

Final Words and Next Steps



Deborah Kurata

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/



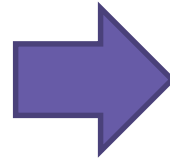
Object-Oriented Programming (OOP)

Identifying classes



- Represents business entities
- Defines properties (data)
- Defines methods (actions/behavior)

Separating responsibilities



- Minimizes coupling
- Maximizes cohesion
- Simplifies maintenance
- Improves testability

Establishing relationships



- Defines how objects work together to perform the operations of the application

Leveraging reuse



- Involves extracting commonality
- Building reusable classes / components
- Defining interfaces



Four Pillars of OOP

A diagram illustrating the 'Four Pillars of OOP' using a classical building metaphor. The building has a triangular pediment at the top containing the title 'Four Pillars of OOP'. Below the pediment are four vertical pillars, each with a label: 'Abstraction', 'Encapsulation', 'Inheritance', and 'Polymorphism'. The pillars are supported by a series of horizontal base layers at the bottom.

Abstraction

Encapsulation

Inheritance

Polymorphism



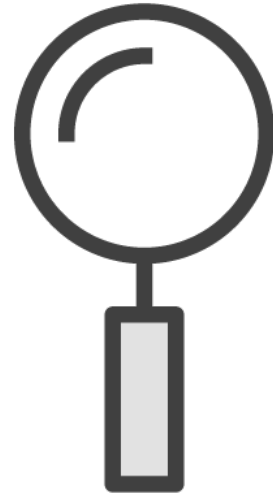
Abstraction



Simplifying reality

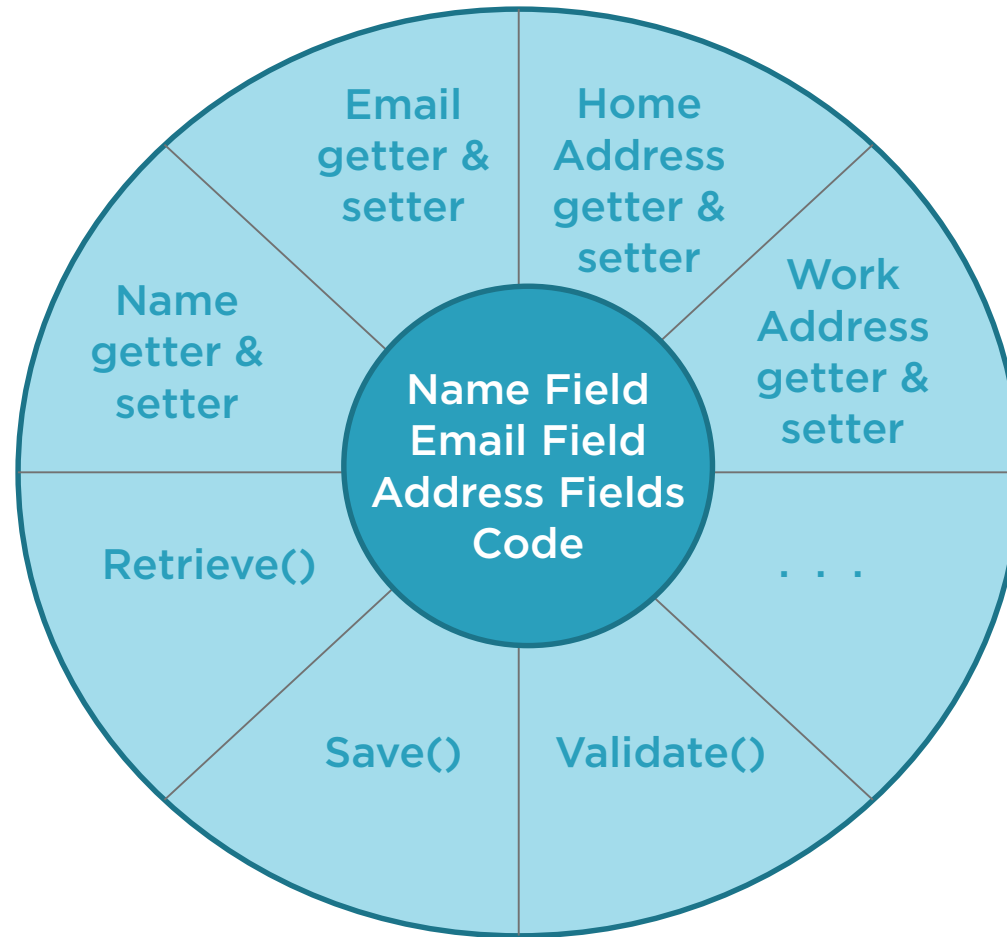


Ignoring extraneous
details



Focusing on what is
important for a
purpose

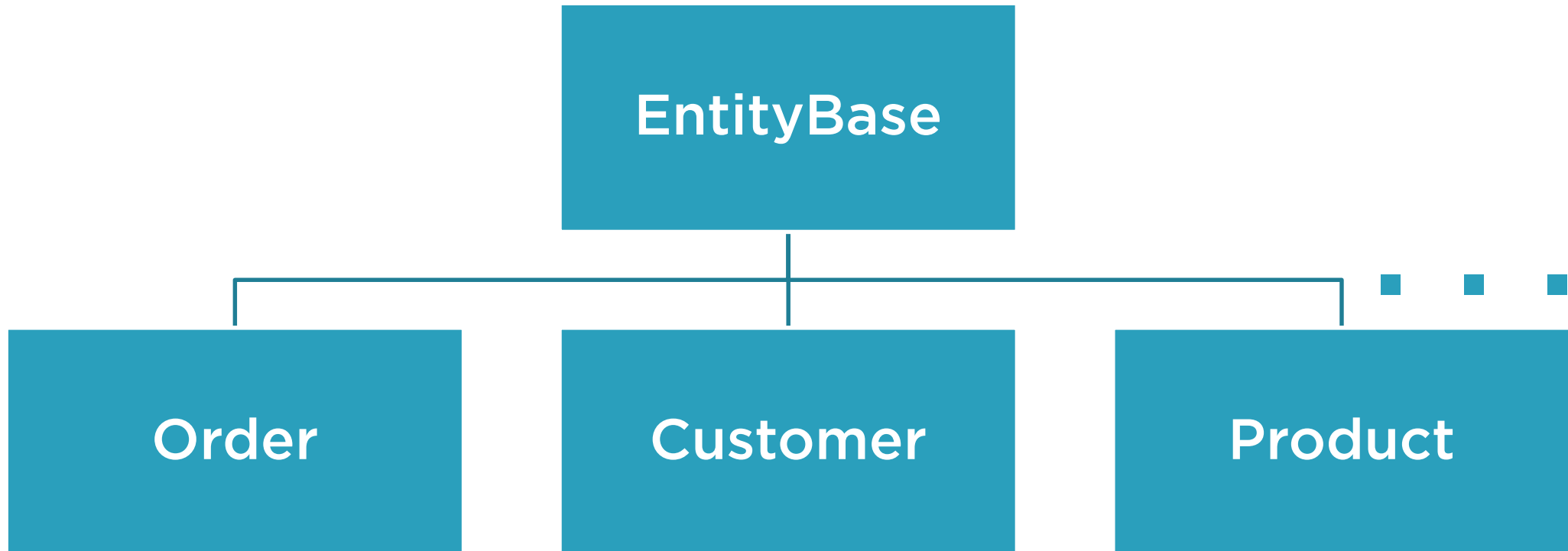
Encapsulation



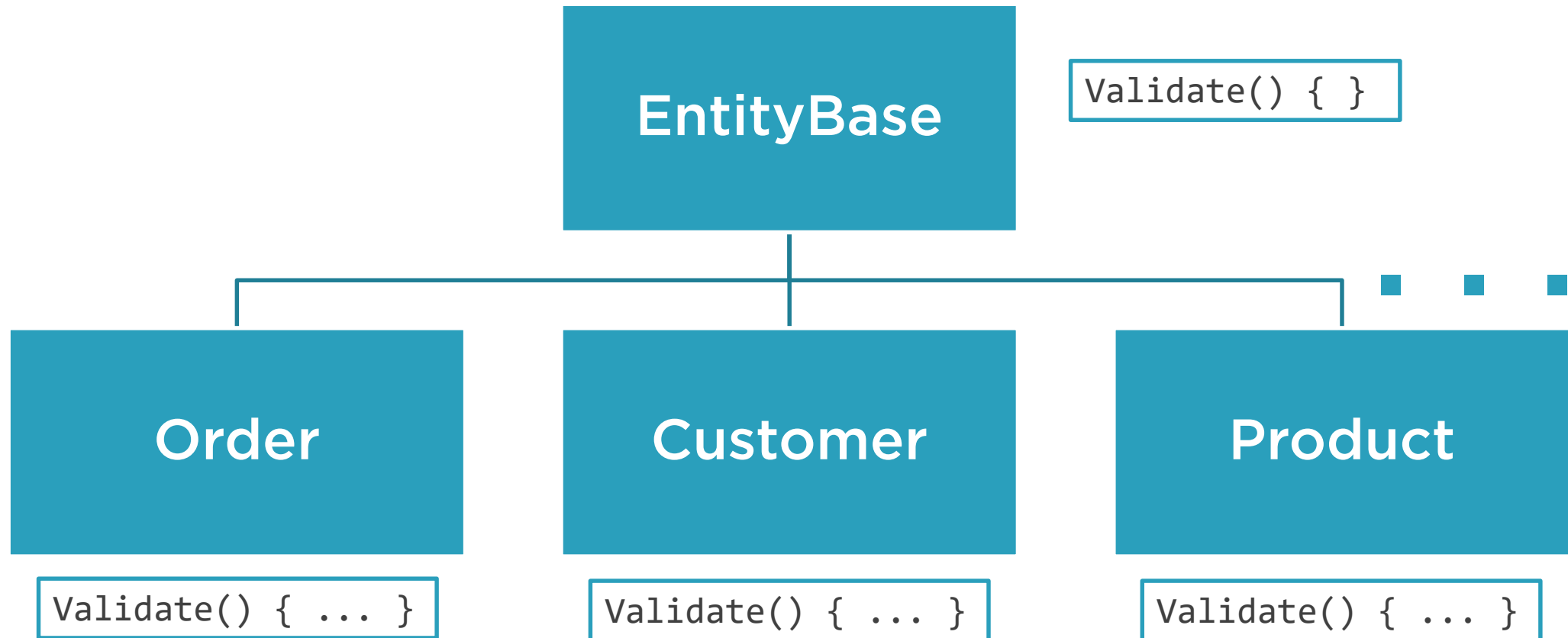
Customer Class



Inheritance



Inheritance-based Polymorphism

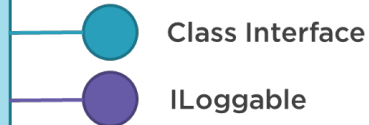


Interface-based Polymorphism

```
public void WriteToFile(List<ILoggable> itemsToLog)
{
    foreach (var item in itemsToLog)
    {
        Console.WriteLine(item.Log());
    }
}
```

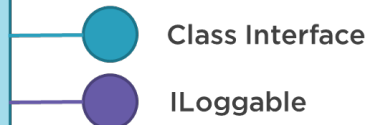
Customer Class

```
public class Customer : ILoggable
{
    public string FirstName { get; set; }
    ...
    public string Log() { ... }
}
```

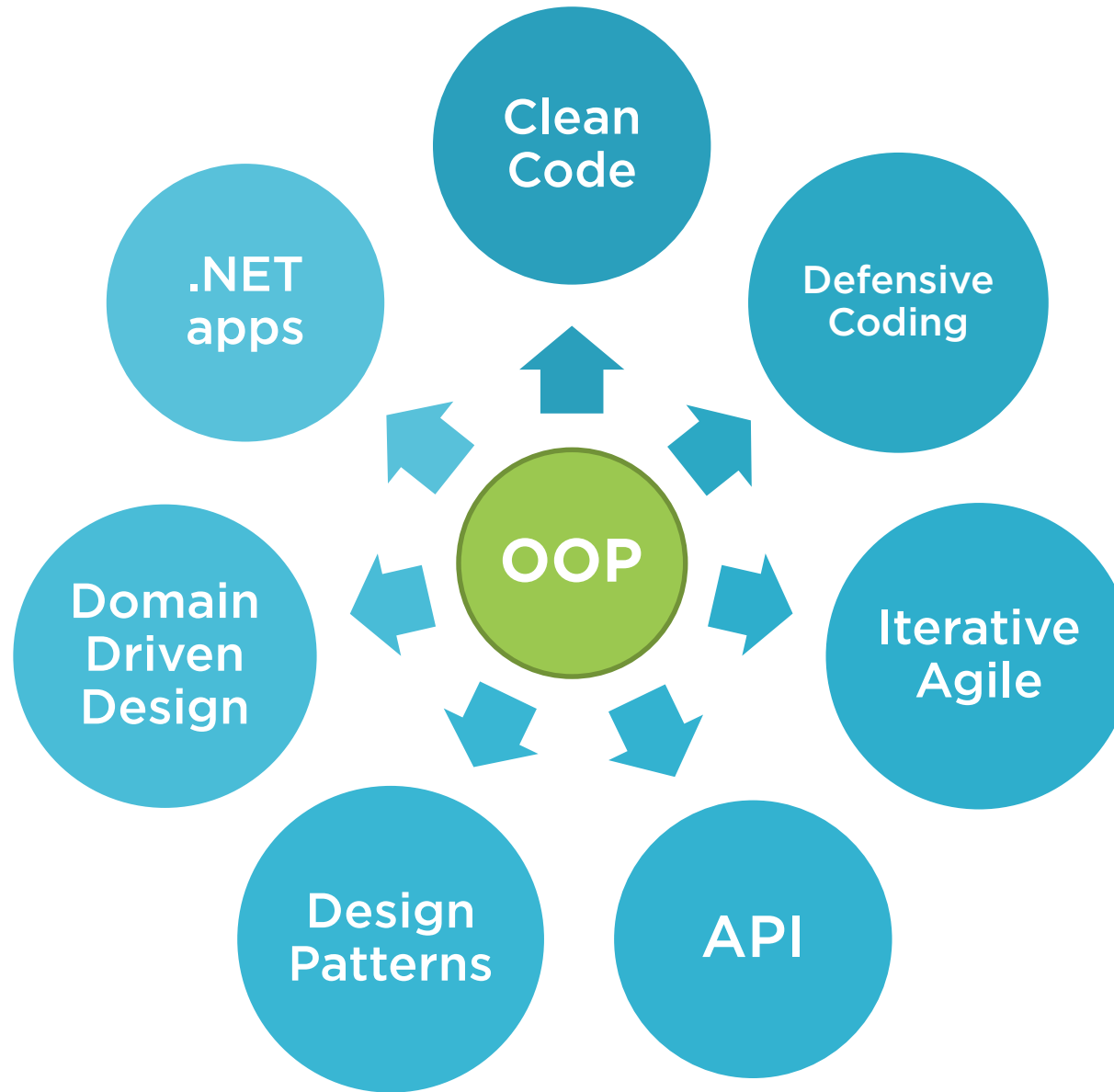


Product Class

```
public class Product : ILoggable
{
    public string ProductName { get; set; }
    ...
    public string Log() { ... }
}
```



OOP Is the Foundation



Next Steps



Additional Pluralsight Courses

- Defensive Coding in C#
- Clean Code: Writing Code for Humans
- C# Interfaces
- Design Patterns On-Ramp
- C# Best Practices: Improving on the Basics
- C# Best Practices: Collections and Generics



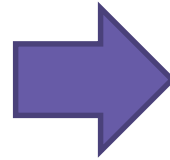
Object-Oriented Programming (OOP)

Identifying classes



- Represents business entities
- Defines properties (data)
- Defines methods (actions/behavior)

Separating responsibilities



- Minimizes coupling
- Maximizes cohesion
- Simplifies maintenance
- Improves testability

Establishing relationships



- Defines how objects work together to perform the operations of the application

Leveraging reuse



- Involves extracting commonality
- Building reusable classes / components
- Defining interfaces

