Understanding Interfaces

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

"In computing, an **interface** is a shared boundary across which two or more separate components of a computer system exchange information.

The exchange can be between software, computer hardware, peripheral devices, humans and combinations of these."

- Wikipedia 1/9/19

User Interface

Web API

Class Interface

Module Outline

Class interface

Defining an interface

Implementing an interface

Interface-based polymorphism

Class Interface

```
public class Customer : EntityBase
{
    public List<Address> AddressList { get; set; }

    public int CustomerType { get; set; }

    public static int InstanceCount { get; set; }

    public string LastName...

    public string FirstName { get; set; }

    public string EmailAddress { get; set; }

    public int CustomerId { get; private set; }

    public string FullName...

    public override bool Validate()...

    public override string ToString() => FullName;
}
```

Class Interface

Customer Class

```
public string FirstName { get; set; }
public string LastName { get; set; }
...
public bool Validate() { ... }
```

Product Class

```
public string ProductName { get; set; }
public decimal CurrentPrice { get; set; }
...
public bool Validate() { ... }
```

Interface

Customer Class

```
public string FirstName { get; set; }
public string LastName { get; set; }
...
public bool Validate() { ... }
```

```
public string Log() { ... }
```

Product Class

```
public string ProductName { get; set; }
public decimal CurrentPrice { get; set; }
...
public bool Validate() { ... }
```

```
public string Log() { ... }
```

```
public interface ILoggable
{
    string Log();
}
```

Interface

Customer Class

```
public string FirstName { get; set; }
public string LastName { get; set; }
...
public bool Validate() { ... }
```

```
public string Log() { ... }
```

Product Class

```
public string ProductName { get; set; }
public decimal CurrentPrice { get; set; }
...
public bool Validate() { ... }
```

```
public string Log() { ... }
```

ACM Application

- ACM.WPF
- ACM.Web
- ACM.BL
- ACM.DL
- Acme.Common

Interfaces Define "Roles"

Default person Interface

IParent

IPartner

IEmployee

Customer Class

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

Class Interface

ILoggable

Interfaces are a "Contract"

```
public interface ILoggable
{
    string Log();
}
```

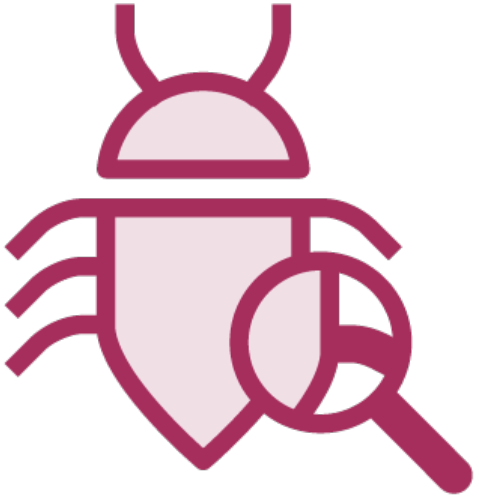
Customer Class

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

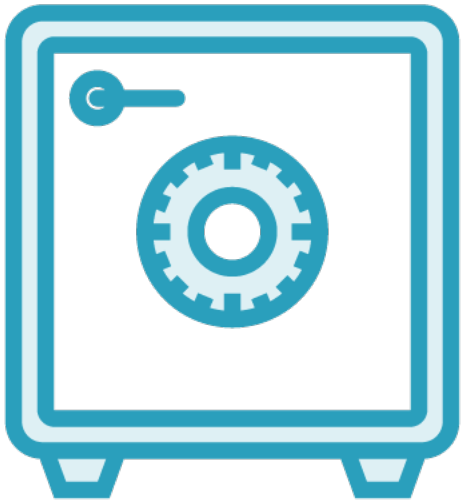
Class Interface

ILoggable

Logging



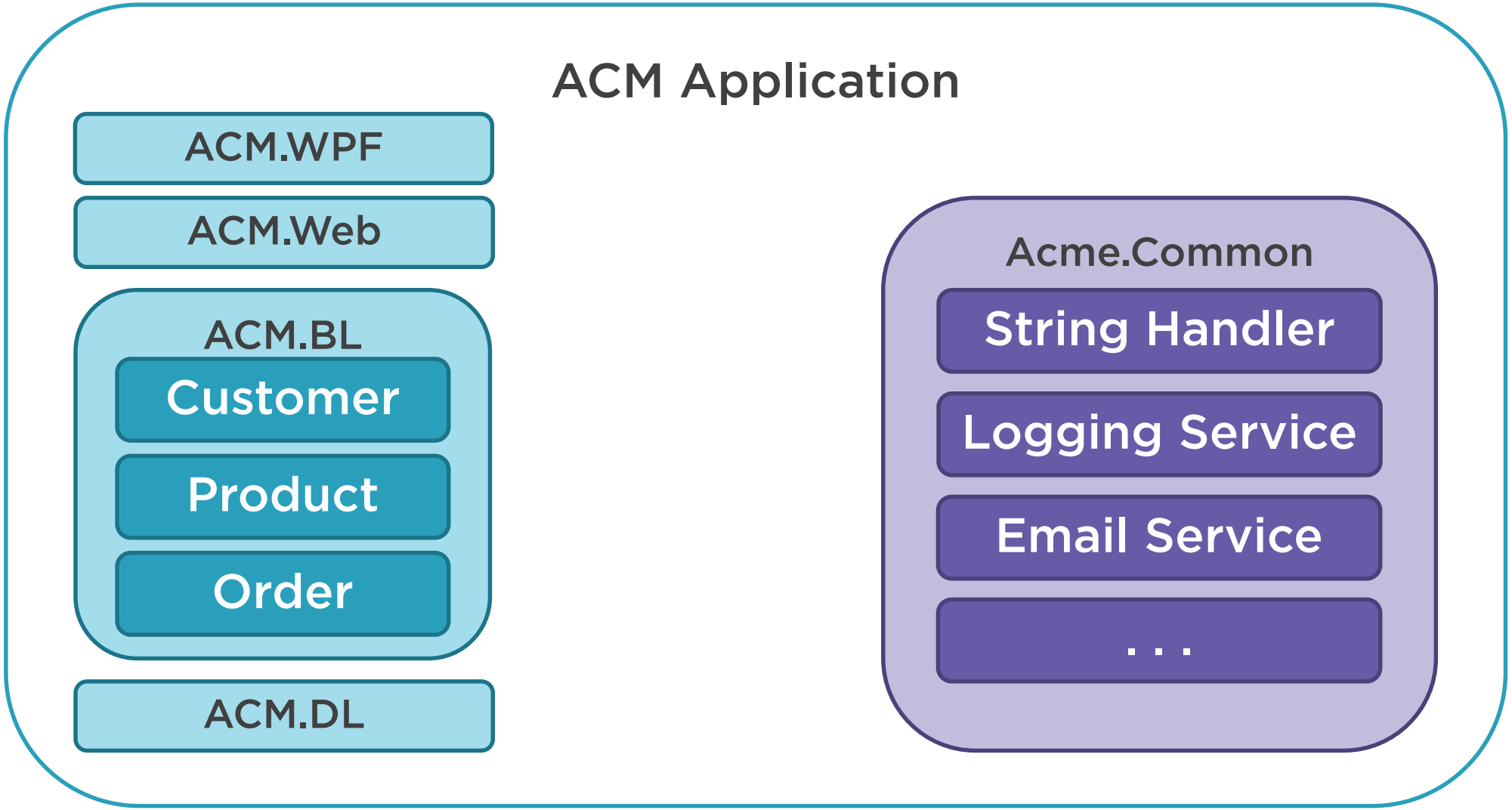
Resolving bugs



Security

Data analysis

Reusable Library Component



Interface

```
public interface ILoggable
{
    string Log();
}
```

```
public interface IEmailable
{
    bool CopyToSender { get; set; }

    string Send(string sendTo, string[] ccTo);
}
```

Defining an Interface

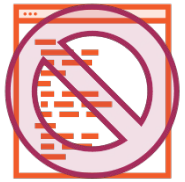
Add a new interface item to a project

Prefix the interface name with "I"

Specify the `public` access modifier

Add property, method, event, or indexer signatures

No need for an access modifier on the members



No implementation of the members

Implementing an Interface

Add the interface to the class signature

```
public class Customer : ILoggable, IEmailable
```

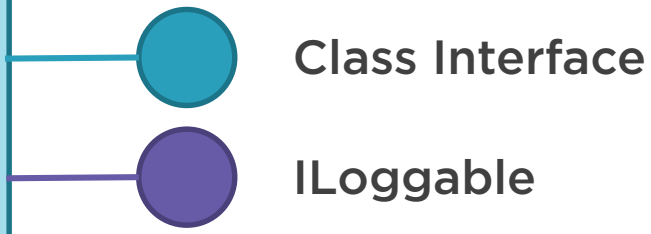
Implement each member of the interface

```
public string Log() { ... }
```

Implementing an Interface

Customer Class

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



Implementing an Interface

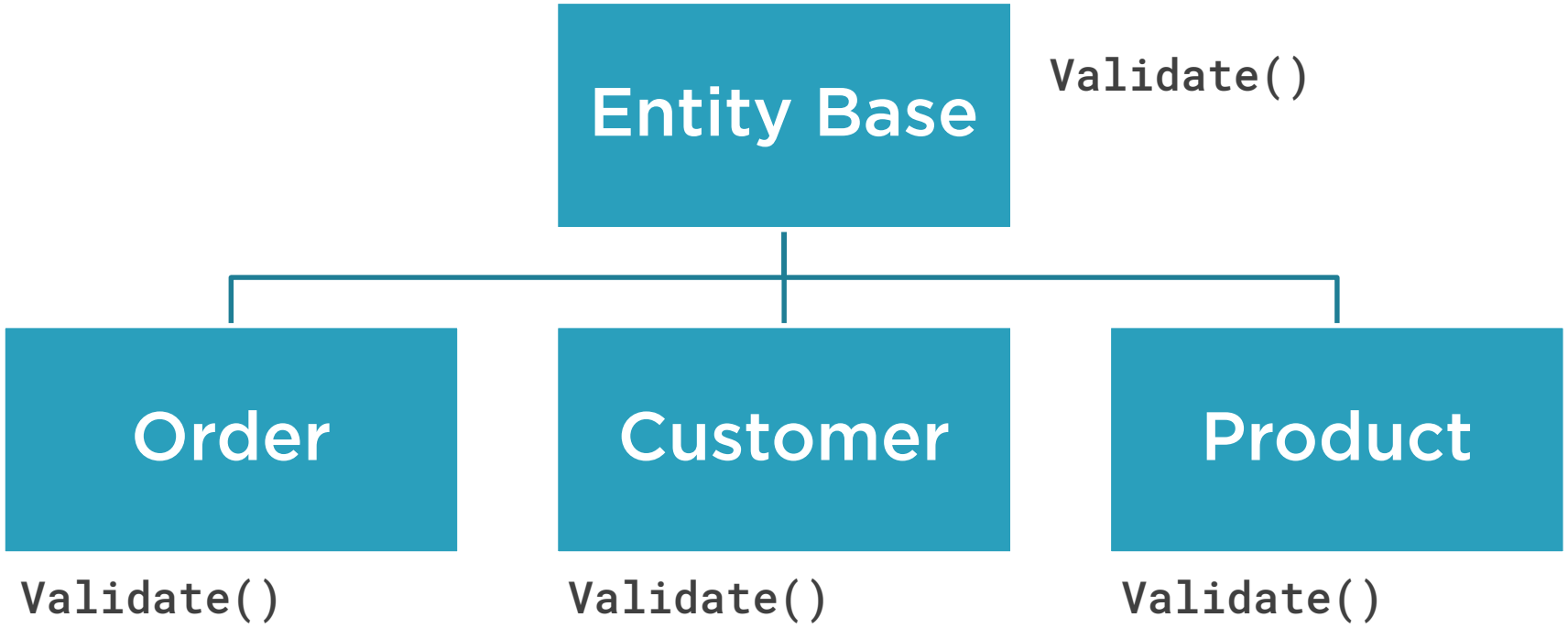
Customer Class

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

  public string Send(string sendTo) { ... }

}
```

Inheritance-based Polymorphism



Interface-based Polymorphism

Customer Class

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

Class Interface

ILoggable

Product Class

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

Class Interface

ILoggable

Interface-based Polymorphism

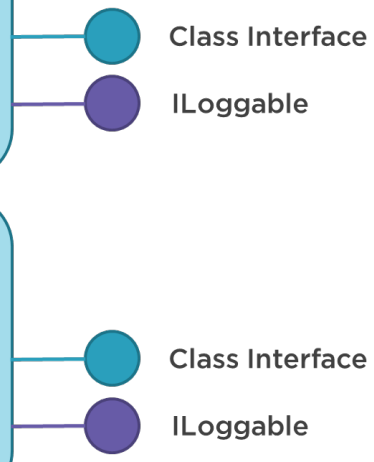
```
public void WriteToFile(List<ILoggable> itemsToLog)
{
    foreach (var item in itemsToLog)
    {
        Console.WriteLine(item.Log());
    }
}
```

Customer Class

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

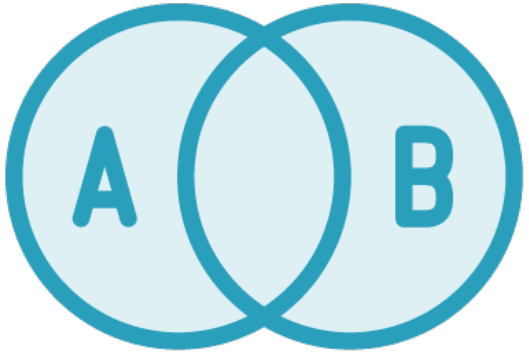
Product Class

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

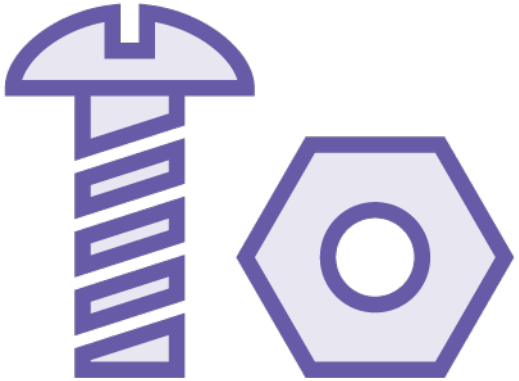


Benefits of Interfaces and Polymorphism

Strong typing



Defines commonality among unrelated classes



Aids in building generalized utility methods with class-unique functionality

Interfaces

Define a role that an object can play

Define a contract

An interface is comprised of a list of properties, methods, events and iterators

- Denoting the data and operations an object can perform

Class Interface

Every class has a basic interface

Defined by the public properties and methods of the class

The application uses this interface to work with the class in its primary role

Interface

Define any number of additional interfaces

Define an interface using the `interface` keyword

```
public interface ILoggable
{
    string Log();
}
```

Specify the signatures of the interface members with no code

Implementing an Interface

Any class can implement an interface

Implement an interface by adding it to the class signature after a colon

```
public class Customer : ILoggable, IEmailable
```

The class must then implement every property and method defined in that interface

Interface-based Polymorphism

Many shapes

One method name specified in an interface, multiple implementations of that method

One implementation in each class that implements the interface

Allows us to work with otherwise unrelated classes in a generalized, reusable way