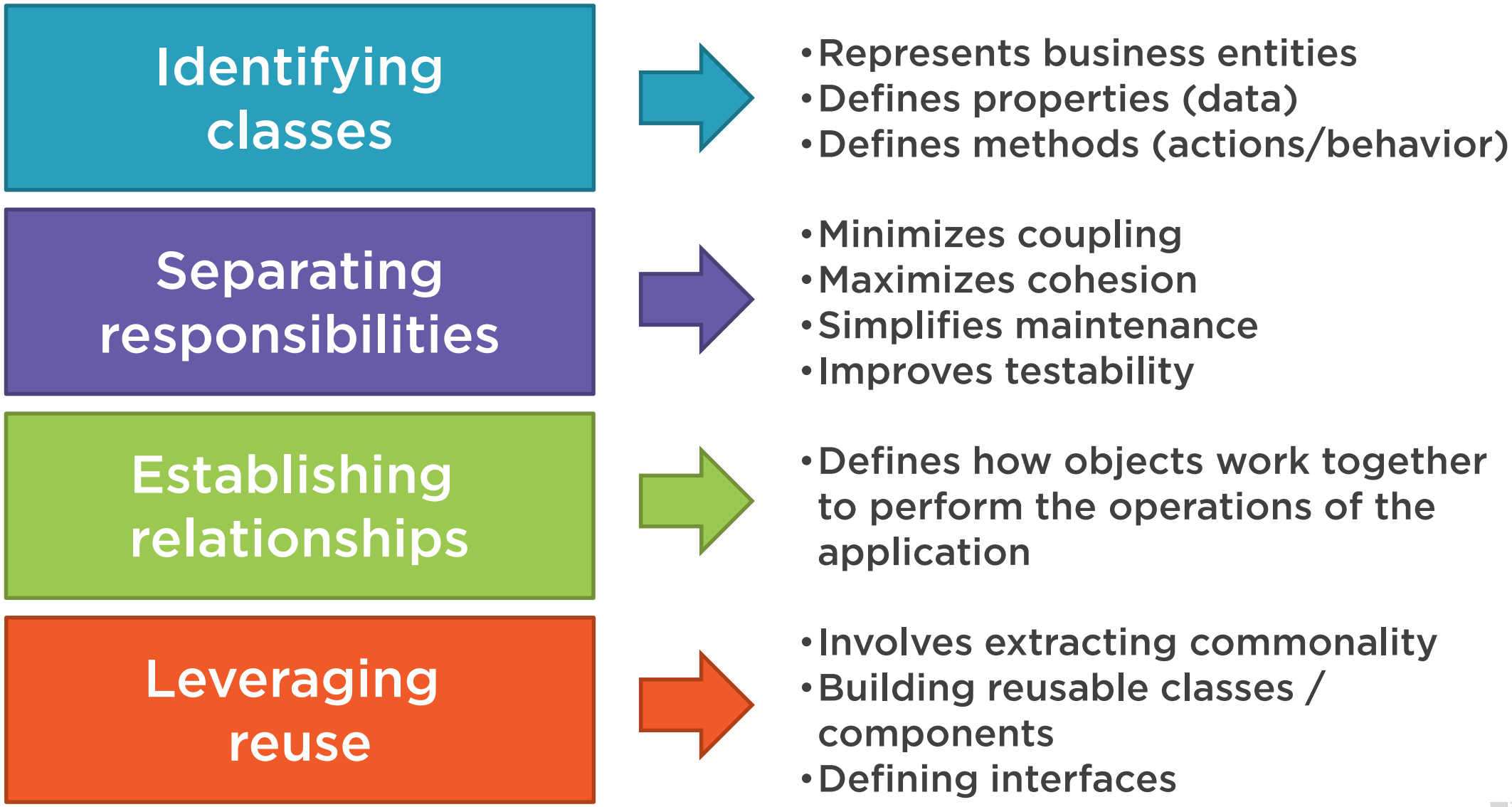
Leveraging Reuse through Inheritance

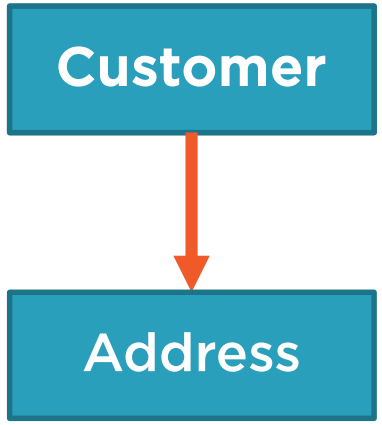
Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Object-Oriented Programming (OOP)

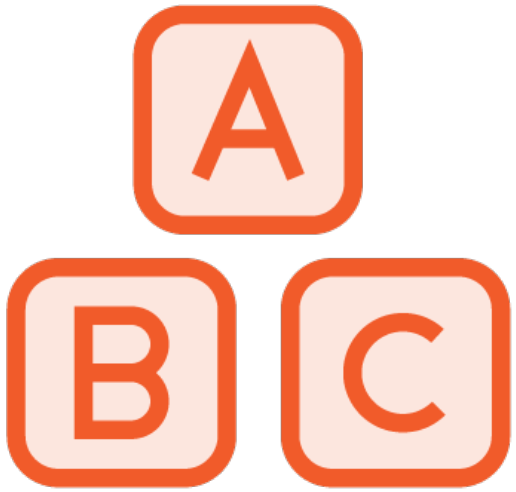


Techniques for Leveraging Reuse

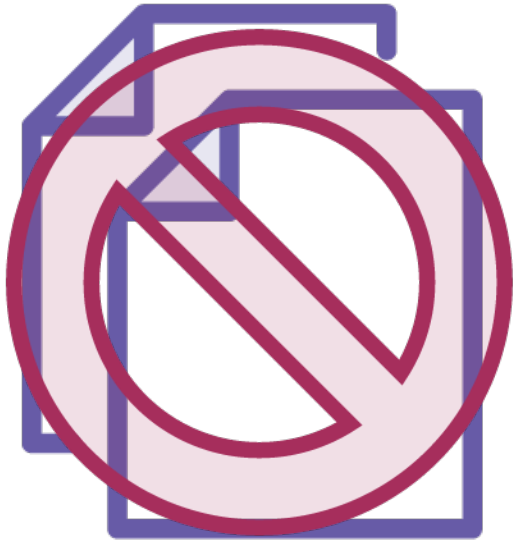


Collaboration / Composition

Inheritance

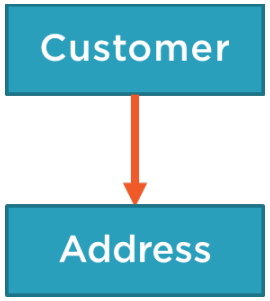


Components



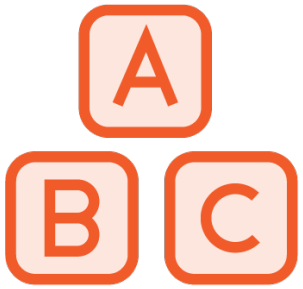
Copy/Paste

Techniques for Leveraging Reuse

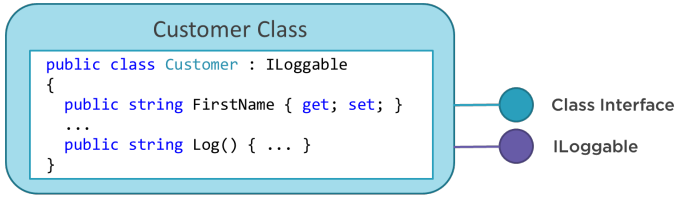


Collaboration / Composition

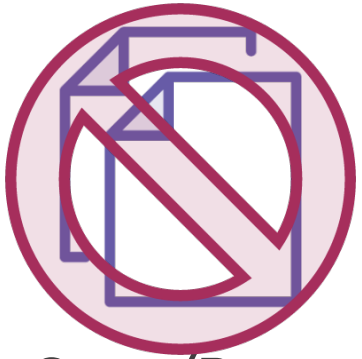
Inheritance



Components



Interfaces



Copy/Paste

Module Outline

The .NET Object class

Overriding base class functionality

Polymorphism

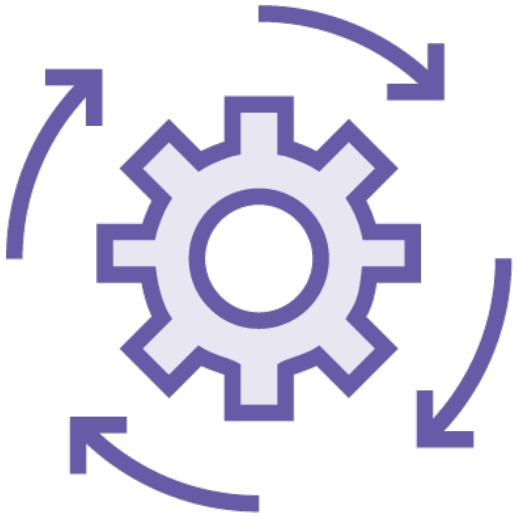
Building a base class

Secrets of Reuse

Build it once

Test it

Reuse it!



Update it in one place

Advantages of Reuse

Reduces amount of code

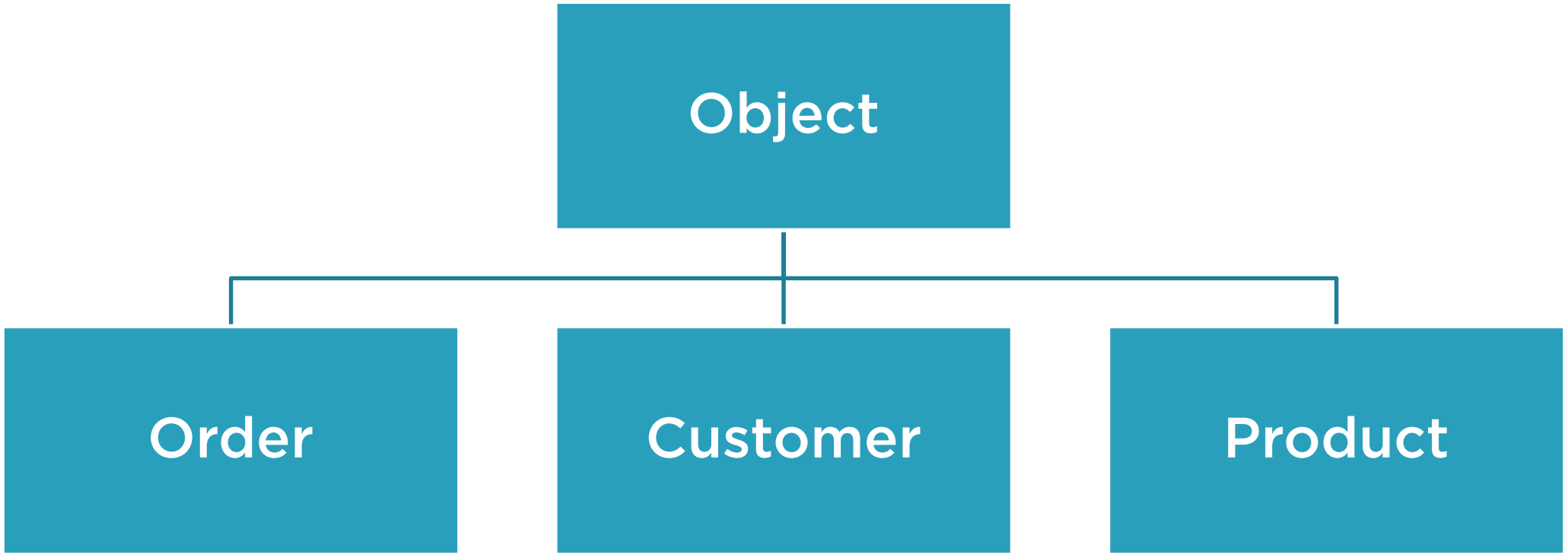
Reduces development time

Reduces costs

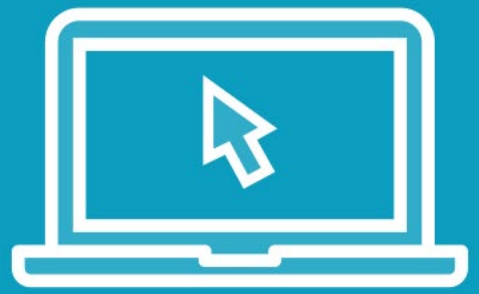
Reduces bugs

Inheritance

.NET Object Class

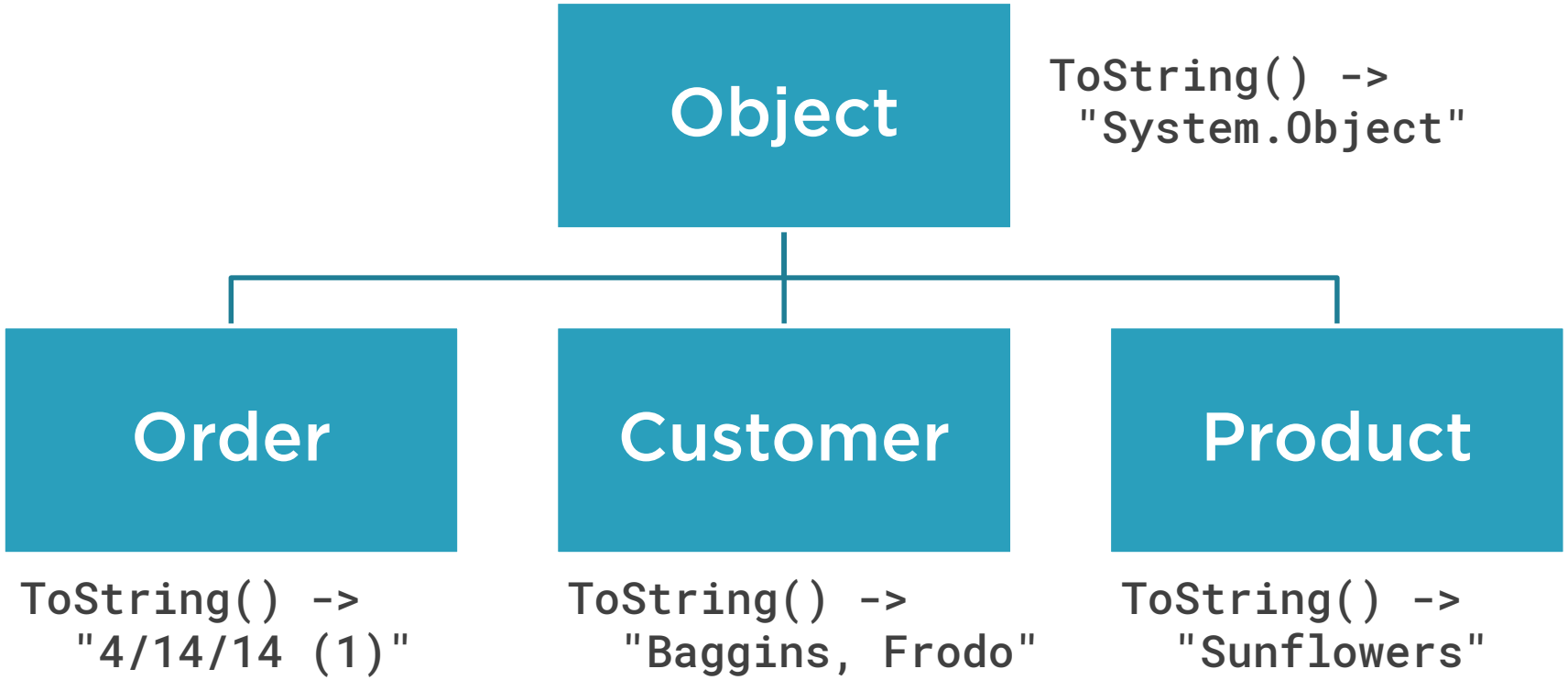


Demo



Overriding base class functionality

Polymorphism



Saving

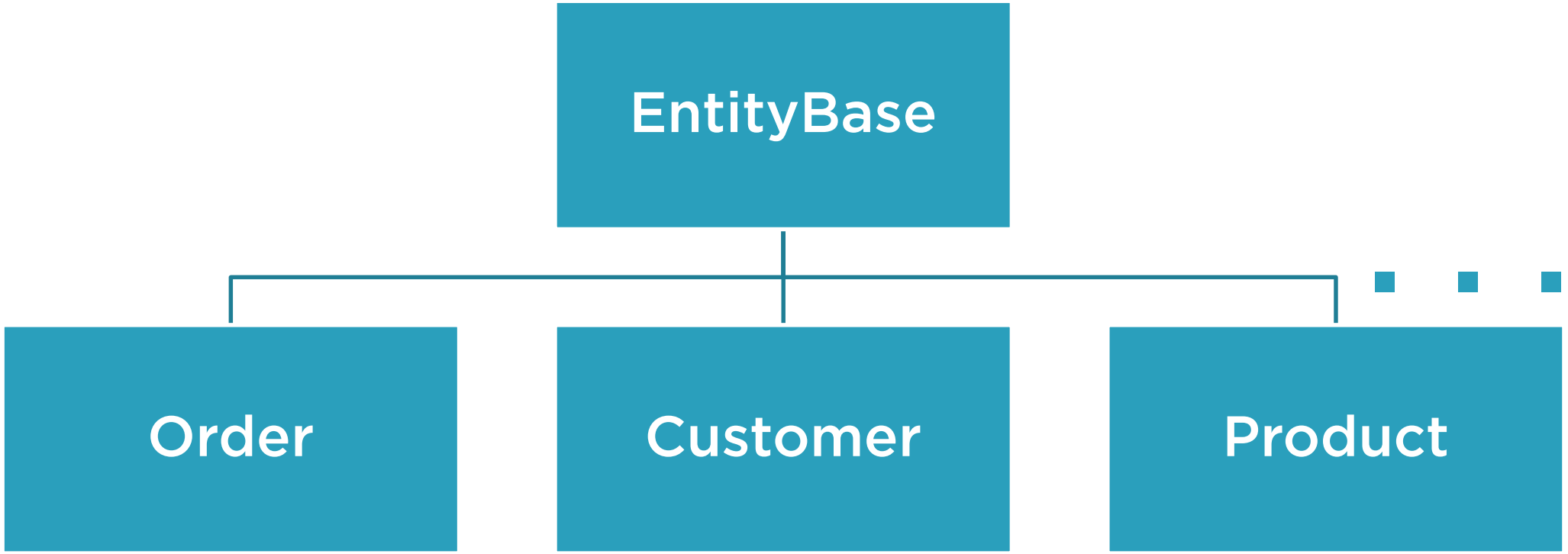
```
public bool IsNew { get; private set; }
```

```
public bool HasChanges { get; set; }
```

```
public bool IsValid => Validate();
```

```
public EntityStateOption EntityState { get; set; }
```

Base Class



Building a Base Class

Abstract Class

Incomplete, with at least one property or method not implemented

Cannot be instantiated

Intended for use as a base class

```
public abstract class EntityBase
{

}
```

Concrete Class

Normal class

Can be instantiated

Can be used as a base class

```
public class EntityBase
{

}
```

Sealed Class

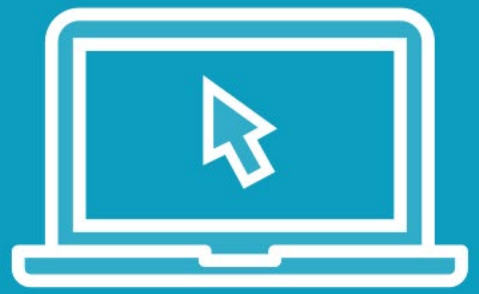
Cannot be extended through inheritance

Sealed using the `sealed` keyword

```
public sealed class Customer
{

}
```

Demo



Building a base class

```
public bool IsNew { get; private set; }
```

```
public bool HasChanges { get; set; }
```

```
public bool IsValid => Validate();
```

```
public EntityStateOption EntityState { get; set; }
```

Sealed Members

By default, class members are sealed and cannot be overridden

Expose members using

- Abstract
- Virtual

Preparing Overridable Base Class Members

Abstract

Method signature as place holder with no implementation

Only use in abstract classes

Must be overridden by derived class

```
public abstract bool Validate();
```

Virtual

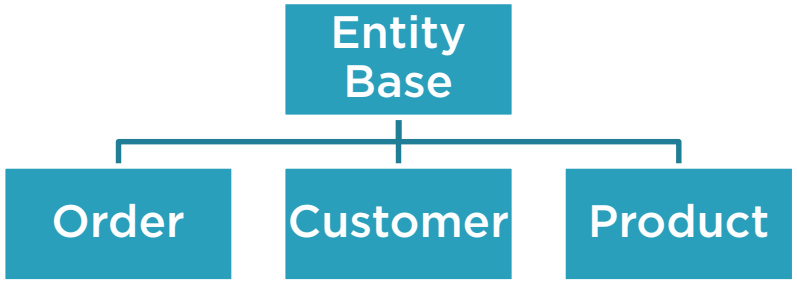
Method with default implementation

Use in abstract or concrete classes

Optionally overridden by derived class

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

Inheritance



Define a base class with common functionality

```
public class EntityBase
{
    public bool HasChanges { get; set; }

}
```

Inherit from that class to reuse its functionality

```
public class Product : EntityBase
{
}
```

Abstract Class

An incomplete class with one or more members that are not implemented

An abstract class cannot be instantiated

Intended for use as a base class

```
public abstract class EntityBase
{

}
```

Sealed Class

A concrete class that cannot be extended through inheritance

Use it to prevent overriding the class functionality

```
public sealed class Customer
{

}
```

Abstract vs. Virtual Methods

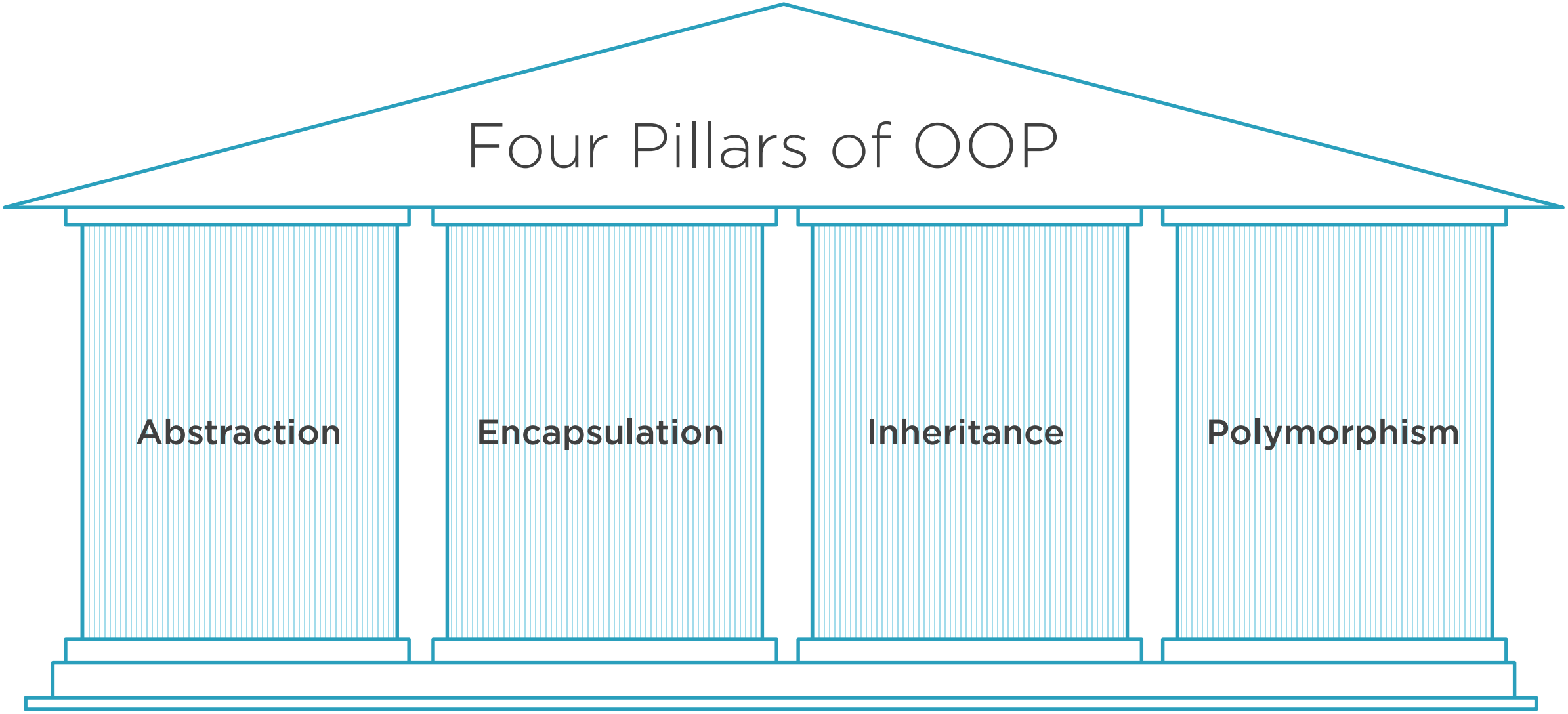
Abstract method is a placeholder, no implementation, that must be overridden

```
public abstract bool Validate();
```

Virtual method is a method with a default implementation that can be overridden

```
public virtual bool Validate()
{
    // Default implementation
}
```

Four Pillars of OOP



Object-Oriented Programming (OOP)