

Identifying Classes from Requirements



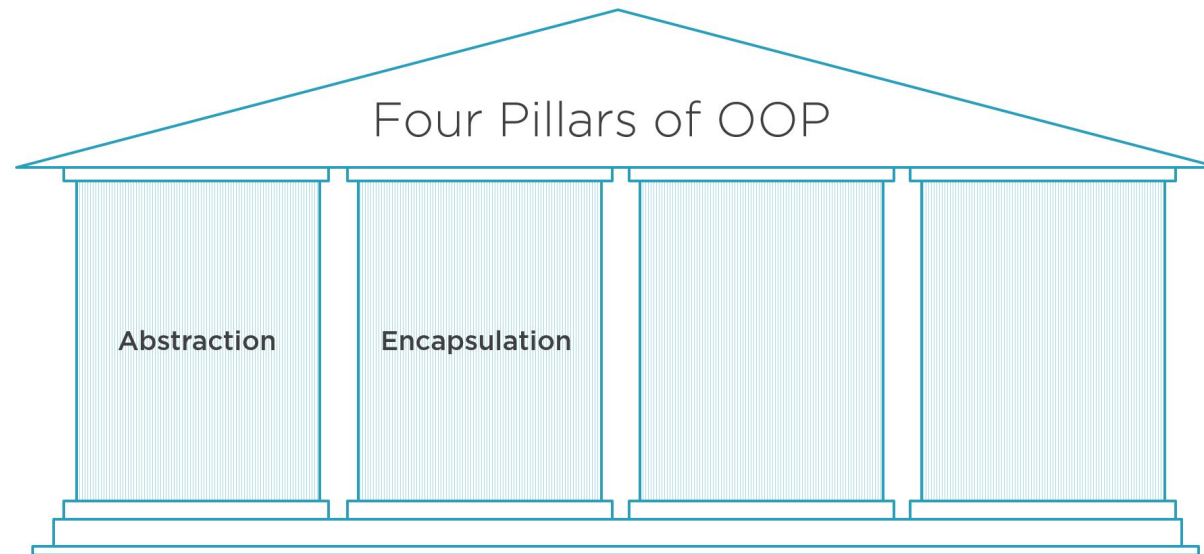
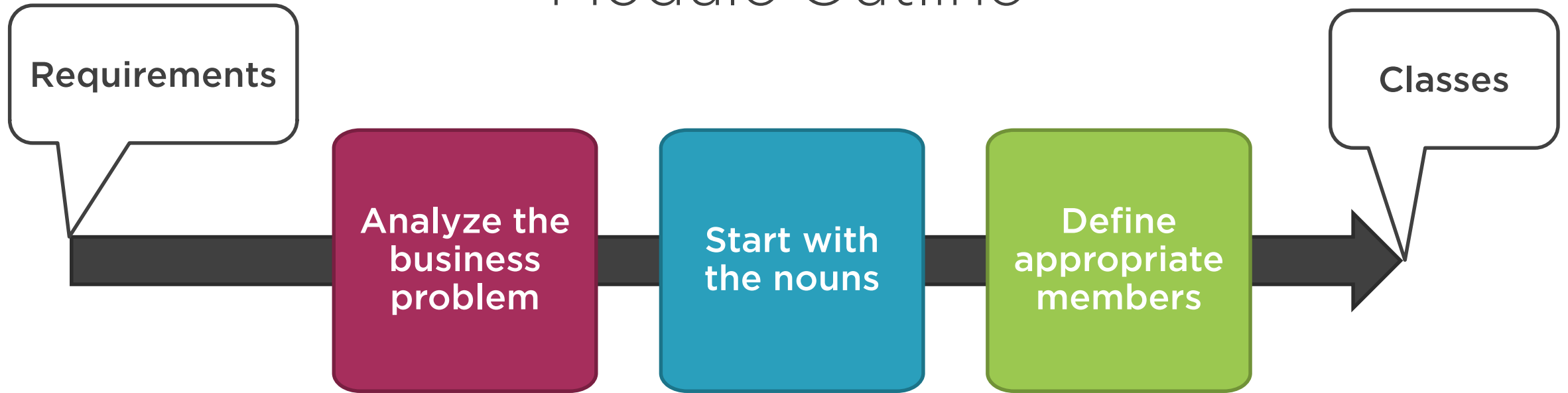
Deborah Kurata

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/



Module Outline



Business Requirements



Object-Oriented Programming (OOP)

**Identifying
classes**



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

**Separating
responsibilities**

**Establishing
relationships**

**Leveraging
reuse**





Acme Customer Management System



Manage business,
residential, government,
and educator types of
customers



Manage our products



Accept orders from
customers online or
through our call center



Start with the nouns.



Acme Customer Management System



Manage business,
residential, government,
and educator types of
customers

Customer



Manage our products

Product



Accept orders from
customers online or
through our call center

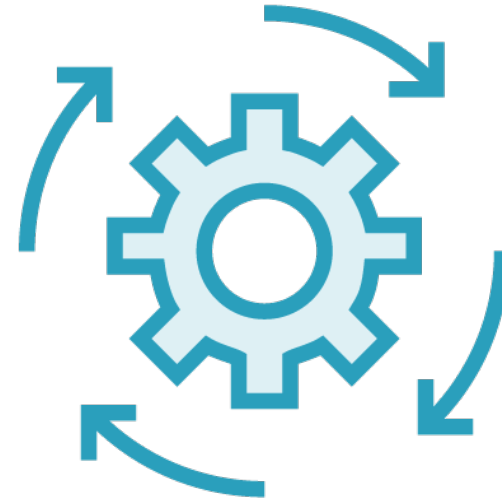
Order



Defining Appropriate Members



Properties: data



Methods: operations

Acme Customer Management System



- Customer's name (Last name, first name)
- Email address
- Home and work addresses



- Product name
- Description
- Current price



- Customer
- Order date
- Shipping address
- Products and quantities ordered



Define Appropriate Members

Customer

- Name
- Email address
- Home address
- Work address

Product

- Product name
- Description
- Current price

Order

- Customer
- Order date
- Shipping address
- Product
- Quantity



Define Appropriate Members

Customer

- Name
- Email address
- Home address
- Work address

Product

- Product name
- Description
- Current price

Order

- Customer
- Order date
- Shipping address
- Order items

Order Item

- Product
- Quantity



Define Appropriate Members

Customer

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

Product

- Product name
- Description
- Current price
- Validate()
- Retrieve()
- Save()

Order

- Customer
- Order date
- Shipping address
- Order items
- Validate()
- Retrieve()
- Save()

Order Item

- Product
- Quantity
- Validate()
- Retrieve()
- Save()





Consider the effect of data changes
over time



Define Appropriate Members

Customer

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

Product

- Product name
- Description
- Current price
- Validate()
- Retrieve()
- Save()

Order

- Customer
- Order date
- Shipping address
- Order items
- Validate()
- Retrieve()
- Save()

Order Item

- Product
- Quantity
- Validate()
- Retrieve()
- Save()



Define Appropriate Members

Customer

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

Product

- Product name
- Description
- Current price
- Validate()
- Retrieve()
- Save()

Order

- Customer
- Order date
- Shipping address
- Order items
- Validate()
- Retrieve()
- Save()

Order Item

- Product
- Quantity
- Purchase price
- Validate()
- Retrieve()
- Save()



Abstraction

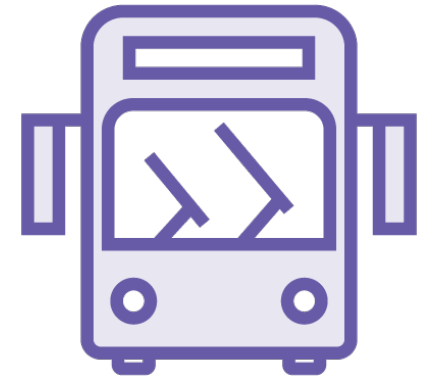
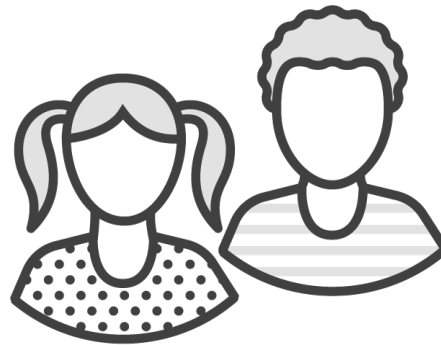
**Manage business, residential,
government, and educator
types of customers**



Customer



Joe Smith
Joe@aol.com
123 Main St.



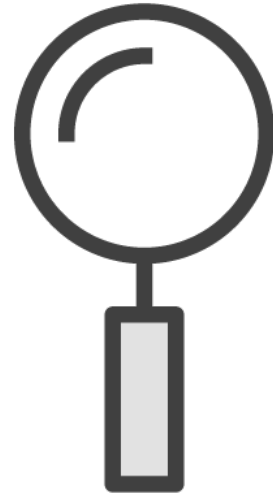
Abstraction



Simplifying reality

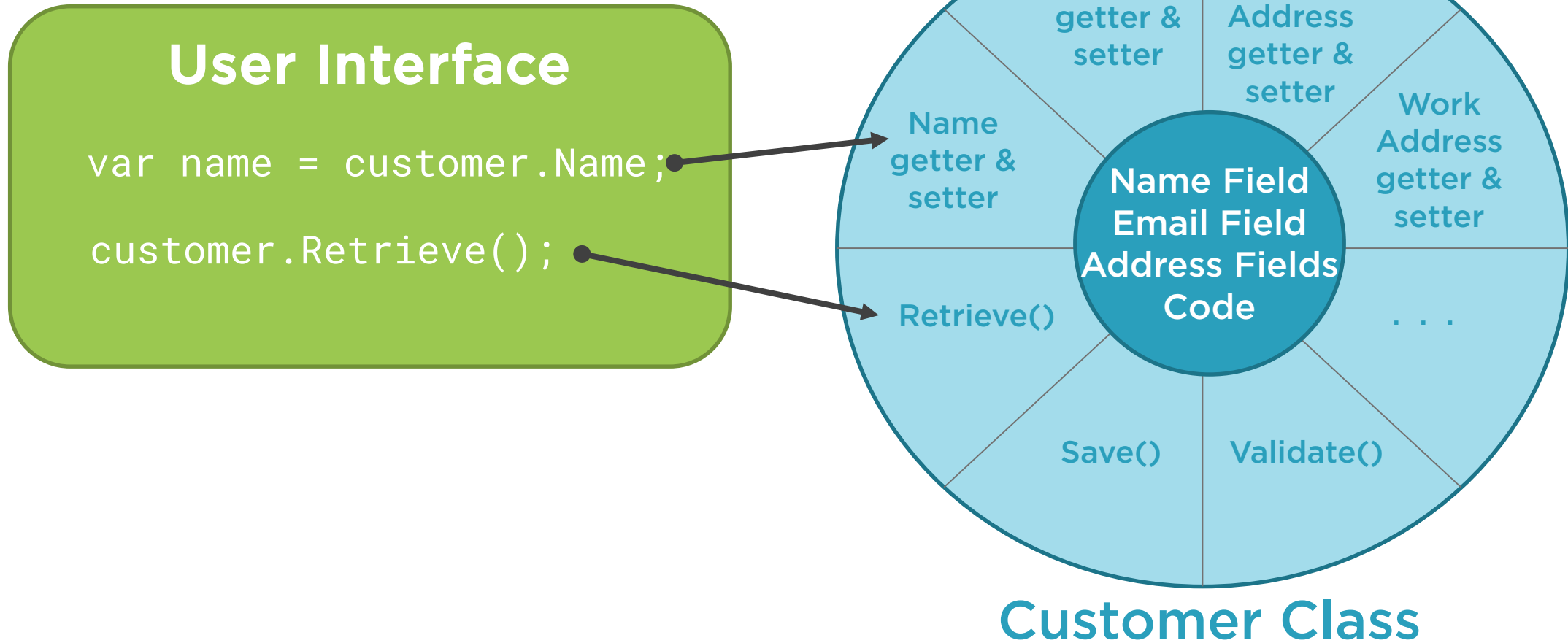


Ignoring extraneous
details

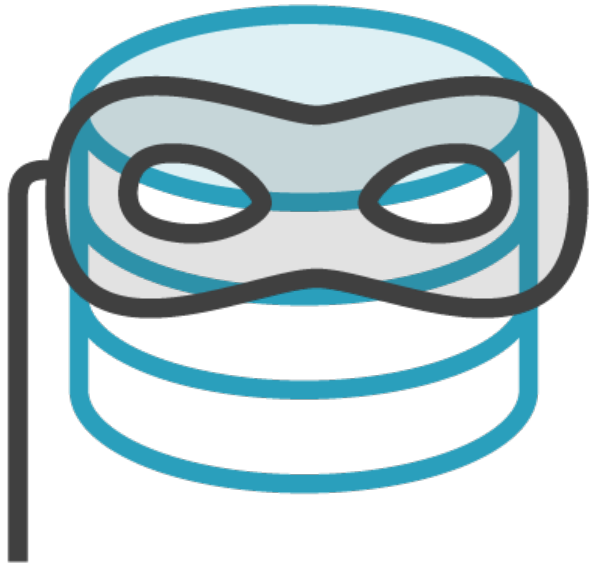


Focusing on what is
important for a
purpose

Encapsulation



Encapsulation

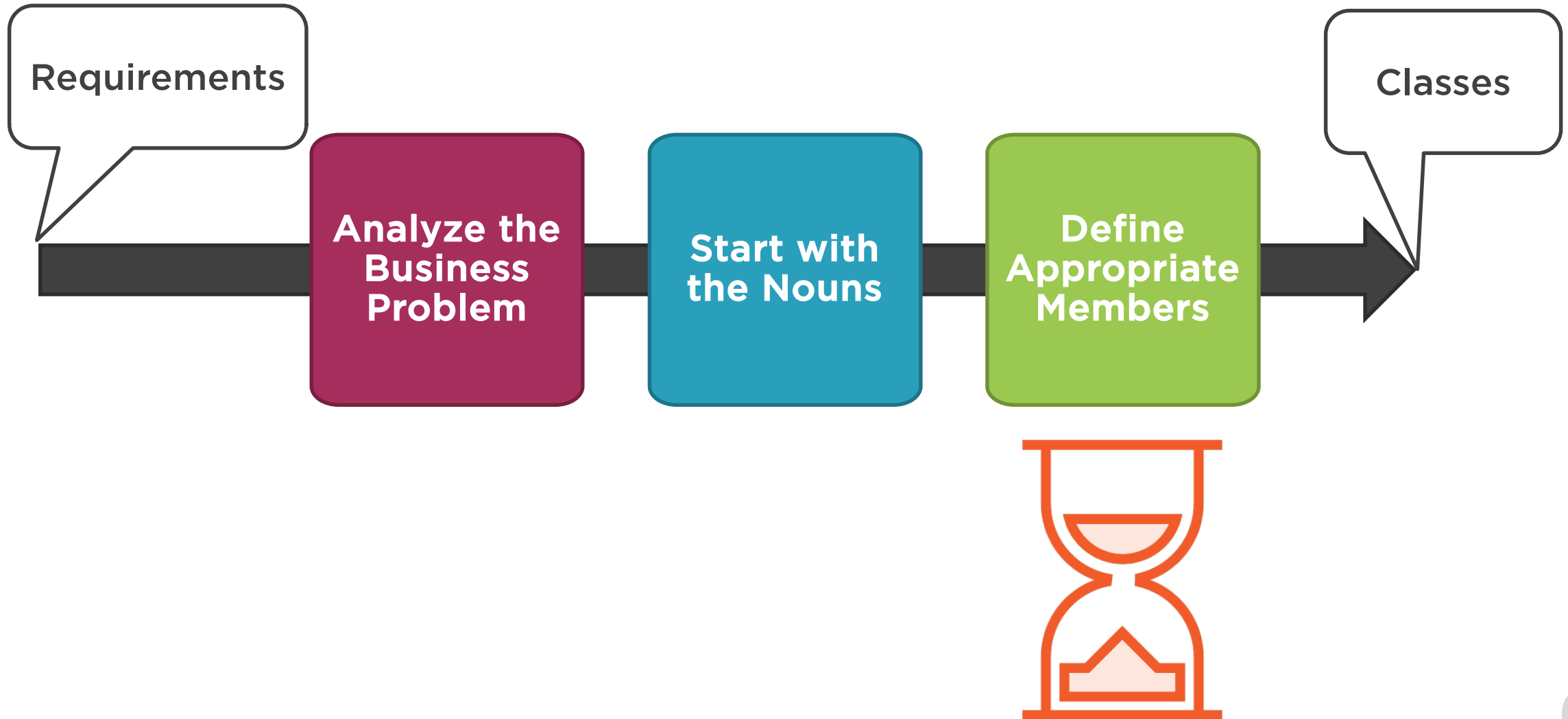


- Protects the data
- Allows for authorization before getting the data
- Allows for validation before setting the data



- Helps manage complexity
- Only the class needs to understand the implementation
- Implementation can be changed without impacting the application

Identifying Classes



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 textured blue surface. The first pillar on the left is labeled 'Abstraction' and the second pillar is labeled 'Encapsulation'. The third and fourth pillars are empty. The pillars rest on a series of three horizontal steps that form the base of the structure.

Abstraction

Encapsulation



Object-Oriented Programming (OOP)

**Identifying
classes**



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

**Separating
responsibilities**

**Establishing
relationships**

**Leveraging
reuse**

