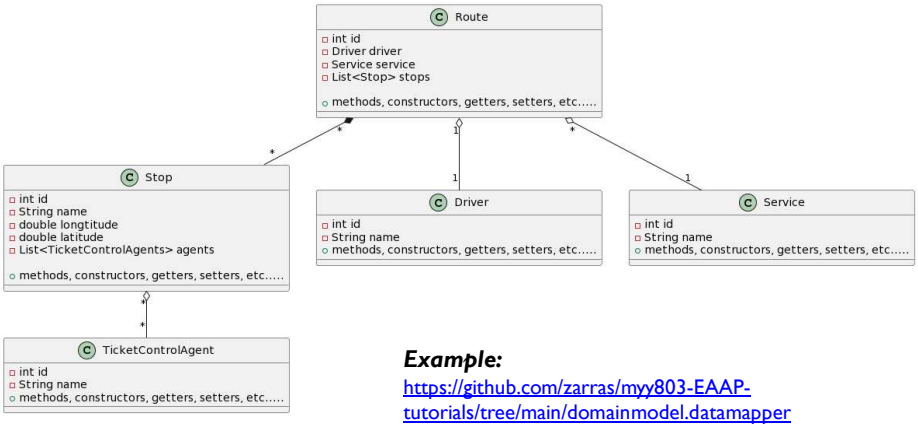
3

How do we organize Domain layer logic?

4

Domain Model

An object model of the domain that incorporates both behavior and data. A different object per table tuple/row …..



Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/domainmodel.datamapper

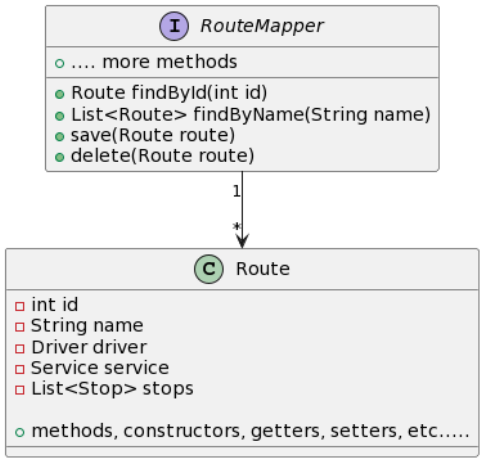
5

How do we transfer data from/to the Domain layer to/from the Data Layer ?

6

Domain Model and Data Mapper

A layer of Mappers moves data between objects and a database while keeping them independent of each other and the mapper itself.



Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/domainmodel.datamapper

https://github.com/zarras/myy803-EAAP-tutorials/tree/main/domainmodel.datamapper.springboot

7

Data Mapper

A layer of Mappers moves data between objects and a database while keeping them independent of each other and the mapper itself.

The Data Mapper is a layer of software that separates the in-memory objects from the database. Its responsibility is to transfer data between the two and also to isolate them from each other.

With Data Mapper the in-memory objects needn't know even that there's a database present.

Fits well with Domain Model.

8

How to map objects to tuples (table rows) ?

9

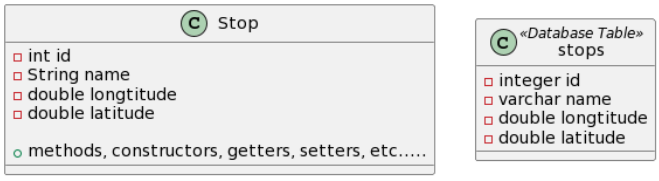
Identity Field

Saves a database ID field in an object to maintain identity between an in-memory object and a table row.

Intent

Reading data from a database is all very well, but in order to **write data back you need to tie the database to the in-memory object system**.

In essence, ***Identity Field*** *is mind-numbingly simple. All you do is store the* primary key of the relational database table in the object's fields.



Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/foreignkey.mapping

10

Foreign Key Mapping

Maps an association between objects to a foreign key reference between tables.

Objects can refer to each other directly by object references. To **save** these **objects** to a **database**, it's vital to **save these references**.

A ***Foreign Key Mapping maps*** *an* ***object reference*** *to a* ***foreign key*** *in the* database.

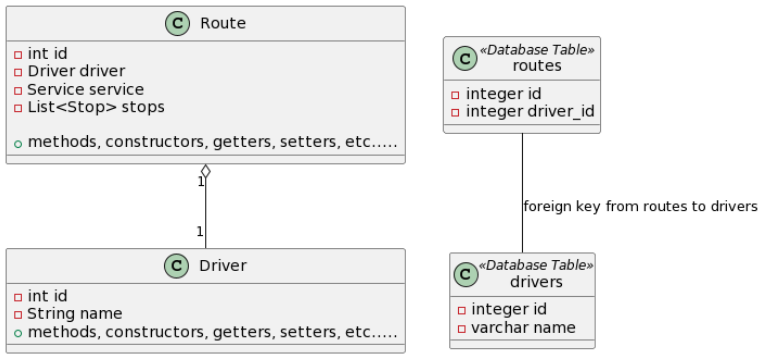
Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/foreignkey.mapping

11

Foreign Key Mapping

Maps an association between objects to a foreign key reference between tables.

One-to-One relation – Route to Driver

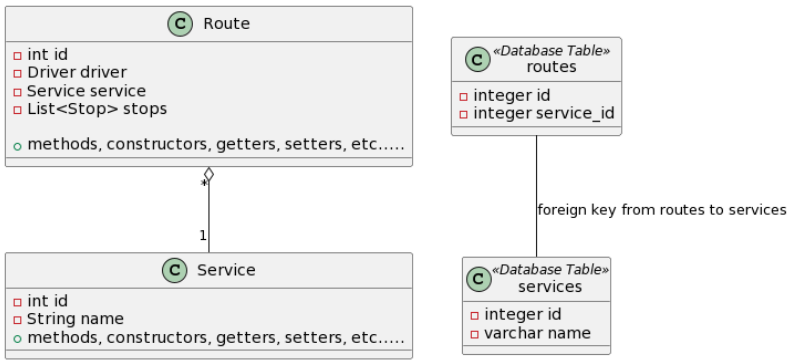


12

Foreign Key Mapping

***Maps** an **association** between **objects** to a **foreign key** reference **between tables**.*

Many-to-One relation – Route to Service

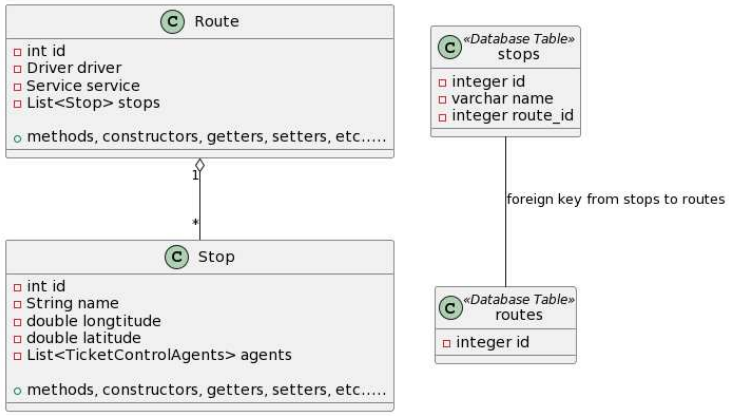


13

Foreign Key Mapping

***Maps** an **association** between **objects** to a **foreign key** reference **between tables**.*

One-to-Many relation – Route to Stop

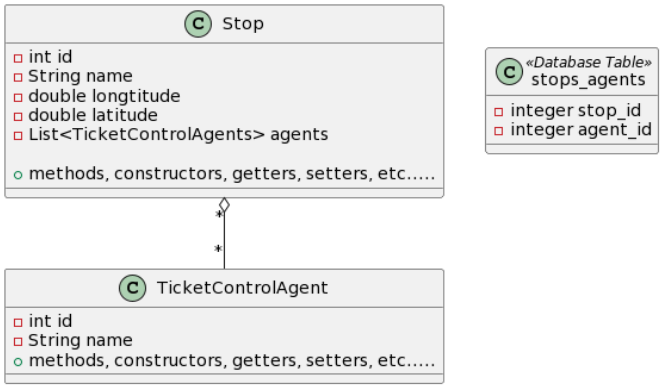


Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/foreignkey.mapping

14

Association Table Mapping

Saves a Many-to-Many association as a table with foreign keys to the tables that are linked by the association.

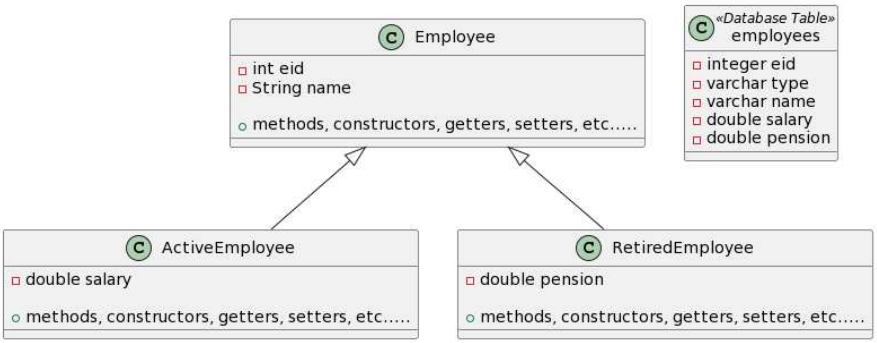


Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/foreignkey.mapping

15

Single Table Inheritance

Represents an inheritance hierarchy of classes as a single table that has columns for all the fields of the various classes.



Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/singletable.inheritence

16

Single Table Inheritance

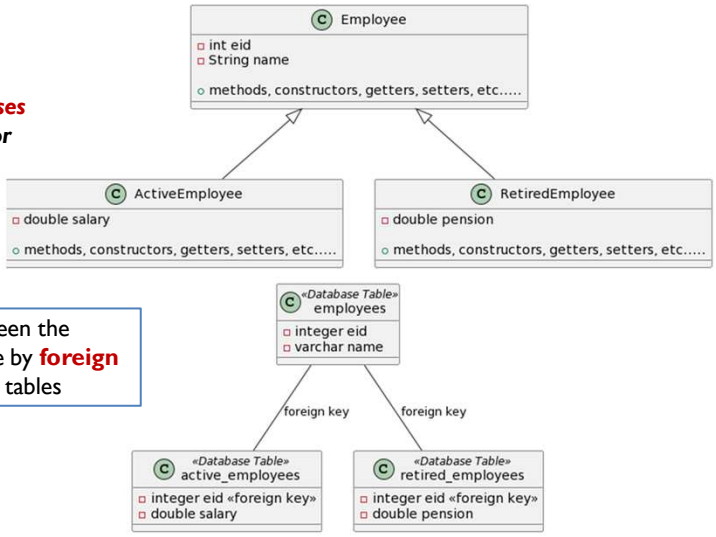
There's only a single table to worry about on the database. There are no joins in retrieving data. Any refactoring that pushes fields up or down the hierarchy doesn't change the DB.

Fields are sometimes relevant and sometimes not, which can be confusing. Columns used only by some subclasses lead to wasted space in the database. The single table may end up being too large.

17

Class Table Inheritance

Represents an inheritance hierarchy of classes with one table for each class.



The **linking** between the tables can be done by **foreign keys** between the tables

Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/classtable.inheritence

18

Class Table Inheritance

Represents an inheritance hierarchy of classes with one table for each class.

All columns are relevant for every row so tables are easier to understand and don't waste space.

The relationship between the domain model and the database is straightforward.

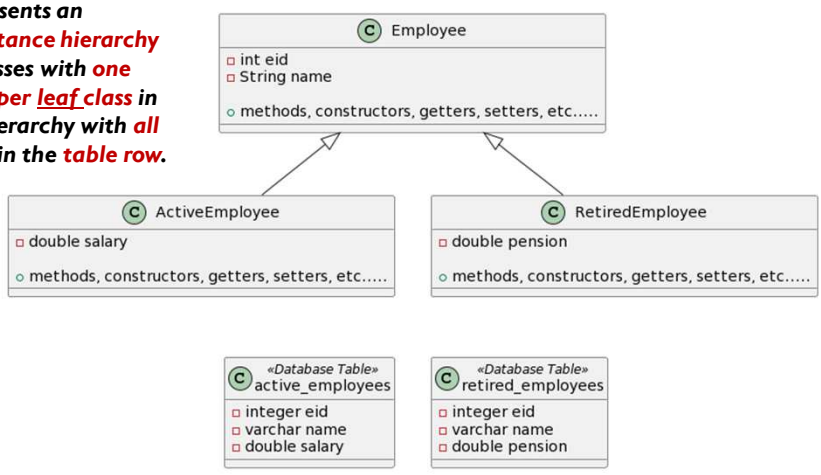
Need to touch multiple tables to load an object, which means a join or multiple queries and sewing in memory.

Any refactoring of fields up or down the hierarchy causes database changes.

19

Concrete Table Inheritance

Represents an inheritance hierarchy of classes with one table per leaf class in the hierarchy with all fields in the table row.



Example:
https://github.com/zarras/myy803-EAAP-tutorials/tree/main/concretetable.inheritence

20

Concrete Table Inheritance

Represents an inheritance hierarchy of classes with one table per leaf class in the hierarchy with all fields in the table row.

Each table is self-contained and has no irrelevant fields.

There are no joins to do when reading the data from the concrete mappers.

Each table is accessed only when that class is accessed, which can spread the access load.

Redundancy in the data

With fields are pushed up or down the hierarchy, you don't have to alter the table definitions.
If a superclass field changes, you need to change each table.

21