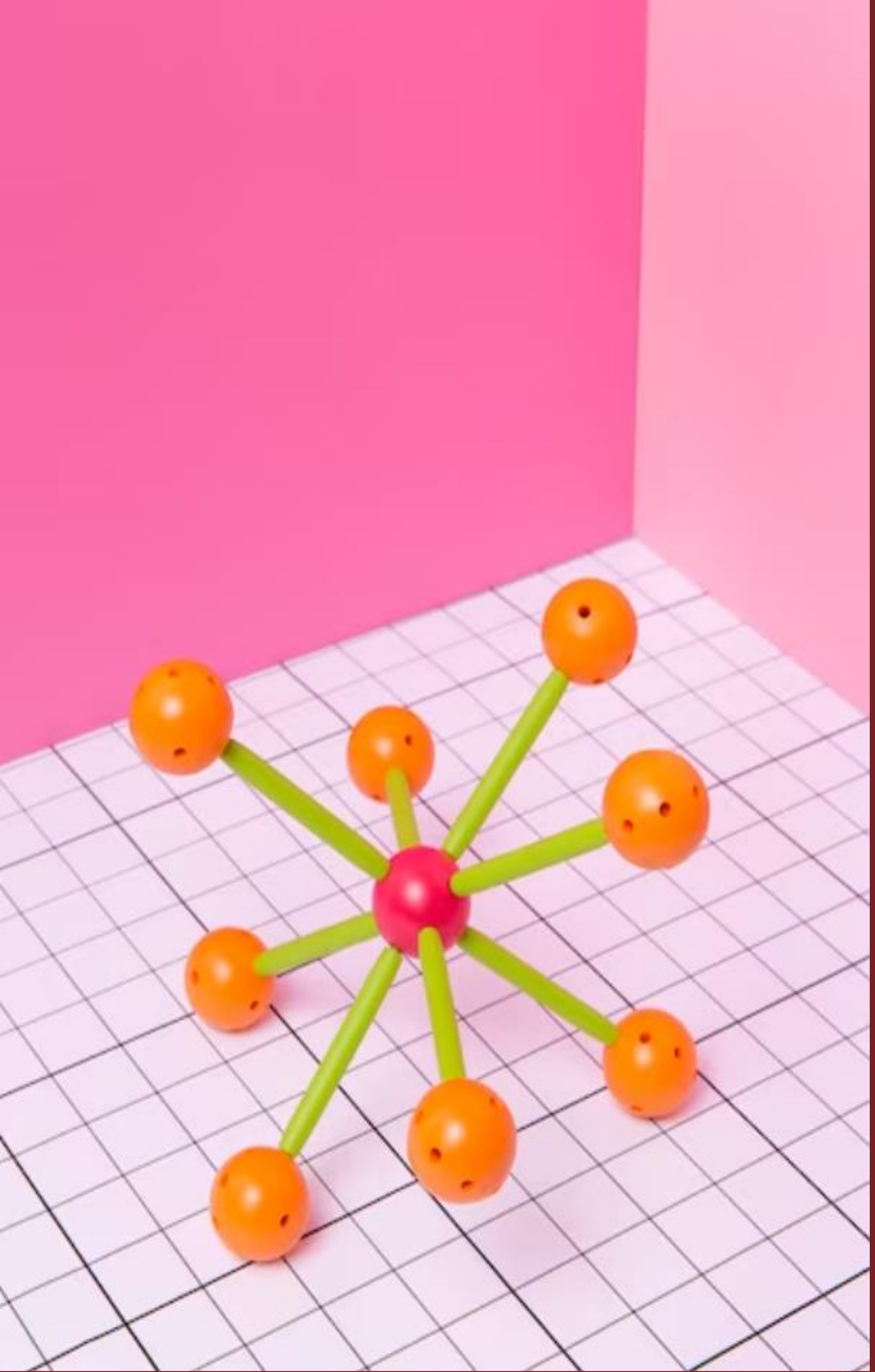
Object Oriented Programming Fundamentals

AGENDA

- Identifying classes from requirements
- Separating responsibilities.
- Defining relationships between classes
- Lavage reuse

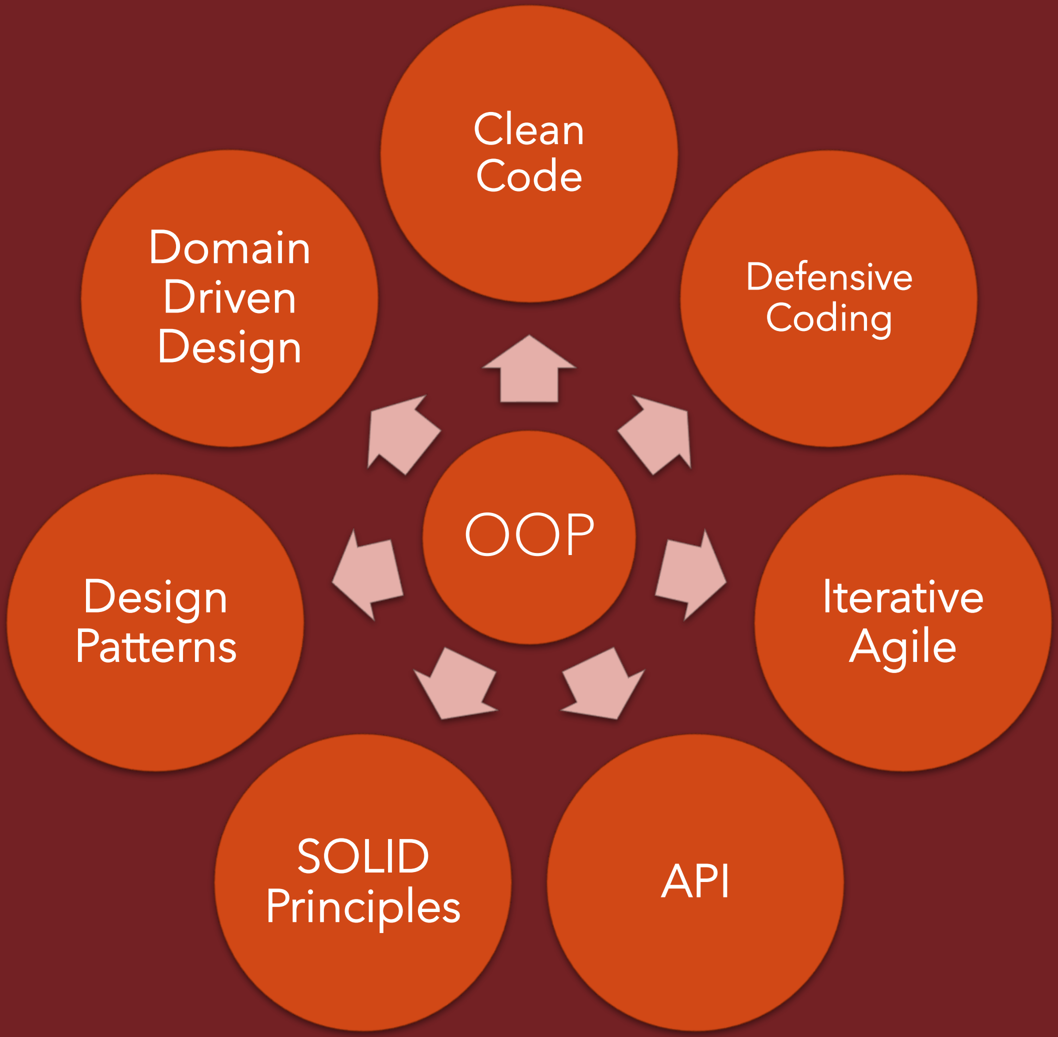


WHY OOP

The more you know about OOP, The more you can better leverage features of C# to build well-crafted, maintainable and testable application.

Code Reusability: Inheritance allows the reuse of code by creating derived classes from existing ones.

OOP Is the Foundation



```
public class Customer
{
    public int CustomerId { get; private set; }

    public string EmailAddress { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public bool Validate() {...}
}
```

We need to define the business objects

Business Object == Class

Entity

Customer Management System

Entity	Customer
Class	Customer • First Name • Last Name • Go On An Adventure
Objects	Mohammad Abo Tier Eslam Hamed

What Is Object Oriented Programming (OOP)

An approach to designing and building applications that are:

- Flexible
- Natural
- Well-crafted
- Testable

by focusing on objects that interact cleanly with another

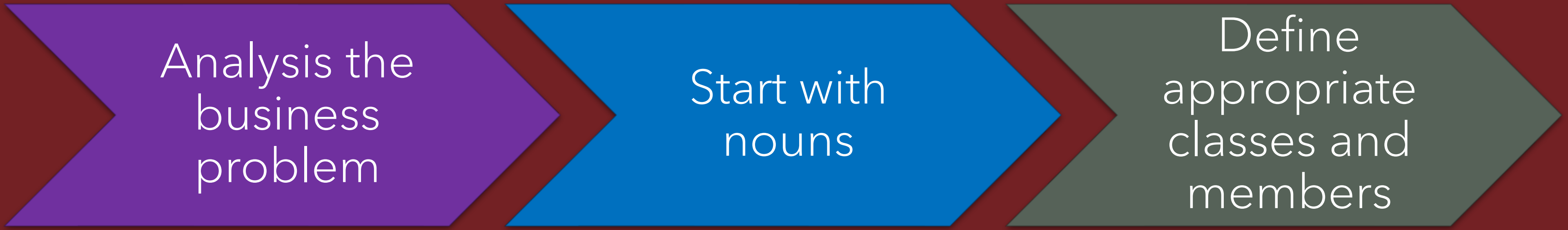
Identifying classes

Separating responsibilities

Establishing relationships

Leveraging reuse

How we extract classes from words


Analysis the business problem
Start with nouns
Define appropriate classes and members

Customer Management System

Manage regular, premium, VIP types of customer

Manage products

Receive orders from customers

Start with the Nouns

Customer

Manage regular, premium, VIP types of customer

Product

Manage products

Order

Receive orders from customers

Define Appropriate Members

- Customer’s name (First name, Last name)
- Contact information (phone number, email)
- address

- Product name
- Product description
- Product price

- Order date
- Total price
- Shipping address

Define Appropriate Members

Customer	Product	Order
• Name • Email • Phone • Address	• Name • Description • Price	• Order date • Shipping address • Customer • Product • Quantity

Define Appropriate Members

Customer

- Name
- Email
- Phone
- Address

Product

- Name
- Description
- Price

Order

- Order date
- Shipping address
- Customer
- Order items

Order Item

- Product
- Quantity

Define Appropriate Members

Customer

- Name
- Email
- Phone
- Address
- Validate()
- Retrieve()
- Save()

Product

- Name
- Description
- Price
- Validate()
- Retrieve()
- Save()

Order

- Order date
- Shipping address
- Customer
- Order items
- Validate()
- Retrieve()
- Save()

Order Item

- Product
- Quantity
- Validate()
- Retrieve()
- Save()

Consider Timing

Customer

- Name
- Email
- Phone
- Address
- Validate()
- Retrieve()
- Save()

Product

- Name
- Description
- Price
- Validate()
- Retrieve()
- Save()

Order

- Order date
- Shipping address
- Customer
- Order items
- Validate()
- Retrieve()
- Save()

Order Item

- Product
- Quantity
- Purchase price
- Validate()
- Retrieve()
- Save()

ABSTRACTION

- Abstraction is a fundamental concept in computer science that helps simplify complex systems, making them more manageable and easier to understand.
- It allows to focus on essential aspects while hiding intricate details.
- It aims to capture the essential functionalities and characteristics while hiding unnecessary details.
- The way you think about classes, and not a programming technique.

ENCAPSULATION

- Encapsulation is a key underlaying principle that makes to possible to build large, full-featured system by breaking complex operations into encapsulated units (classes).
- Encapsulation allows the objects in an application to work together without knowing the details of other object’s implementation.

BENEFITS OF HIDING DATA AND IMPLEMENTATION WITHIN THE CLASS

- Protects the data
- Allows for authorization before getting the data
- Allows for validation before setting the data

- Helps manage complexity by breaking methods down into manageable units
- Only the class understand the implementation.
- Implementation can be changing without impacting the rest of application.

COMMON APPLICATION LAYERS

User interface layer

Business logic layer

Data access layer

Common Code

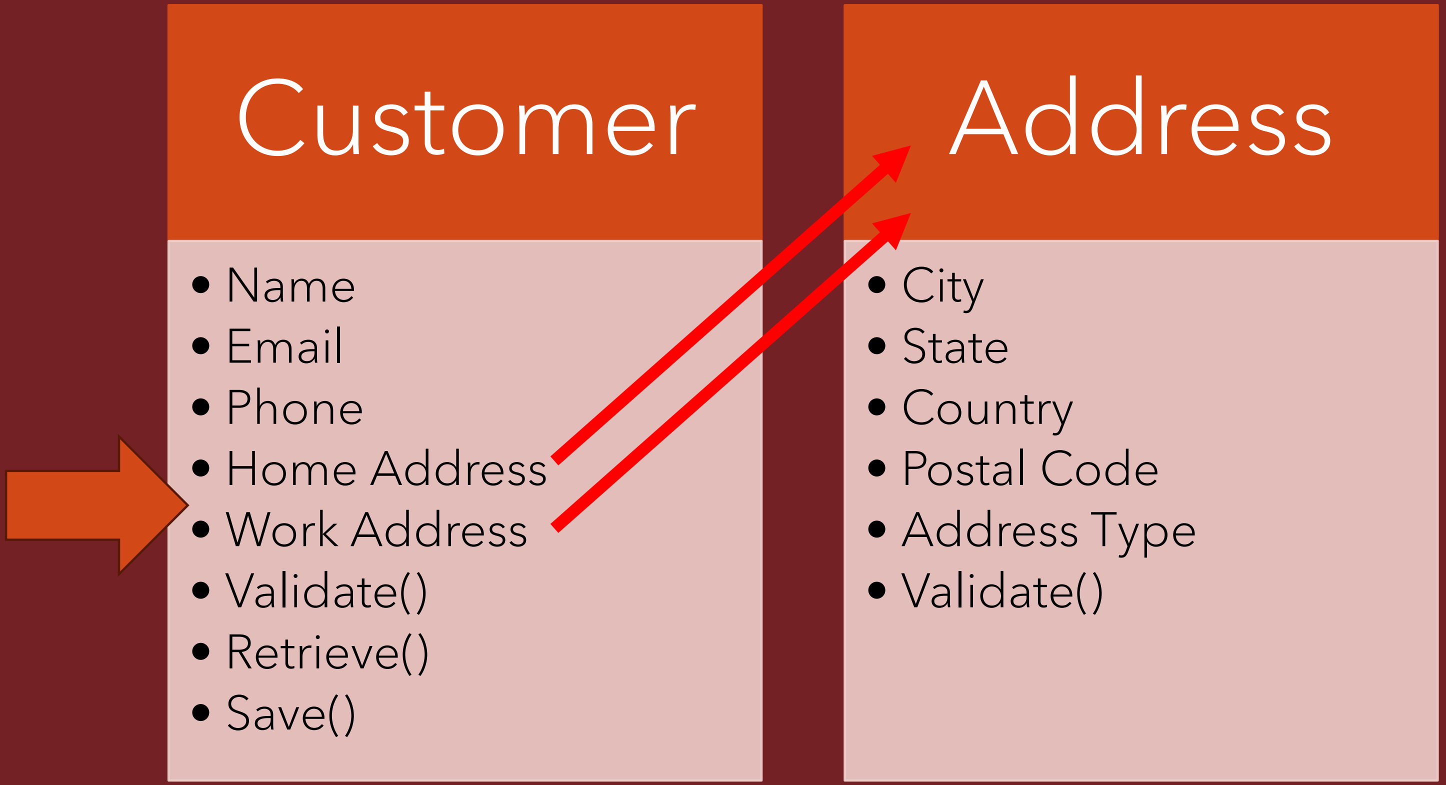
The time for writing some code

Separation of concerns

SEPARATION OF CONCERNS

- Minimizing coupling
- Maximizing cohesion
- Simplifies maintenance
- Improve testability

SEPARATING OF RESPONSIBILITIES



SEPARATING OF RESPONSIBILITIES

Customer Repository	Customer	Address
• Retrieve() • Save()	• Name • Email • Phone • Home Address • Work Address • Validate() • Retrieve() • Save()	• City • State • Country • Postal Code • Address Type • Validate()

SEPARATING OF RESPONSIBILITIES

Customer
- Name
- Email
- Phone
- Work Address
- Home Address
- Validate()

Product
- Name
- Description
- Price
- Validate()

Order
- Order date
- Shipping address
- Customer
- Order items
- Validate()

Order Item
- Product
- Quantity
- Purchase price
- Validate()

Customer Repository
- Retrieve()
- Save()

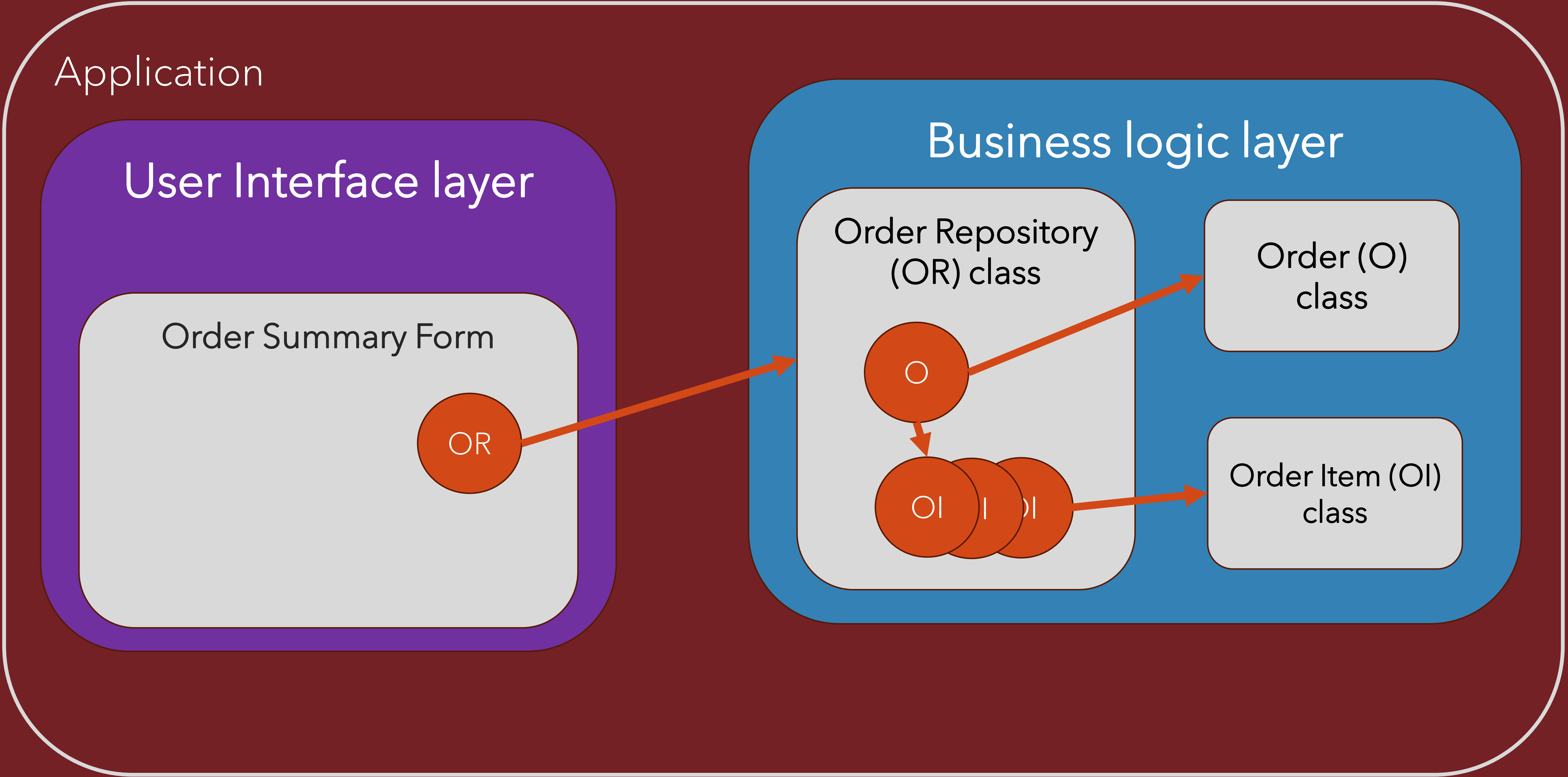
Product Repository
- Retrieve()
- Save()

Order Repository
- Retrieve()
- Save()

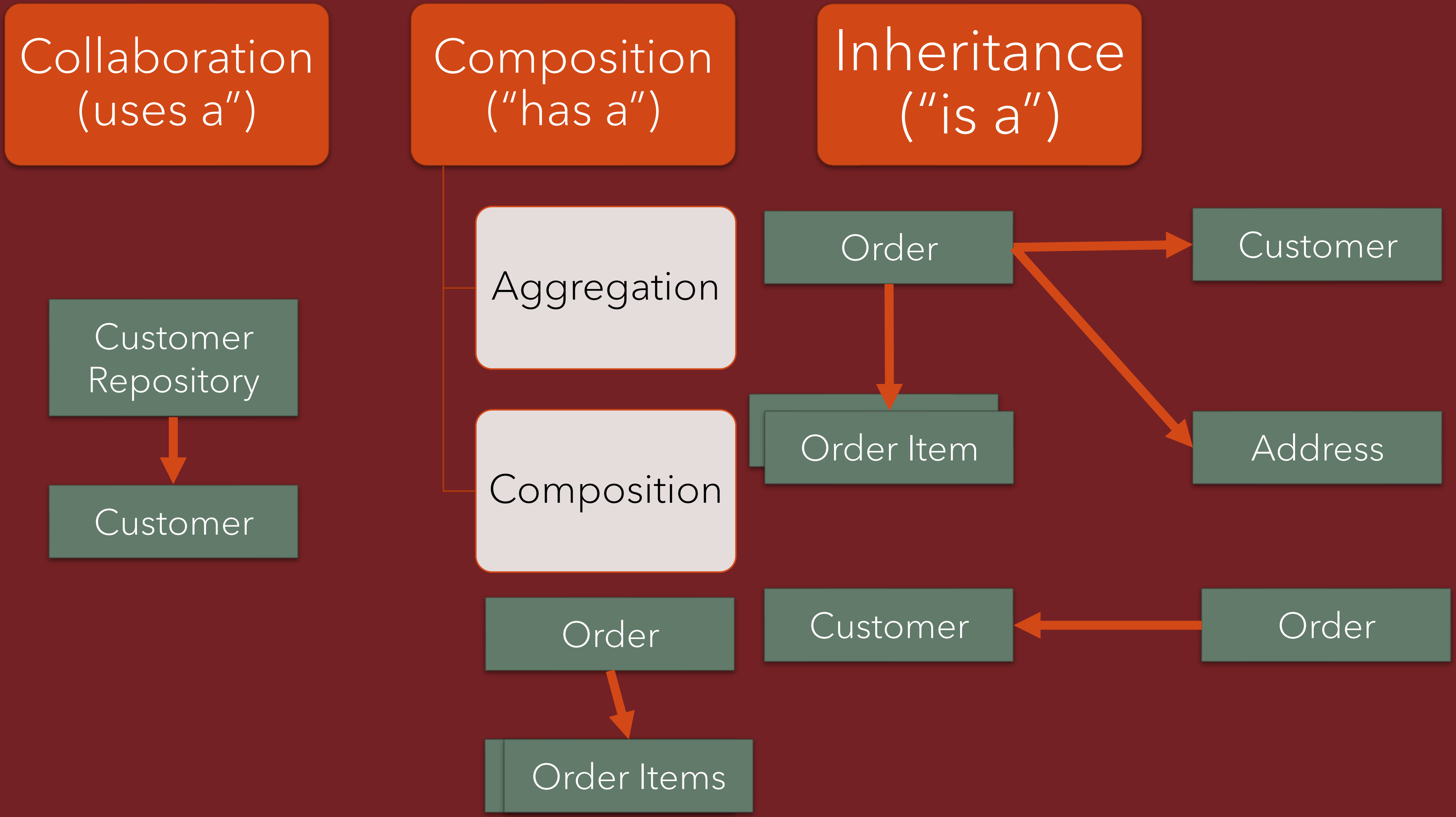
Address
- City
- State
- Country
- Postal Code
- Address Type
- Validate()

Establishing Relationships

Working Together



Types of Relationships



THANK YOU !