Application Program Development

Segment : ORM's

Mahboob Ali

Objectives

- UML
- Object-Relational Patterns
- Persistence

Domain-Driven Design (DDD)

- ❑DDD is one of the focused software development philosophies that overcome the complexities of the problem.
- ❑Domain: is the knowledge, influence or activity around which we work to solve the problem on hand.
- *Example problem:* In a banking originations engine, you need to determine how to create a loan application? and how that application will be processed through the system?
- Domain → in this case is the whole of the banking and originations process.
  - We need to **model** the domain in order to solve the problem.
- *Domain Model:* an object model of the domain that incorporates both **behavior** and **data.**

UML (Unified Modeling Language)

- The most common language used for **modeling**.
- There is total 13 diagrams in UML used as references
  - Object Diagram
  - Class Diagram

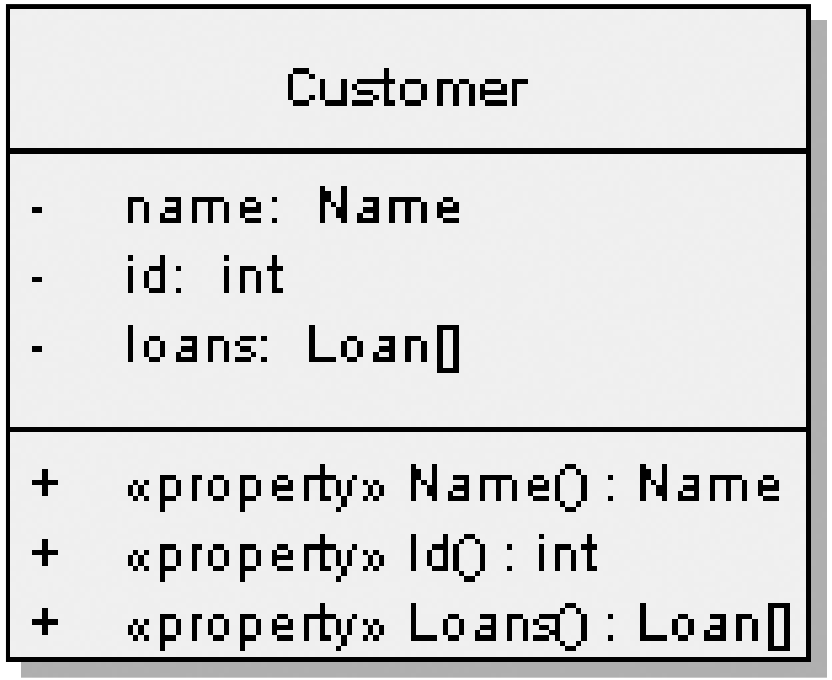
Class Diagram

- Used to depict and capture logical structure.

- Displays static model with all the attributes and

relationships between classes and interfaces.

- Can also include
  - Inheritance
  - Composition
  - Associations
  - Aggregations

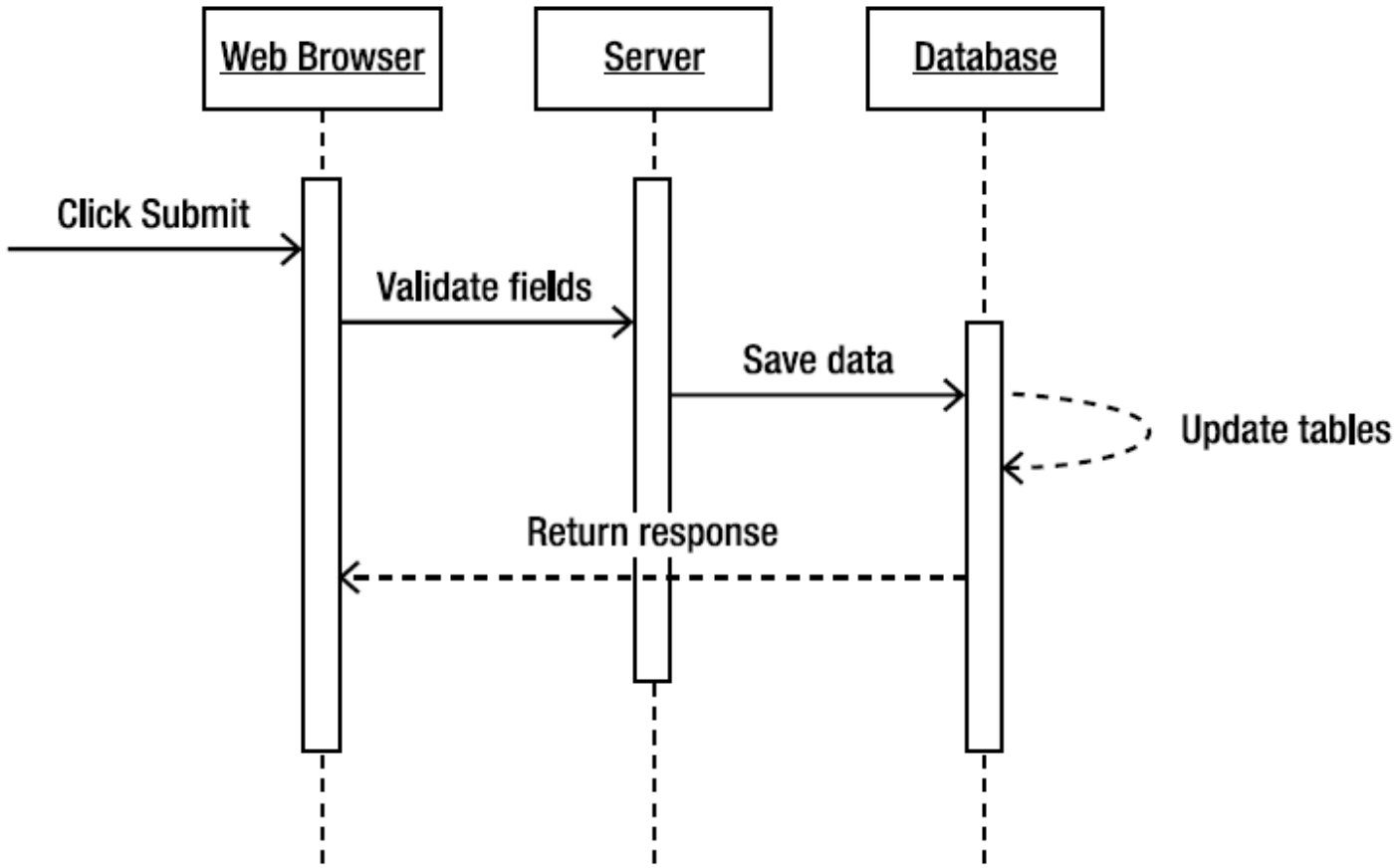


Object Diagram

- Is a simplified version of a class diagram
- Depicts the role of an object plays when **instantiated** and **its multiplicity.**

- In simple words both of these diagrams shows the relationships between classes, interfaces and object role.

Sequence Diagram



- Represent stereotyped elements, such as screens, controllers, entities, and database items.

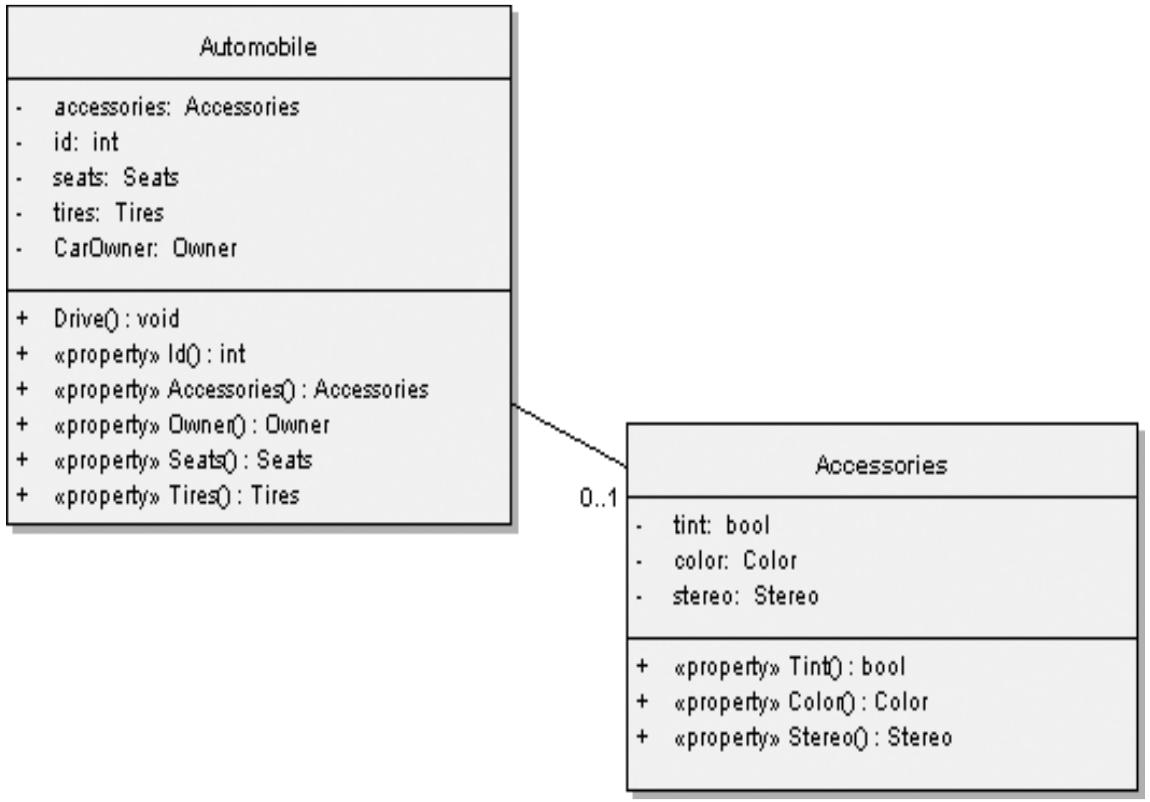
Domain Model Structure

- A domain model can be broken into these pieces
  - Entities
  - Value Objects
  - Services
  - Aggregates
  - Factories
  - Repositories

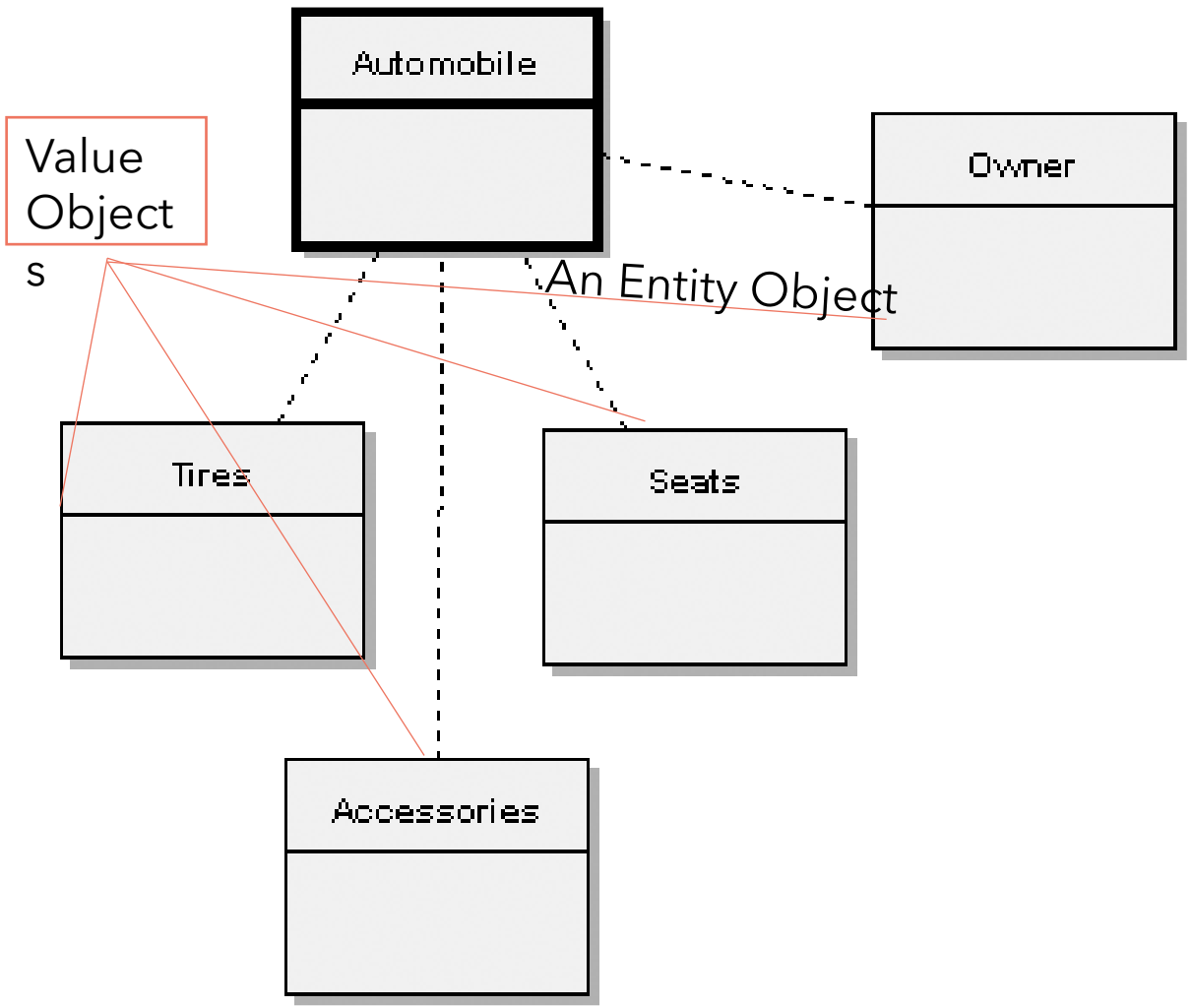
Entities

- Tech definition: In Database *entity* is a table.
- Non-tech definition: Is an *independent, separate* or *self-contained existence*.
- According to the non-tech side of definition looks like *entity* is something real, but the question becomes …
- Can one define a real-existence in terms of software development?
- Identity: which defines uniqueness not only in real world but also in software world.
- Example: Car has a VIN. But you don't refer to your car with the VIN (unless you have a serious problem ☺). You give your car an identity with some attributes like color, model, engine etc.
- Your car *object* which has its identity becomes an entity.

Value Objects



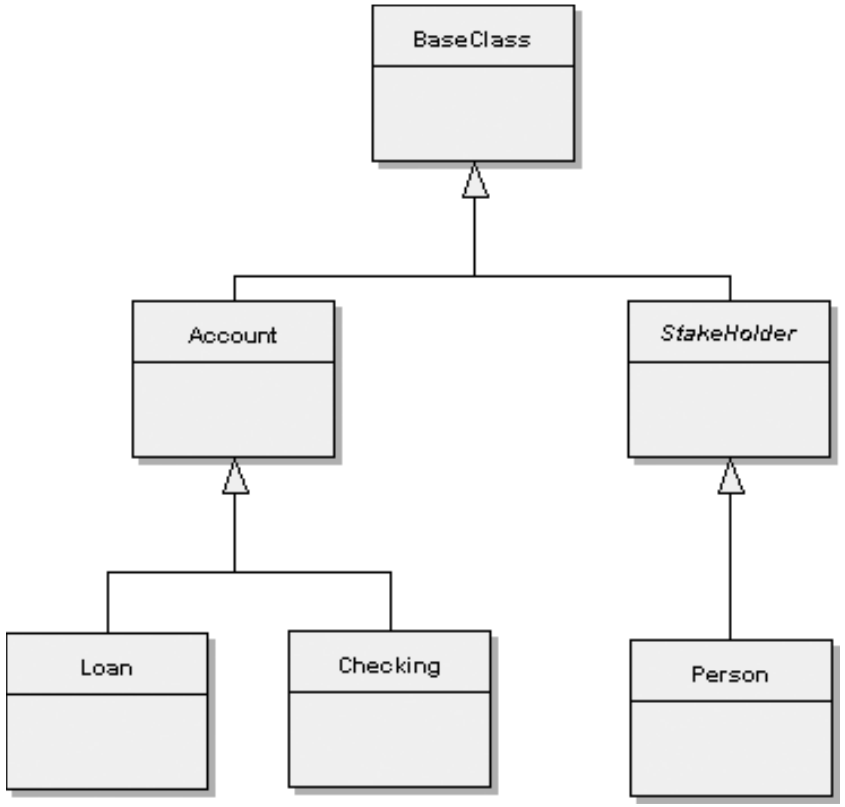
- Value object that is what responsible for storing and manipulating the data of the model.
- They can't exist by themselves.
- Example: Your attributes, like model, engine, color etc. can't exist on their own.
- A value object is typically an immutable object with no identity, containing values used to describe entities.



Aggregate

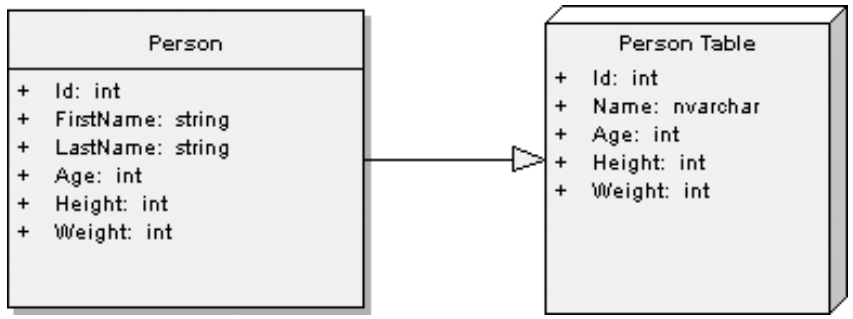
- *Aggregates*, in terms of the domain model, are just that: a logical grouping of entities and value objects into a collection. So that complex relationships that can lead to data inconsistencies in your application and database can be reduced.

Object-Relational Patterns



- **Domain Model**
- *A Domain Model creates a web of interconnected objects, where each object represents some meaningful individual, whether as large as a corporation or as small as a single line on an order form.*

Object-Relational Patterns

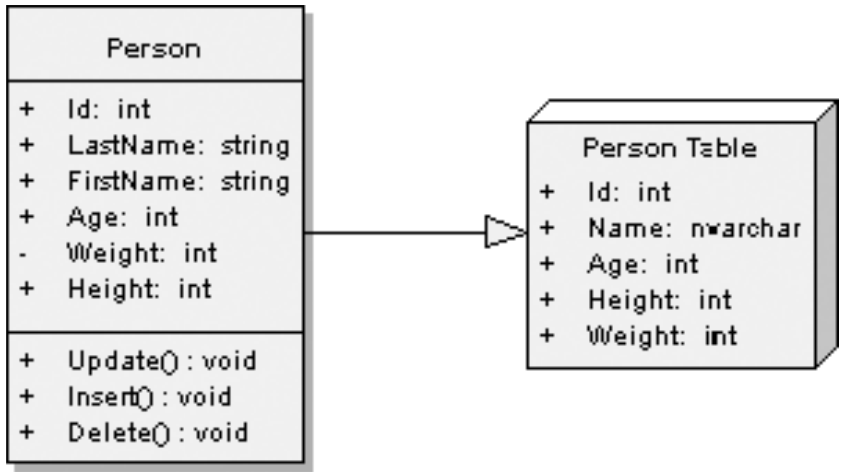


- **Table Module**

➢ The purpose of the table module is to create a class per table, and in many cases when not using a mapper, you create or use some sort of disconnected container to hold the tables and build relationships.

➢ Code generators and most ORM tools throughout the .NET Framework utilize this technique

Object-Relational Patterns

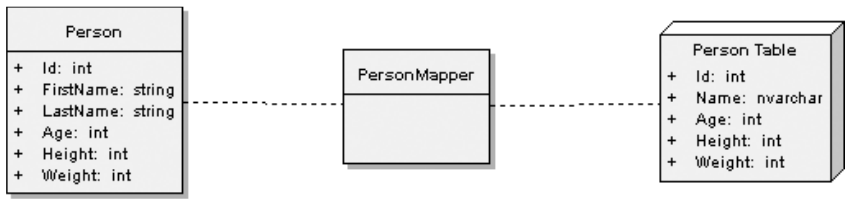


- **Active Record**

➢ An *active record* is similar to the table module except that the table module is an architectural design principle, whereas an active record falls under the data access pattern category (like database mapper).

➢ The active record pattern also encapsulates database access functionality.

Object-Relational Patterns



- **Database Mapper**

➢ The essence of the DB mapper, is to map objects to a relational database.

➢ Accomplished by using *metadata* to describe the attributes and relationships that exist between the object model and the data model.

➢ Metadata can come in many forms, but for the most part Extensible Markup Language (XML) is a consistent option for the metadata mappings that the ORM uses.

➢ Also a mapping engine of some sort is needed to translate the metadata into something useable.

➢ EF and LINQ to SQL are both examples of DB mappers.

DB Mapper

- Inheritance

➢ The most important part in ORM is inheritance

➢ There are three key inheritance-mapping approaches

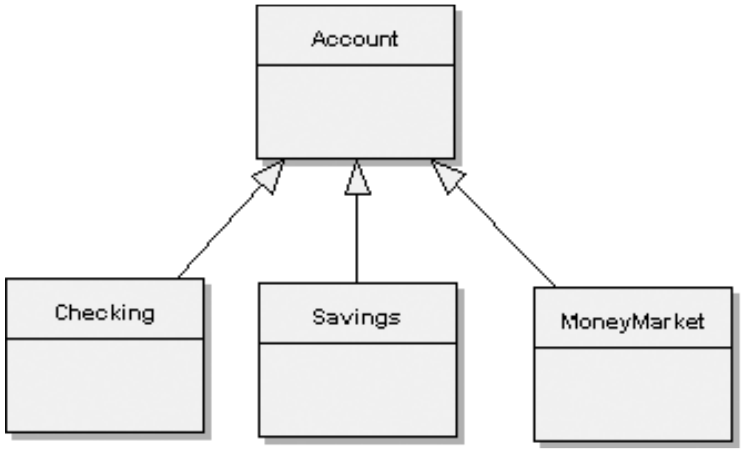
  ➢ A single table per class hierarchy

  ➢ A table per concrete class

  ➢ A table class to its own table

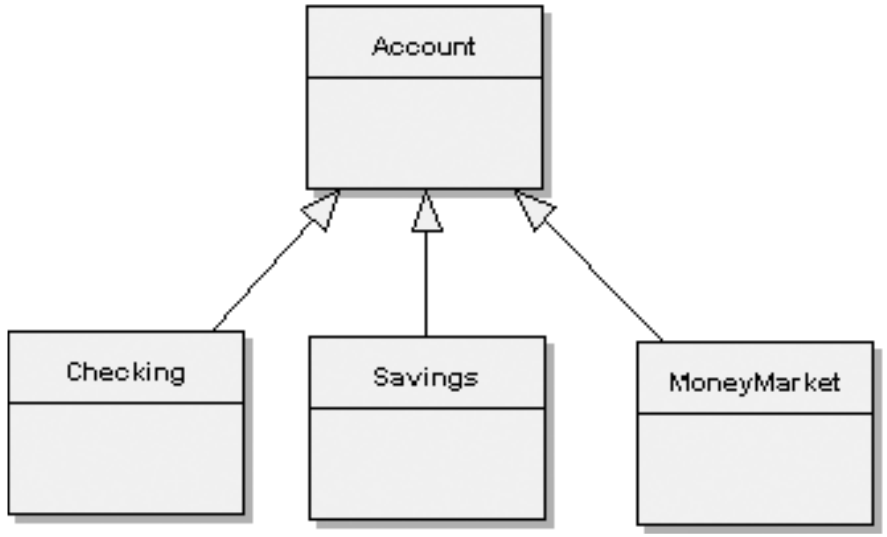
Note: Not all these approaches are fully supported in EF or LINQ to SQL

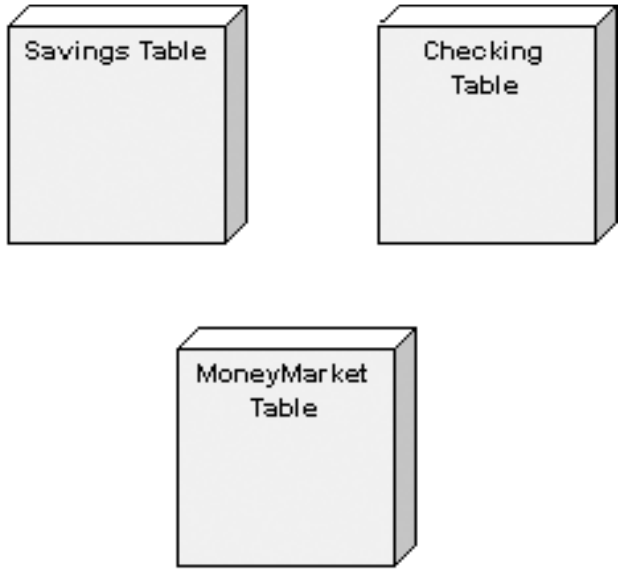
Table per class Hierarchy



- Mapping your entire class structure to a single table.
- Quick and easy approach no complex relations or queries needed.
- If you have more than one types in your class model then this approach becomes very complex as you have to introduce foreign key.

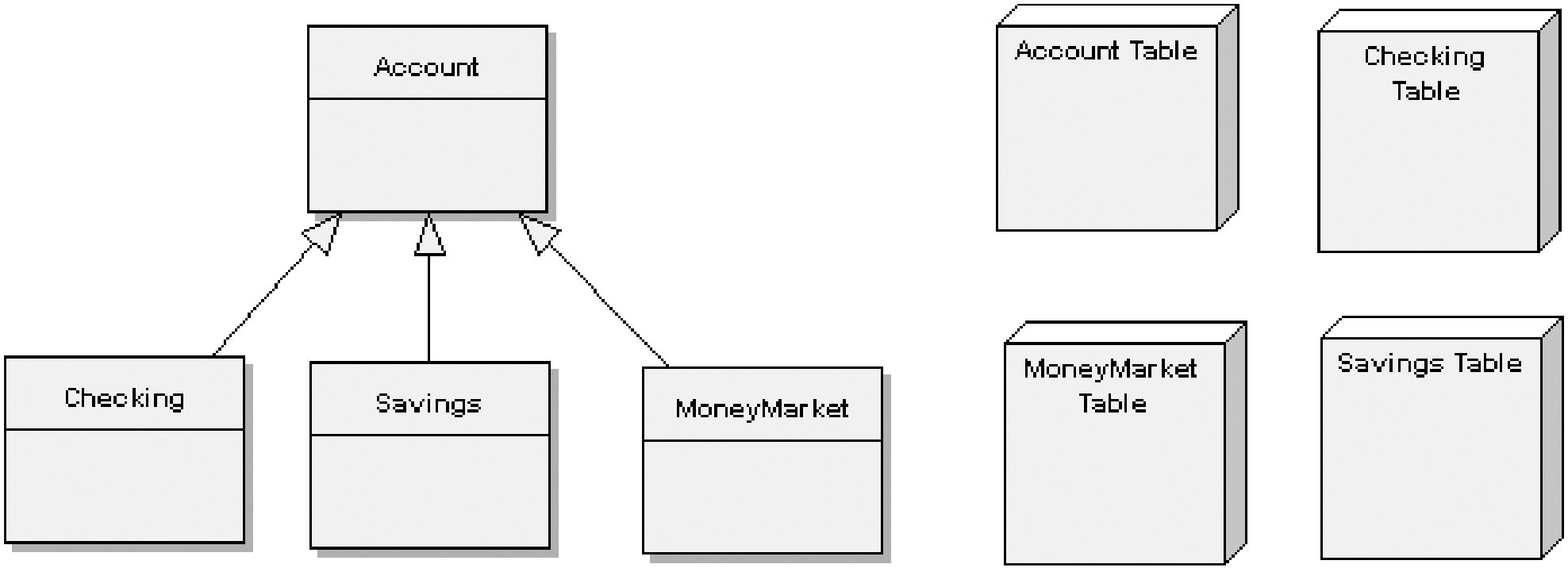
Table per concrete class



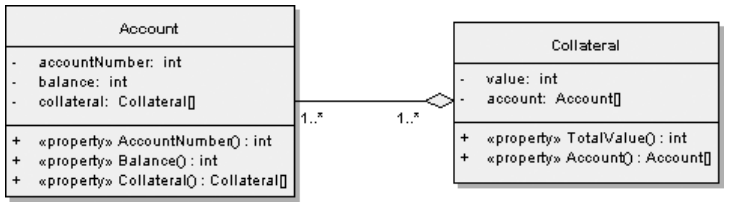


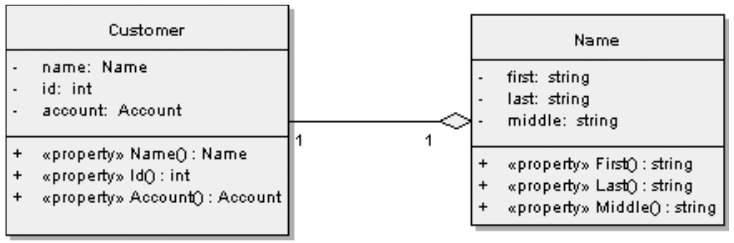
- The most common approach used to handle inheritance in ORM
- This technique maps each non-abstract class to its own table in the database

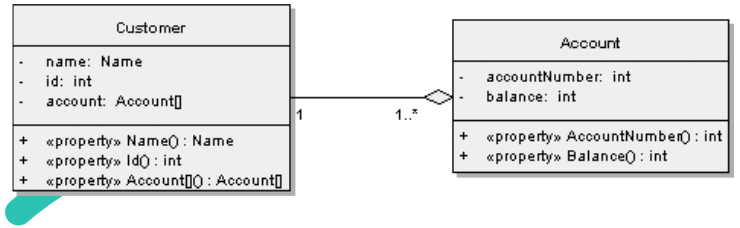
Table per class



Relationships







- Handling relationships in your domain model with an ORM can be a tricky task because of the inherent mismatch between the relational database and your object code.
- *One-to-One*
- *One-to-Many*
- *Many-to-Many*

Laziness/ lazy loading/ lazy initialization

- **Lazy Loading**
- A way to optimize memory utilization of database servers by prioritizing components that need to be loaded into memory when a program is started.
- Example: You need to cook multiple dishes for the new year dinner with your family and you know your kitchen is small. Your first option is that you can take out all the ingredients for all the dishes you're making and fill up your counter space, or you can use an on-demand method, and take out only the ingredients you need as you need them.