# Application Program Development

## APD545

Instructor: Maryam Sepehrinour
Email: Maryam.Sepehrinour@SenecaPolytechnic.ca

# Outcomes

- ✓ Introduction to ORM
- ✓ ORM and Design Pattern

# Outcomes

- ✓ **Introduction to ORM**
- ✓ ORM and Design Pattern

# What is ORM?

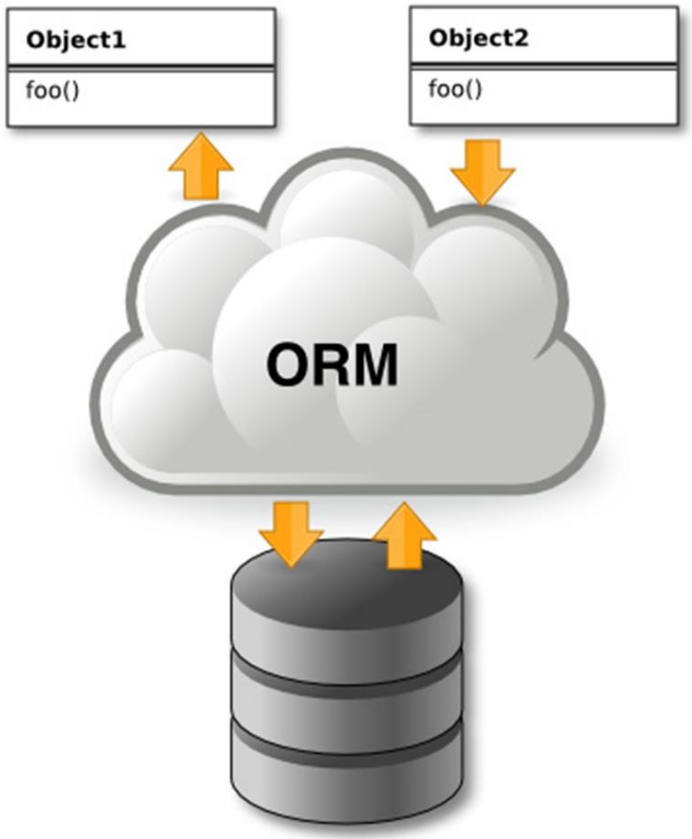
- Object-relational mapping is a programming technique
  - for converting data between incompatible type systems
  - by creating objects using OO programming languages
  - essentially wrapping tables or stored procedures in classes and interact with database using their methods and properties
- ORM is an automated way of
  - connecting an object model, sometimes referred to as a domain model, to a relational database by using metadata as the descriptor of the object and data.
- This creates, in effect, a "virtual object database

# What is ORM?

- ORM is the act of
  - connecting object code, whether it is in C#, Java, or any other object-oriented language, to a relational database.

- This act of mapping is an efficient way to overcome the mismatch that exists between object-oriented development languages and relational databases.
  - Such a mismatch can be classified as an inequality between the native object-oriented language operations and functions and those of a relational database

# What is ORM?




Introduction to ORM

# Relational Models vs. OO Models

- Types (ex. char(n) versus string)
- Identity (keys versus address or equality)
- Relations (foreign keys versus references)
- Many to many relations require linking table in SQL
- Inheritance and polymorphism

# Benefits of ORM?

- Automates the Object-to-table and Table-to-object conversions.
- work in the OO model
  - without having to worry about the underlying data structure
- Allow us to easily save objects to the database and load them from the database
- Typically provide support for the full set of CRUD (create, read, update, delete) operations
- Save time and money (time to market)
- Focus on the business logic
  - rather than database/persistence logic

# Selecting a suitable ORM

- Object-to-database mapping
  - The single most important aspect of an ORM tool.
  - Ability to map business objects to back-end database tables.
- Object cashing
  - Ability to enable object/ data cashing to improve performance.
- GUI mapping
  - Can survive without it but it will be bonus to have in an ORM tool.

# Selecting a suitable ORM (cnt.)

- Dynamic Querying
  - Another important aspect in an ORM tool, which allows the different projection and classless queries based on the user input.
  - LINQ-to-SQL and EF both support this.
- Lazy loading
  - Improves the performance by optimizing the memory utilization of database server.
  - This is done by prioritizing the components, should they be loaded on first come first bases or pre-loaded as the program starts.

# Selecting a suitable ORM (cnt.)

- Nonintrusive persistence
  - This is an important one, meaning no more class, interface or inheritance based specific relations.

# Outcomes

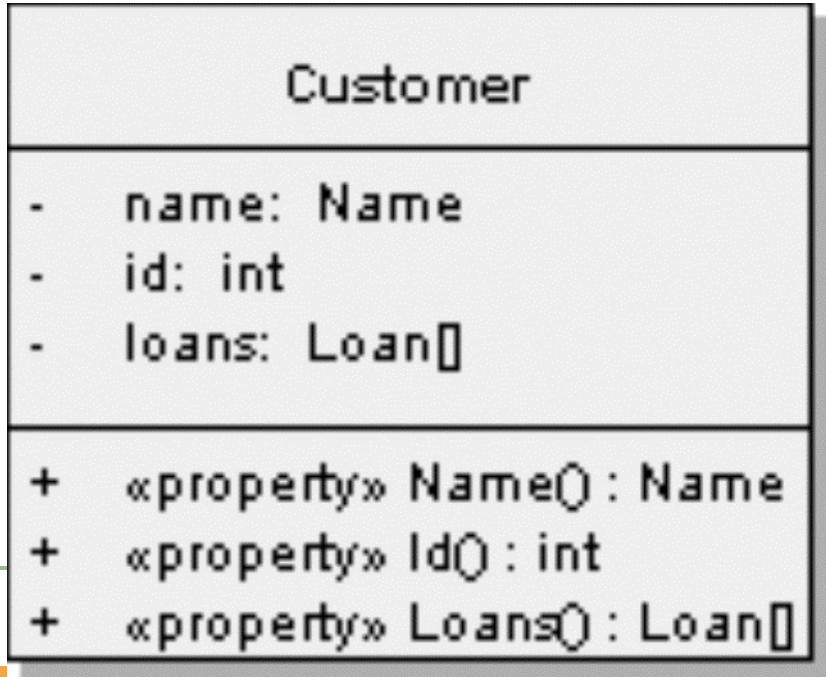
- ✓ **Introduction to ORM**
- ✓ **ORM and Design Pattern**

# UML (Unified Modeling Language)

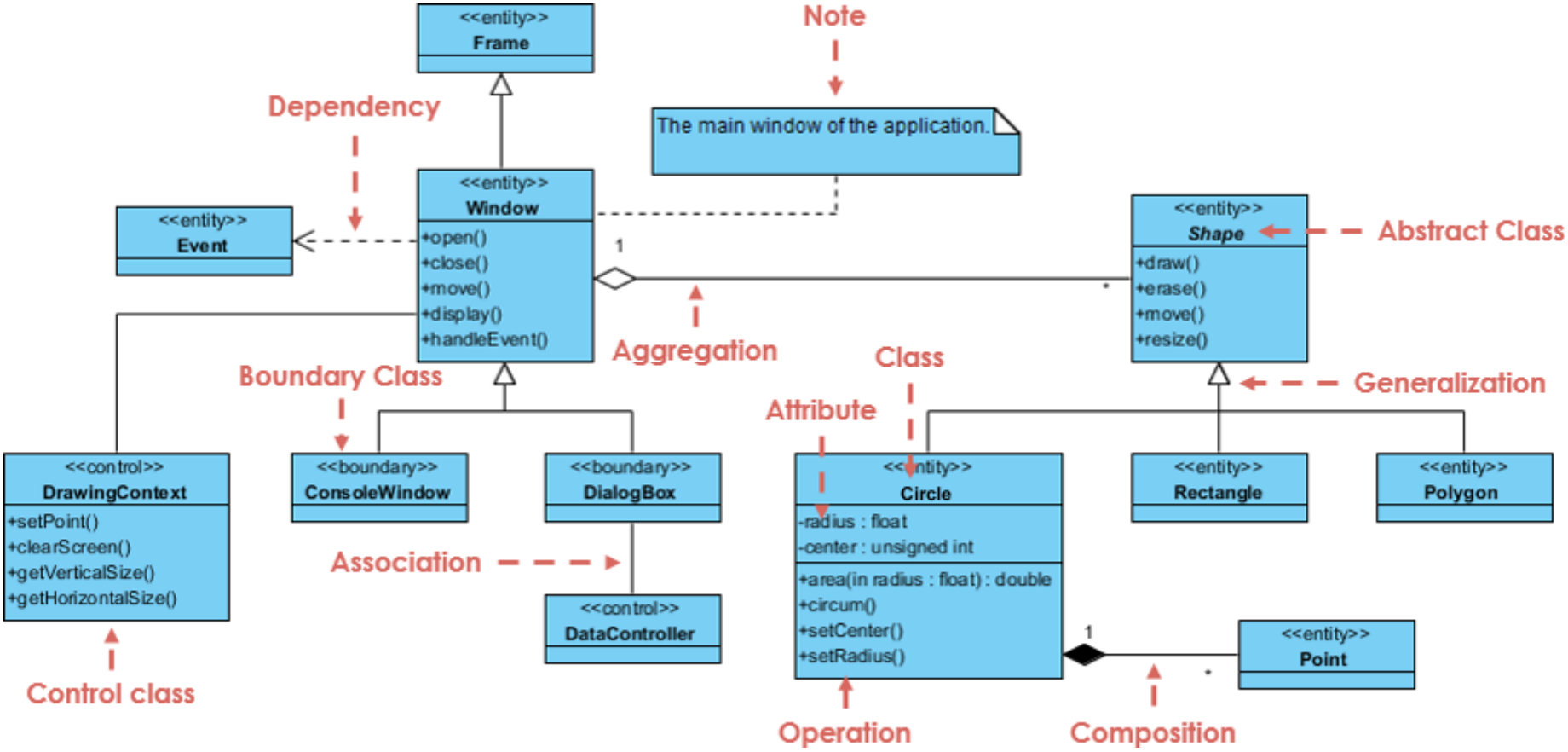
- The most common language used for modeling.
- There is total 13 diagrams in UML used as references
  - Class Diagram
  - Object Diagram
  - Sequence Diagram
  - …..

# 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

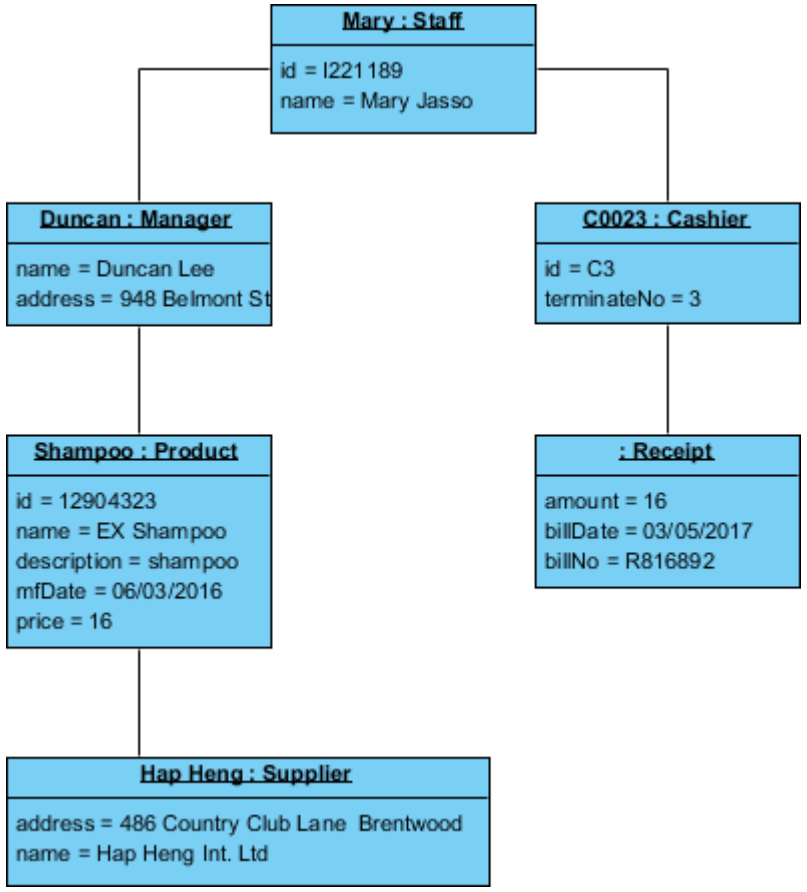


# Class Diagram



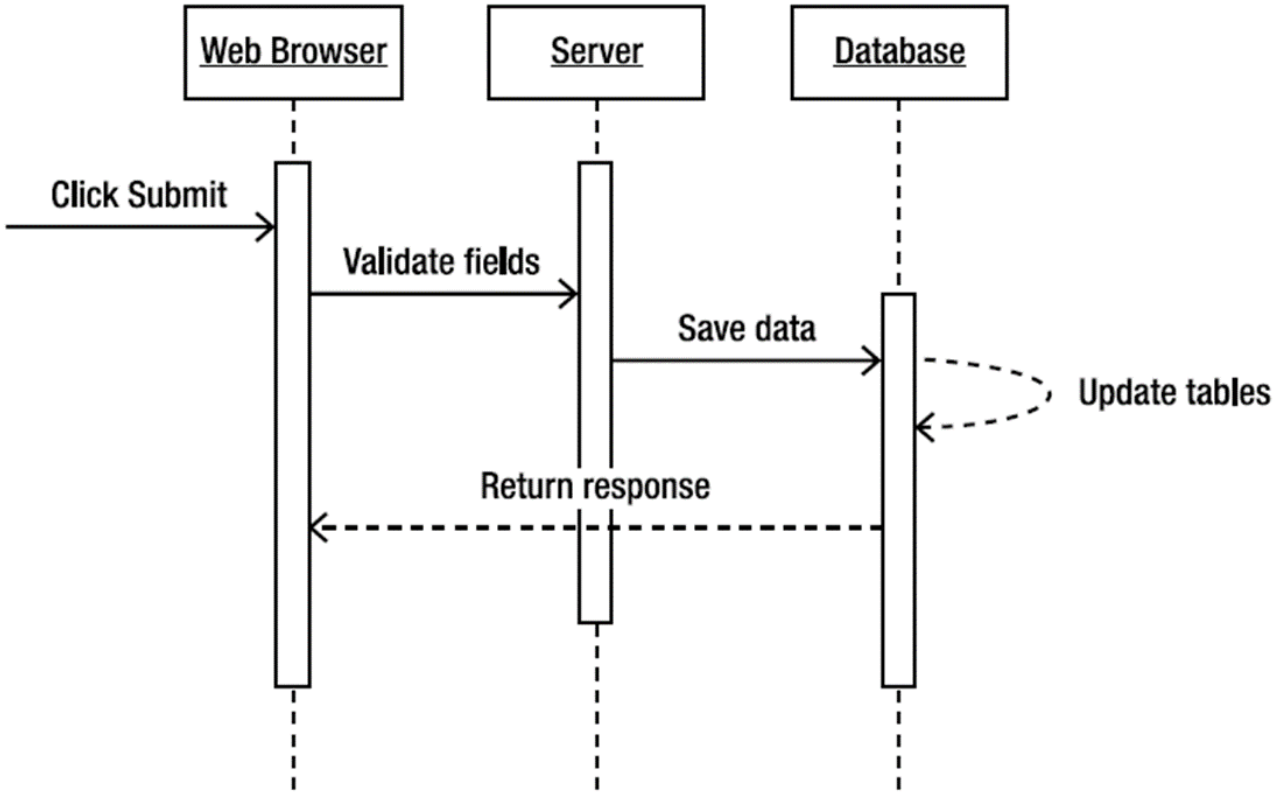
# Object Diagram

- 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.



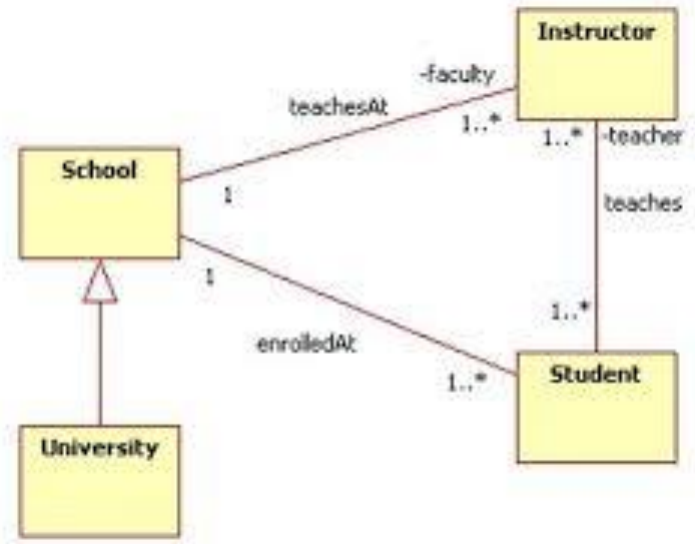

Senec

# Domain Model

- A domain model is a conceptual model of the domain that incorporates both behavior and data. where each object represents some meaningful individual, whether as large as a corporation or as small as a single line on an order form.

➢ A domain model can be broken into these pieces

- ➢ Entities
- ➢ Value Objects
- ➢ Services
- ➢ Aggregates
- ➢ Factories
- ➢ Repositories



# Entities

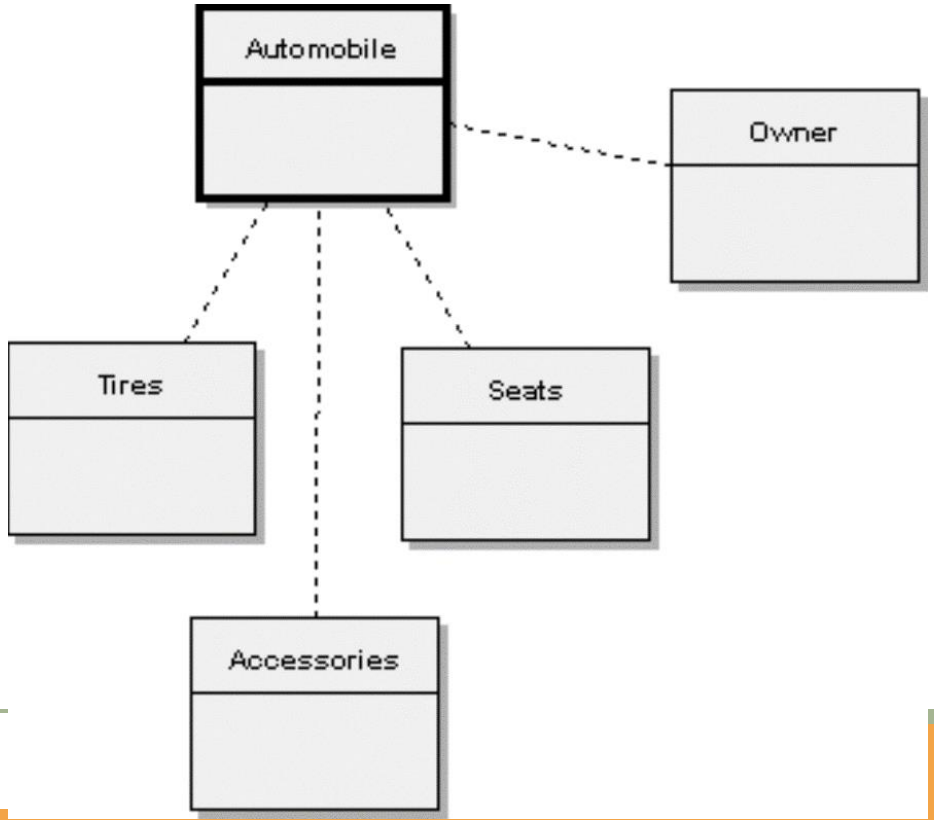
- Tech definition: In Database entity is a table.
- Non-tech definition: Is an independent, separate or self-contained existence.
- Identity: which defines uniqueness not only in real world but also in software world.
- Example: Car has a VIN. But you don't refer to your car with the VIN (unless you 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.

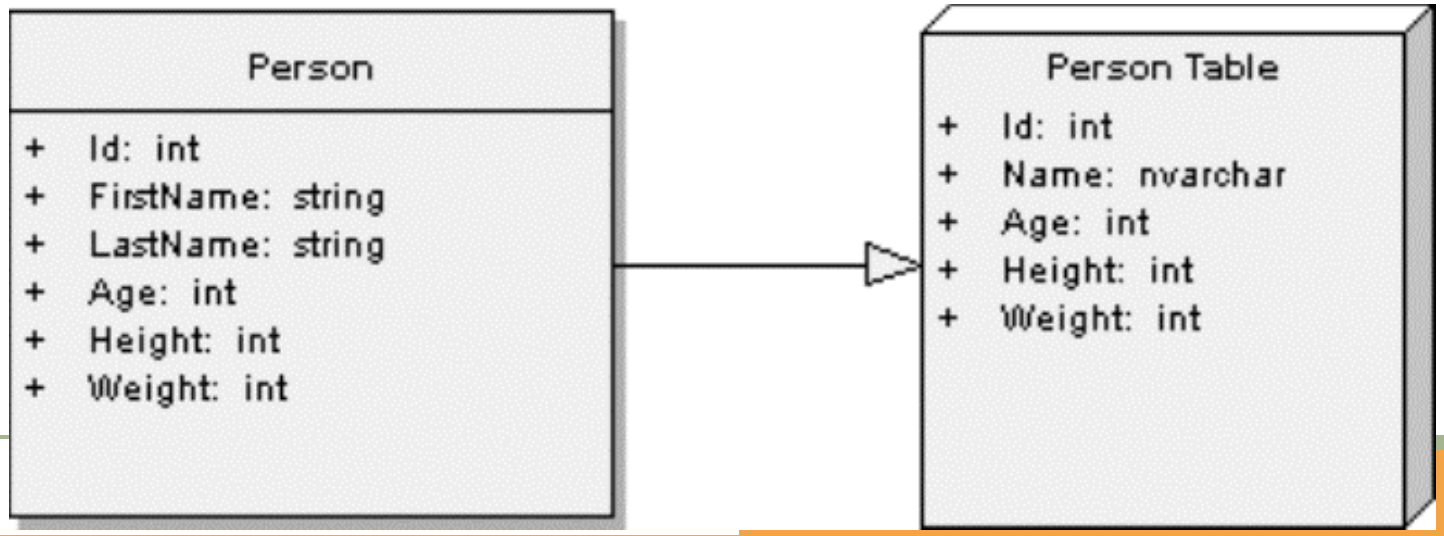
# Aggregate

- 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.



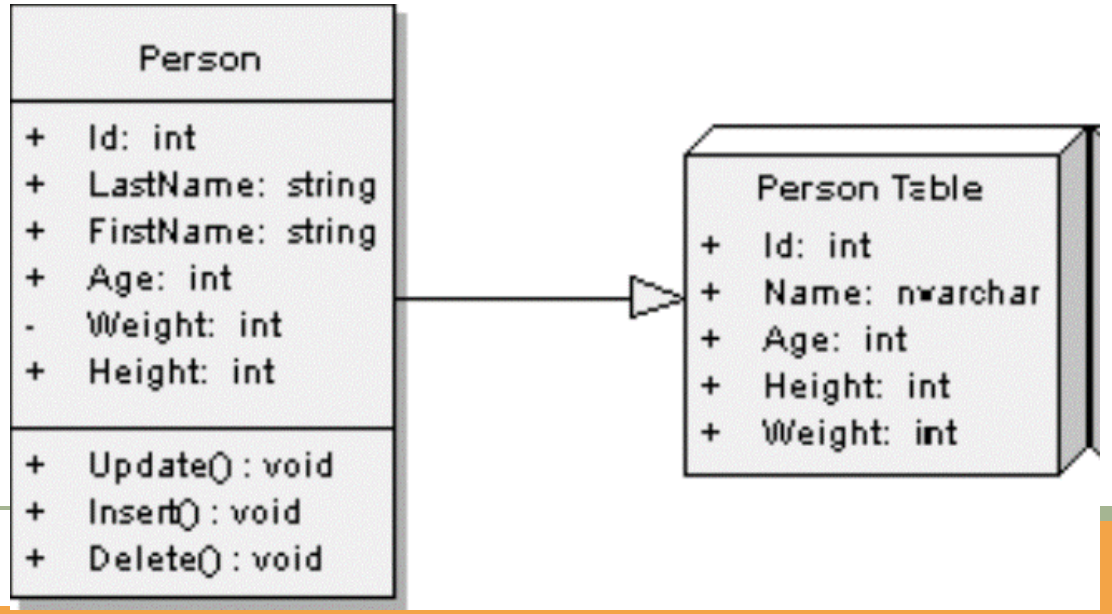
# Object-Relational Patterns

- 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

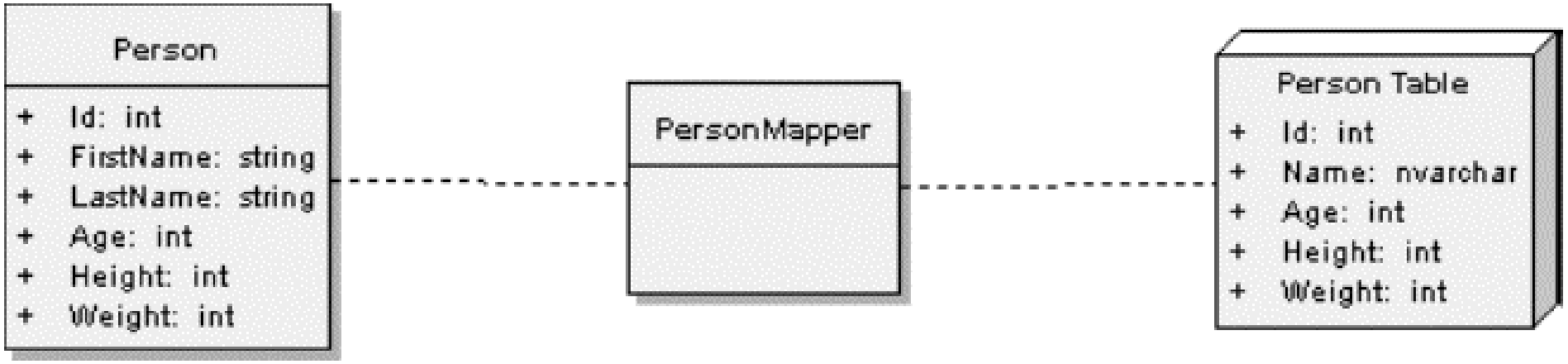
- Active Record
  - An active record is similar to the table module except that the table module is an architectural design principle, whereas an active record falls under the data access pattern category (like database mapper).
  - The active record pattern also encapsulates database access functionality.



# Object-Relational Patterns

- DB Manager
  - 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.

# Object-Relational Patterns

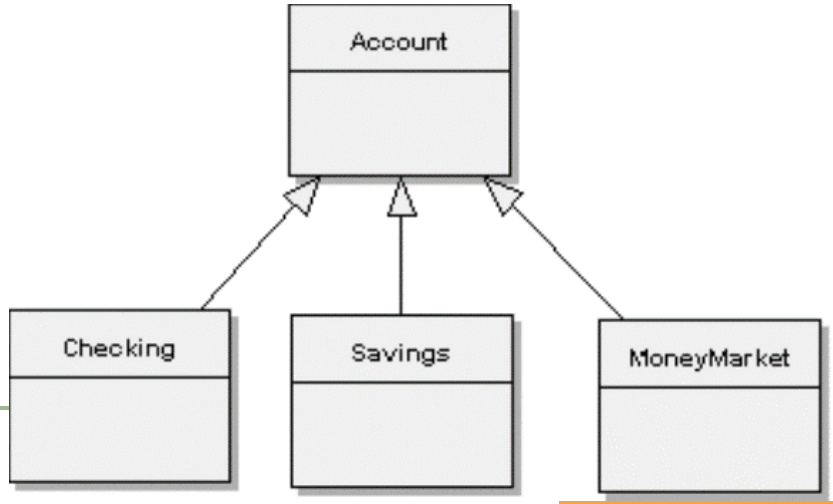


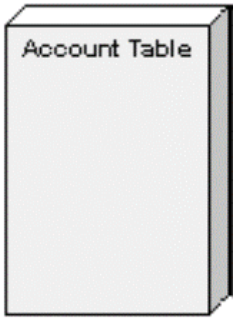
# DB Mapper

- 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

# Table per Class Hierarchy

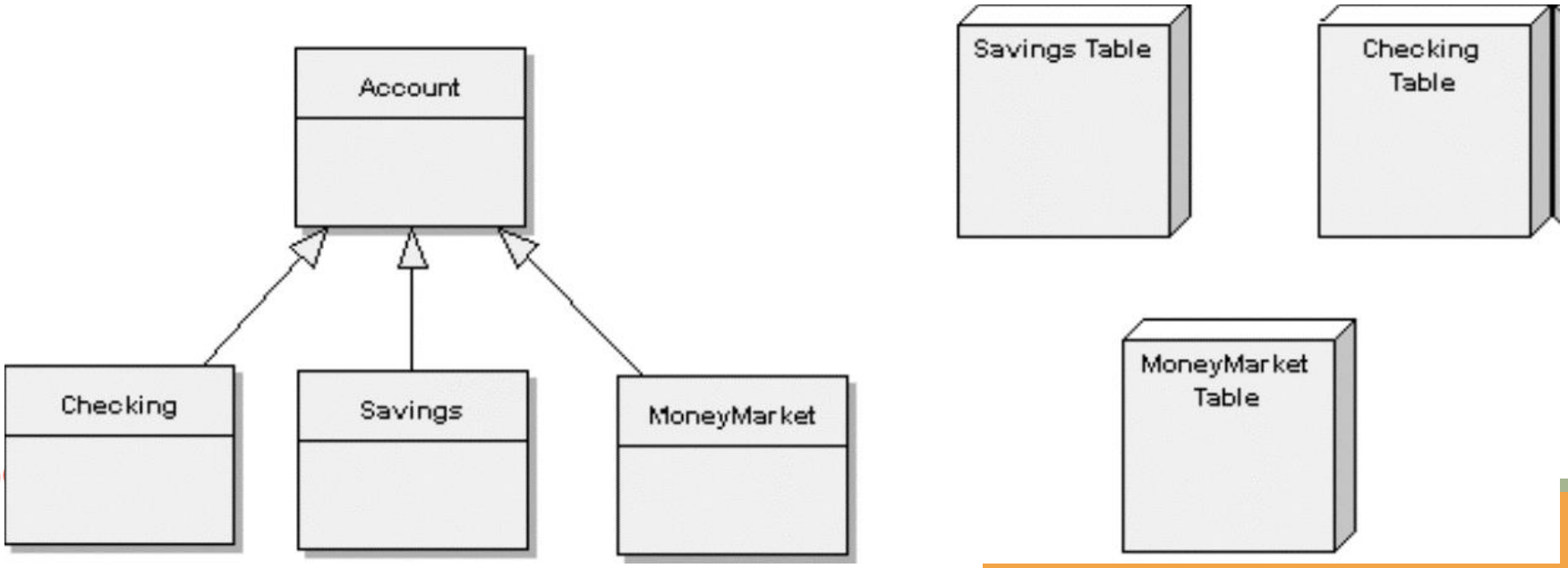
- Mapping your entire class structure to a single table.
- Quick and easy approach no complex relations or queries needed.
- If you have more than one types in your class model then this approach becomes very complex as you have to introduce foreign key.



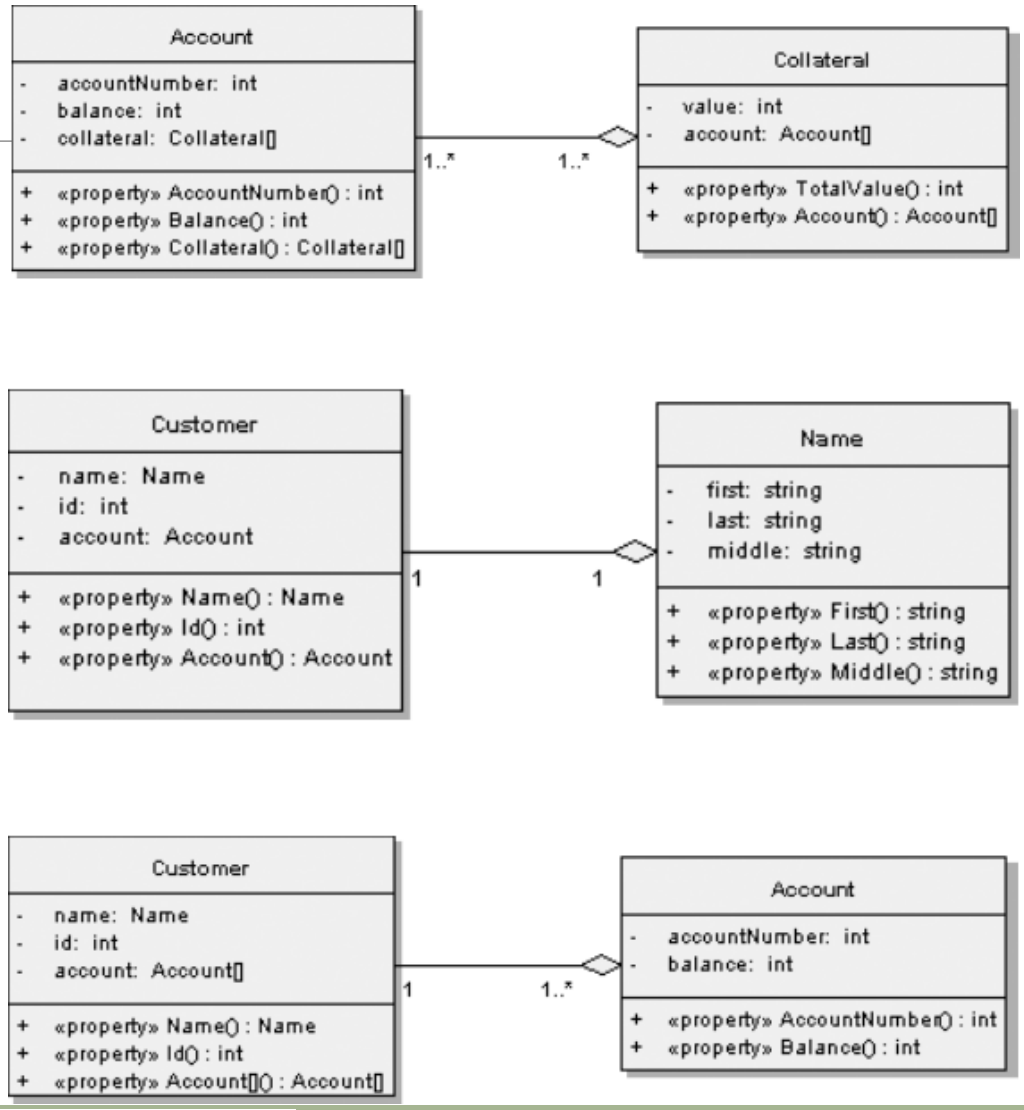
# Table per Concrete Class

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



# 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.

# Thank you!