

Object-Oriented Programming Fundamentals in C#

INTRODUCTION



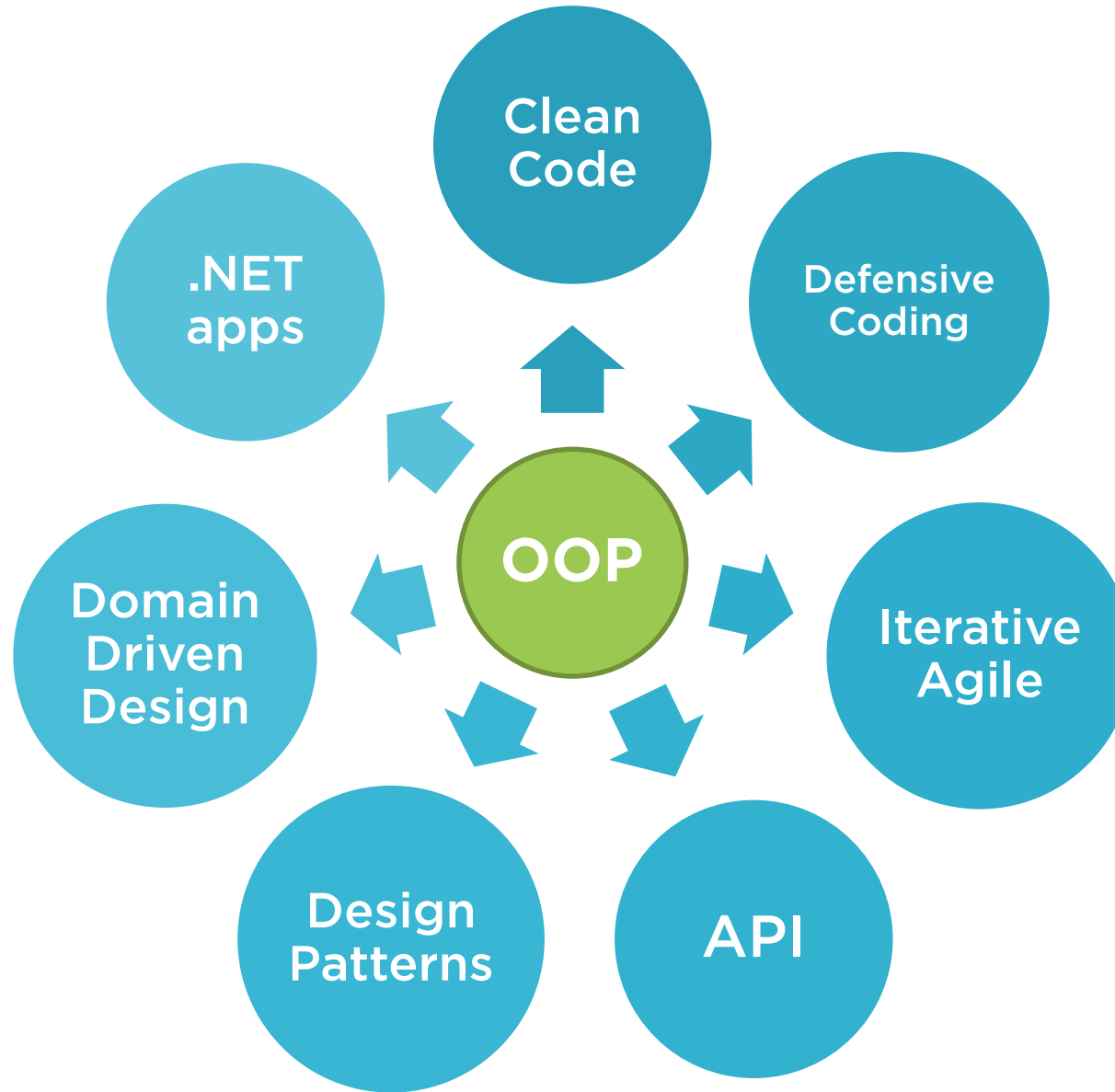
Deborah Kurata

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/



OOP Is the Foundation



Object

!=

Class

Properties

Methods

```
public class Customer
{
    public int CustomerId { get; set; }
    public string EmailAddress { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public bool Validate(){ ... }
}
```



Object

!=

Class

Members

```
public class Customer
{
    public int CustomerId { get; set; }
    public string EmailAddress { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public bool Validate(){ ... }
}
```



Object

!=

Class

```
var customer = new Customer();
```

Object
variable

```
public class Customer
{
    public int CustomerId { get; set; }
    public string EmailAddress { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public bool Validate(){ ... }
}
```



Objects



Class



We need to
define the
**business
objects**

**Business
Object**

=

Class

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

    public string EmailAddress { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

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



Entity



Customer Management System

Entity

customer

Class

Customer

- Last Name
- First Name
- Go On An Adventure

Objects

Bilbo Baggins

Frodo Baggins



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 one another

Identifying
classes

Separating
responsibilities

Establishing
relationships

Leveraging
reuse



Prerequisites

Required

- Basic knowledge of C#
 - Syntax
 - Variables
 - Conditionals
 - Control flow

Suggested

- Visual Studio

Not Required

- OOP experience



Checklist



Review module concepts

Code along assistance

Revisit as you implement your applications



Coding Along

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Test Explorer (Left):** Shows a tree view of tests under 'ACM (17 tests)'. All tests are marked as passed with green checkmarks. The 'Summary' section at the bottom indicates 'Last Test Run Passed' with a total run time of 0:00:00.8 and '17 Tests Passed'.
- Code Editor (Center):** Displays the 'Customer.cs' file. The code defines a `Customer` class that inherits from `EntityBase` and implements `ILoggable`. It includes a parameterless constructor, a constructor taking a `customerId`, and several public properties: `AddressList`, `CustomerId`, `CustomerType`, `EmailAddress`, `FirstName`, and `FullName`.
- Solution Explorer (Right):** Shows the project structure for 'Solution 'ACM' (4 projects)', including 'Tests', 'ACM.BL', and 'Acme.Common'.

```
7 public class Customer : EntityBase, ILoggable
8 {
9     public Customer() : this(0)
10    {
11    }
12
13    public Customer(int customerId)
14    {
15        CustomerId = customerId;
16        AddressList = new List<Address>();
17    }
18
19    public List<Address> AddressList { get; set; }
20
21    public int CustomerId { get; private set; }
22
23    public int CustomerType { get; set; }
24
25    public string EmailAddress { get; set; }
26
27    public string FirstName { get; set; }
28
29    public string FullName
30    {
31        get
32    }
```



Course Outline

Identifying classes

- Identifying classes from requirements
- Building entity classes

Separating responsibilities

- Separating responsibilities

Establishing relationships

- Establishing relationships

Leveraging reuse

- Leveraging reuse
- Building reusable components
- Understanding interfaces

