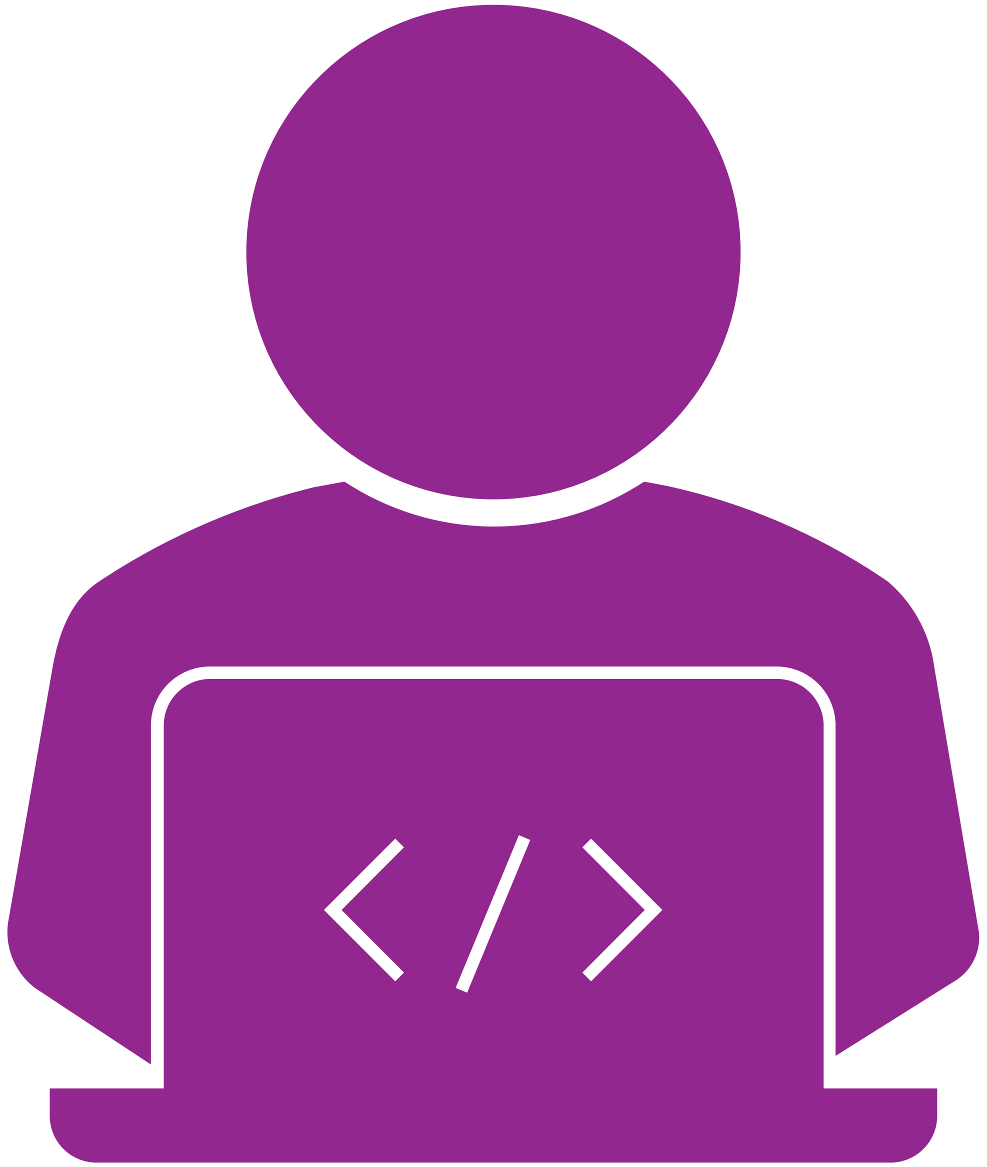
CLASSES

And Object Oriented Programming



What is object oriented programming?

- Some people ask this question
- There is no good answer
- Usually it is described as a set of principles
- Varies somewhat between languages

Principles of Object Oriented Programming

Abstraction – separation of implementation from usage (abstract classes)

Encapsulation – bundling data and methods together

Inheritance – ability to reuse code and represent is-a relationships

Polymorphism – ability to apply functions to different types (like generic methods)

Information hiding – hiding implementation details (related to encapsulation)

Identity – the fact that each instance of an object is separate and can be identified

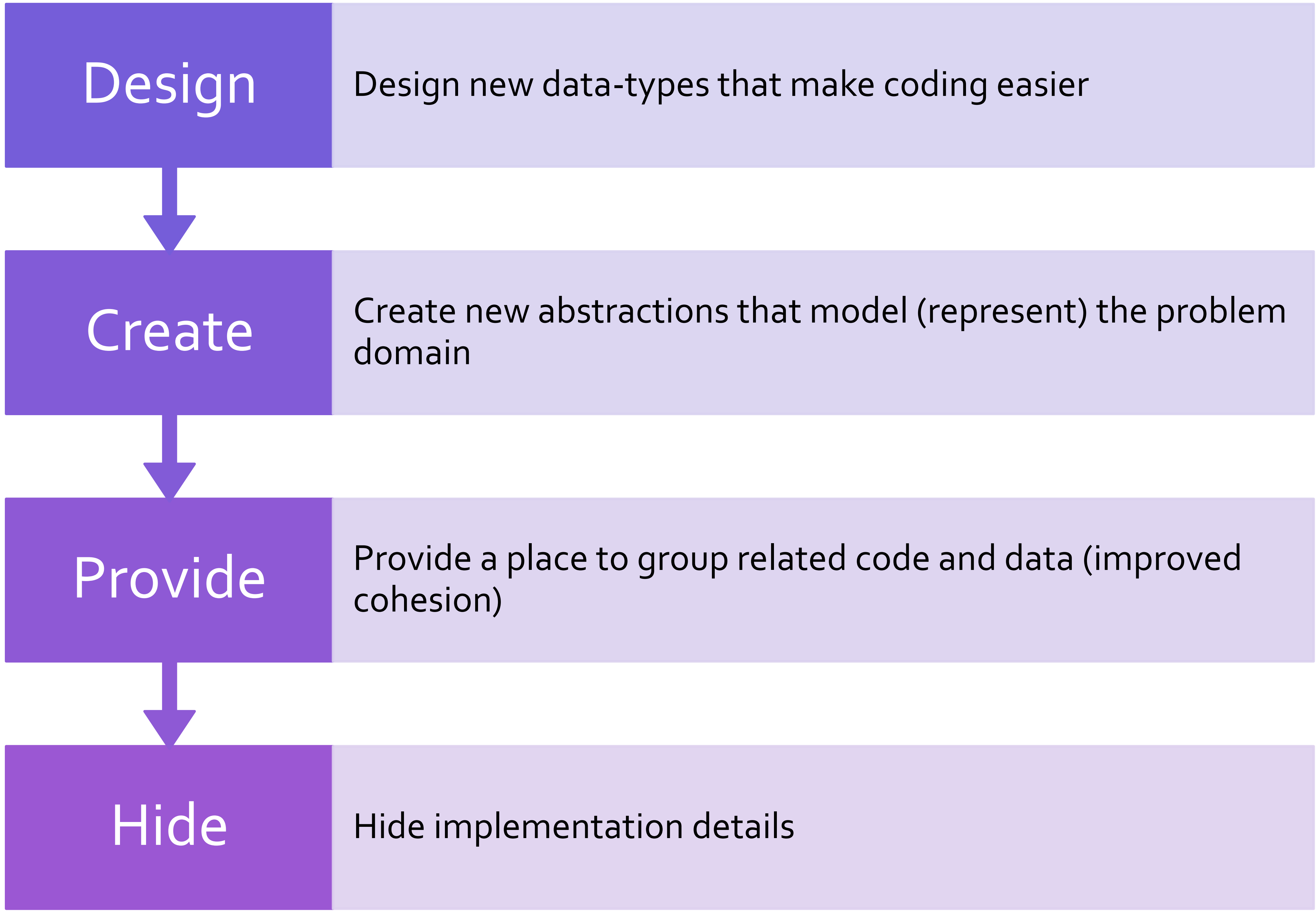
Those principle are not that useful

They are not formal definitions or principles

They are subjective and have been debated for years

Consider them guideposts to help gain an intuition for OOP

But Classes are Useful



Design

Design new data-types that make coding easier

Create

Create new abstractions that model (represent) the problem domain

Provide

Provide a place to group related code and data (improved cohesion)

Hide

Hide implementation details

Grouping of Data and/or Methods

Classes are ways to group related data and/or functions

The data element within a class are called “fields”

The functions within a class are called “methods”

Together they are called “members”

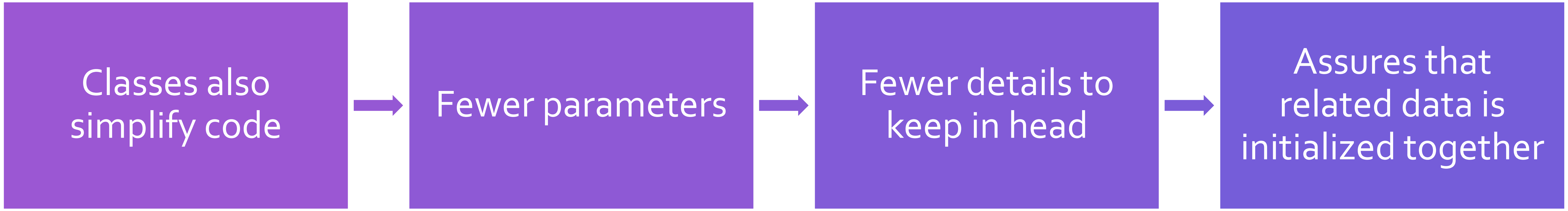
Non-Static Classes

- May have non-static members
- Non-static classes can be instantiated
- Supports the new operator
- May have one or more constructor
- Different instances may have different values for fields
- May also contain static methods and fields if desired

Examples of Instance Methods

- Functions that query properties
  - String.Length
  - List.Count
- Functions that transform the value into another
  - Int.ToString()
  - String.ToLower()
- Functions that change the data on the class instance
  - List.Add()
- Operations
  - String.CompareTo()
  - String.Substring()

Managing Complexity



Classes also simplify code

Fewer parameters

Fewer details to keep in head

Assures that related data is initialized together

Without Classes

- Some tasks become quite challenging
- Have to overload existing data types (like ints or strings) to represent concepts
- Implicit rules not captured or expressed
- Hard to verify
- Leads to errors
- Leads to complexity

Properties

A property is a member that looks like a field but calls a getter function

Read-write properties also allow assignment, which calls a setter function

Properties may also map directly to hidden fields, but this is transparent to a user

Instantiating a Class

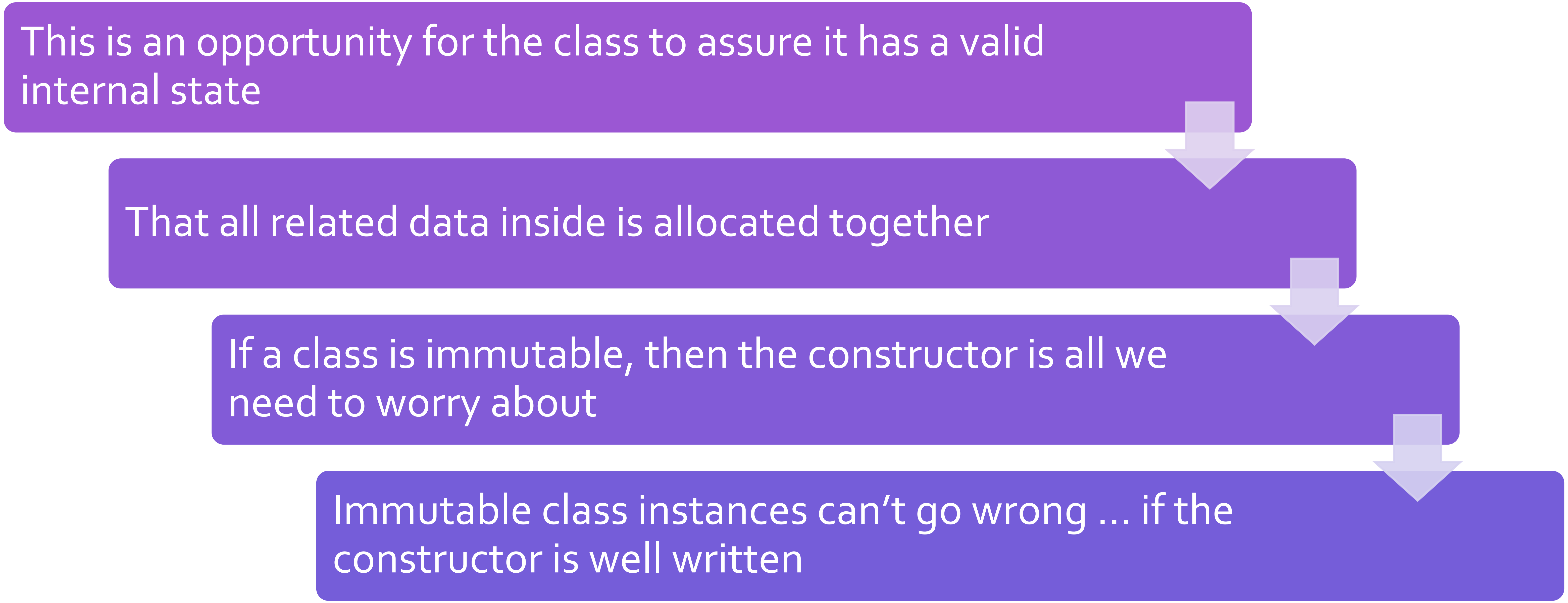
When we call “new” on a class we create an instance of it

This allocates memory for the class instance on the “heap”

All fields are initialized to the default values

The appropriate constructor method is called

Constructors



This is an opportunity for the class to assure it has a valid internal state

That all related data inside is allocated together

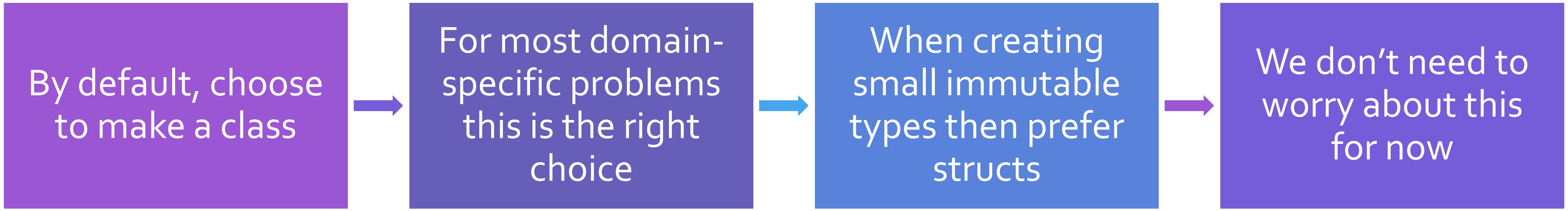
If a class is immutable, then the constructor is all we need to worry about

Immutable class instances can’t go wrong … if the constructor is well written

Well Designed Class

- Has a single clearly defined purpose
- Can’t be transformed into an inconsistent state
- Does not store redundant information
- Can be reused without change
- Does not expose implementation details
- Does not have public fields

Choosing Between Struct and Class



By default, choose to make a class

For most domain-specific problems this is the right choice

When creating small immutable types then prefer structs

We don’t need to worry about this for now

Class as a Data Types

Classes are ways to create new data-types

They might represent a new kind of collection

Or a new type of simple value like a number

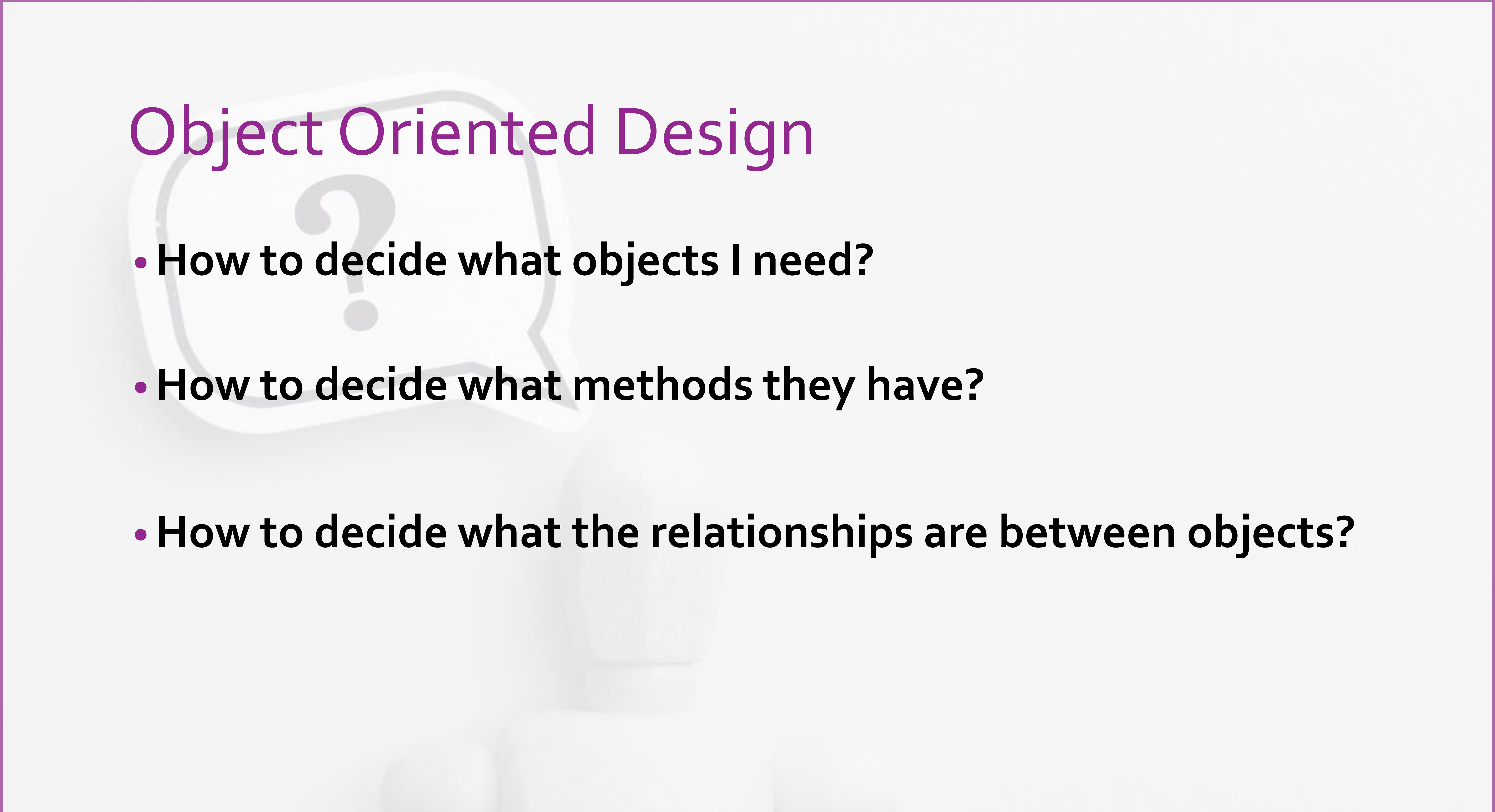
Abstractions

A class is also useful as an abstraction

A way of representing and identifying an idea or concept

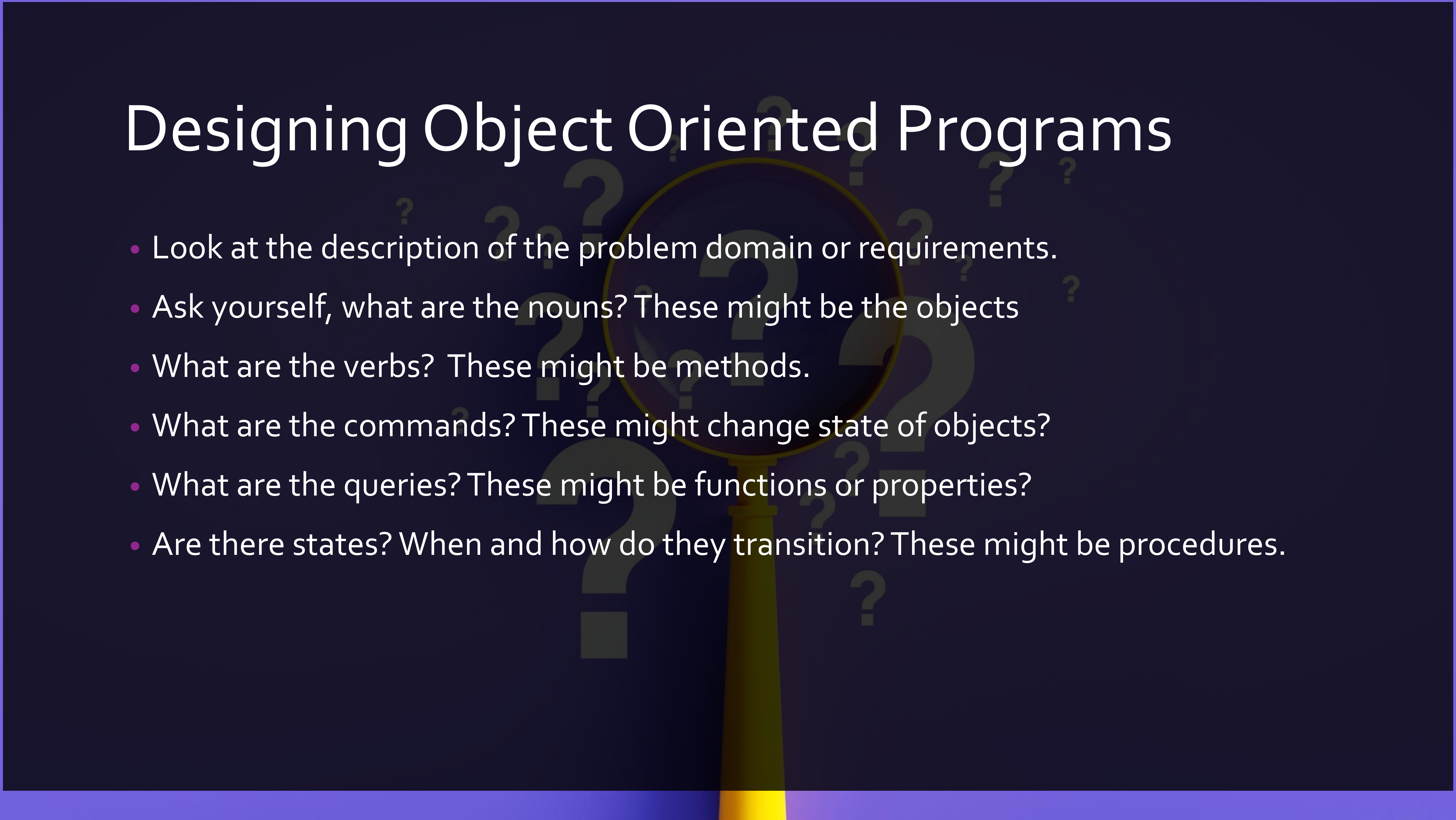
Allows us to work with something, without worrying about the details

Object Oriented Design



- How to decide what objects I need?
- How to decide what methods they have?
- How to decide what the relationships are between objects?

Designing Object Oriented Programs



- Look at the description of the problem domain or requirements.
- Ask yourself, what are the nouns? These might be the objects
- What are the verbs? These might be methods.
- What are the commands? These might change state of objects?
- What are the queries? These might be functions or properties?
- Are there states? When and how do they transition? These might be procedures.

Summary

Objects are a way of creating new data-types in an OOPL

Objects are an effective method for compartmentalizing complex problems into simpler ones.