Building Entity Classes - Methods

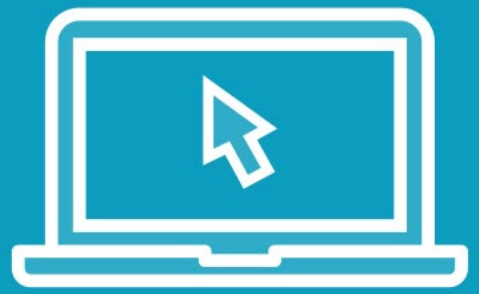
Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Identified Classes

Customer	Product	Order	Order Item
•Name •Email address •Home address •Work address •Validate() •Retrieve() •Save()	•Product name •Description •Current price •Validate() •Retrieve() •Save()	•Customer •Order date •Shipping address •Order items •Validate() •Retrieve() •Save()	•Product •Quantity •Purchase price •Validate() •Retrieve() •Save()

Demo

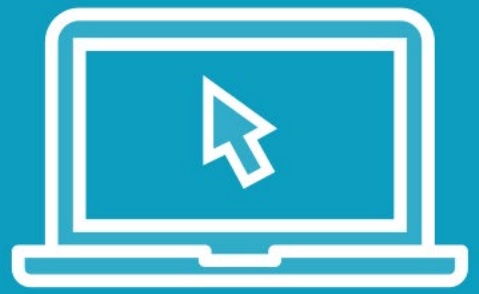


Creating class methods

Customer

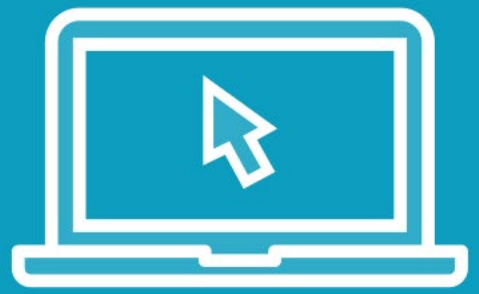
- Name
- Email address
- Home address
- Work address
- Validate()
- Retrieve()
- Save()

Demo



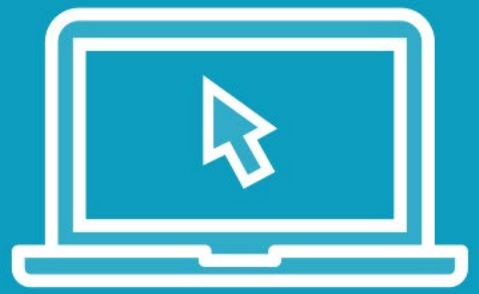
Testing methods

Demo



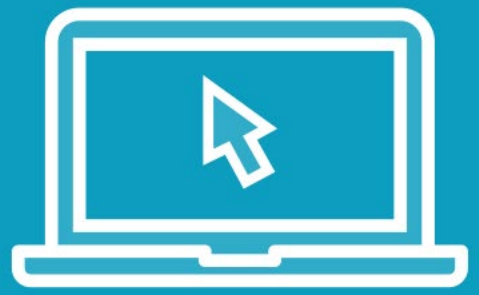
Method terms

Demo



Constructors

Demo



Building the remaining classes

Product	Order	Order Item
•Product name •Description •Current price •Validate() •Retrieve() •Save()	•Customer •Order date •Shipping address •Order items •Validate() •Retrieve() •Save()	•Product •Quantity •Purchase price •Validate() •Retrieve() •Save()

Classes

Customer	Product	Order	Order Item
•Name •Email address •Home address •Work address •Validate() •Retrieve() •Save()	•Product name •Description •Current price •Validate() •Retrieve() •Save()	•Customer •Order date •Shipping address •Order items •Validate() •Retrieve() •Save()	•Product •Quantity •Purchase price •Validate() •Retrieve() •Save()

Creating Methods

Set the appropriate access modifier

Specify the desired return type or void

Give the method a good name

```
public bool Validate()
{
  var isValid = true;

  if (string.IsNullOrWhiteSpace(LastName)) isValid = false;
  if (string.IsNullOrWhiteSpace(EmailAddress)) isValid = false;

  return isValid;
 }
```

Unit Testing Methods

Define tests for valid and invalid scenarios

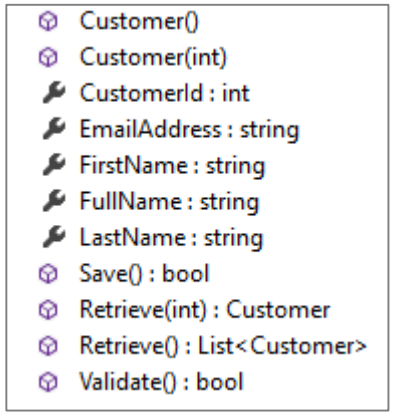
Organize the test

- Arrange: Set up the test
- Act: Call the method being tested
- Assert: Determine the result

Method Terminology

```
public Customer Retrieve(int customerId)
```

Signature



Contract

```
public Customer Retrieve(int customerId)

public List<Customer> Retrieve()
```

Overloading

```
public Customer() { }

public Customer(int customerId) { }
```

Constructor

Object-Oriented Programming (OOP)

Identifying classes

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

Separating responsibilities

Establishing relationships

Leveraging reuse