

# Module 11: "Façade"



# Agenda

- ▶ Introductory Example: Long Walks
- ▶ Challenges
- ▶ Implementing the Façade Pattern
- ▶ Pattern: Façade
- ▶ Overview of Façade Pattern
- ▶ A Slight Word of Warning



# Introductory Example: Long Walks

```
IBluetoothSettings bluetooth = settingsManager.Bluetooth;  
bluetooth.IsEnabled = false;  
string trackerAppName = "Exomondo";  
applicationController.Start(trackerAppName);  
...  
applicationController.Stop(trackerAppName);  
bluetooth.IsEnabled = true;
```

```
interface ISettingsManager  
{  
    IBluetoothSettings Bluetooth { get; }  
}
```

```
interface IApplicationController  
{  
    bool Start( string name );  
    void Stop( string name );  
}
```



# Challenges

- ▶ Do we really have to go through that every time?
- ▶ Does every client need to figure out the logic of all the subsystems?

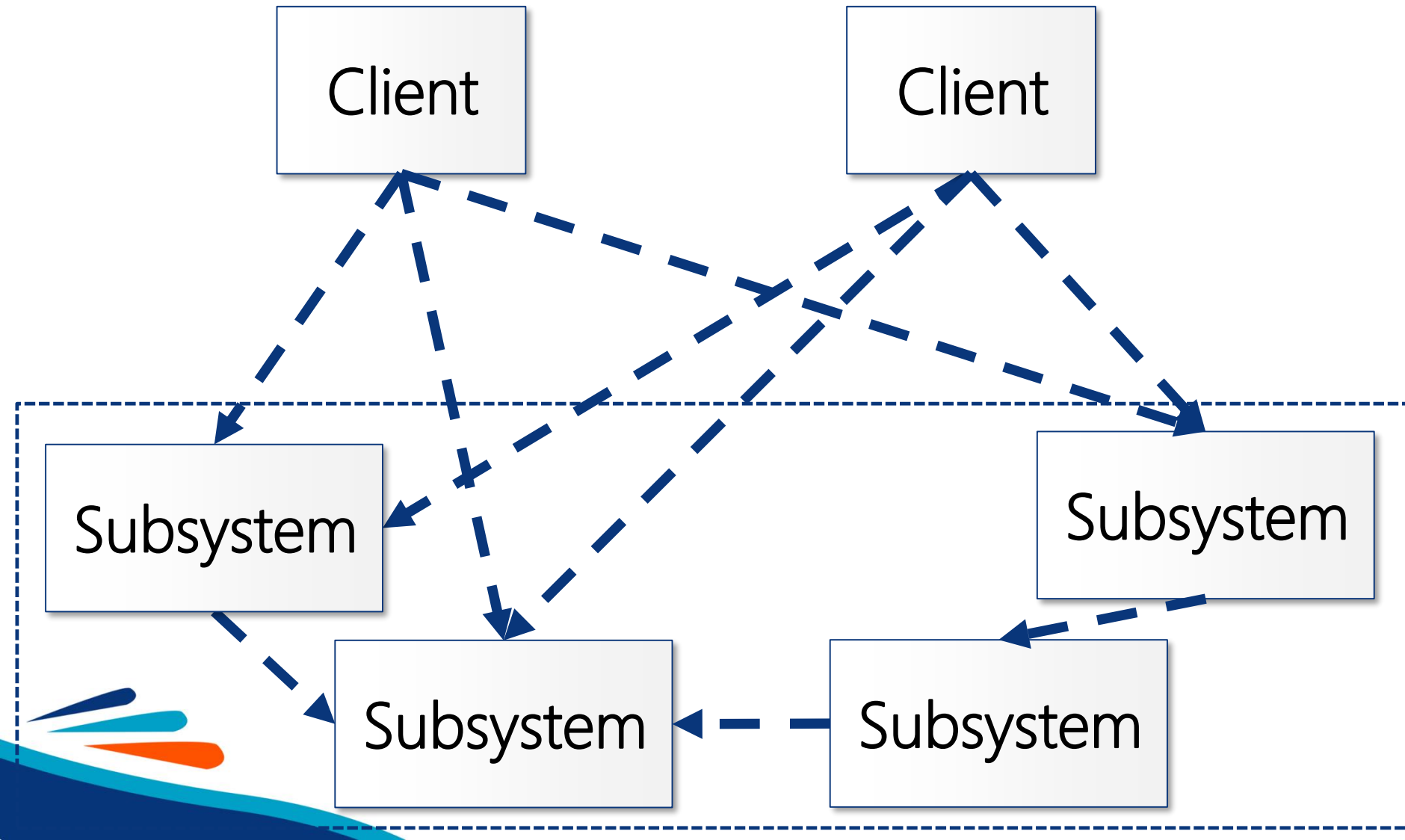


# Pattern: Façade

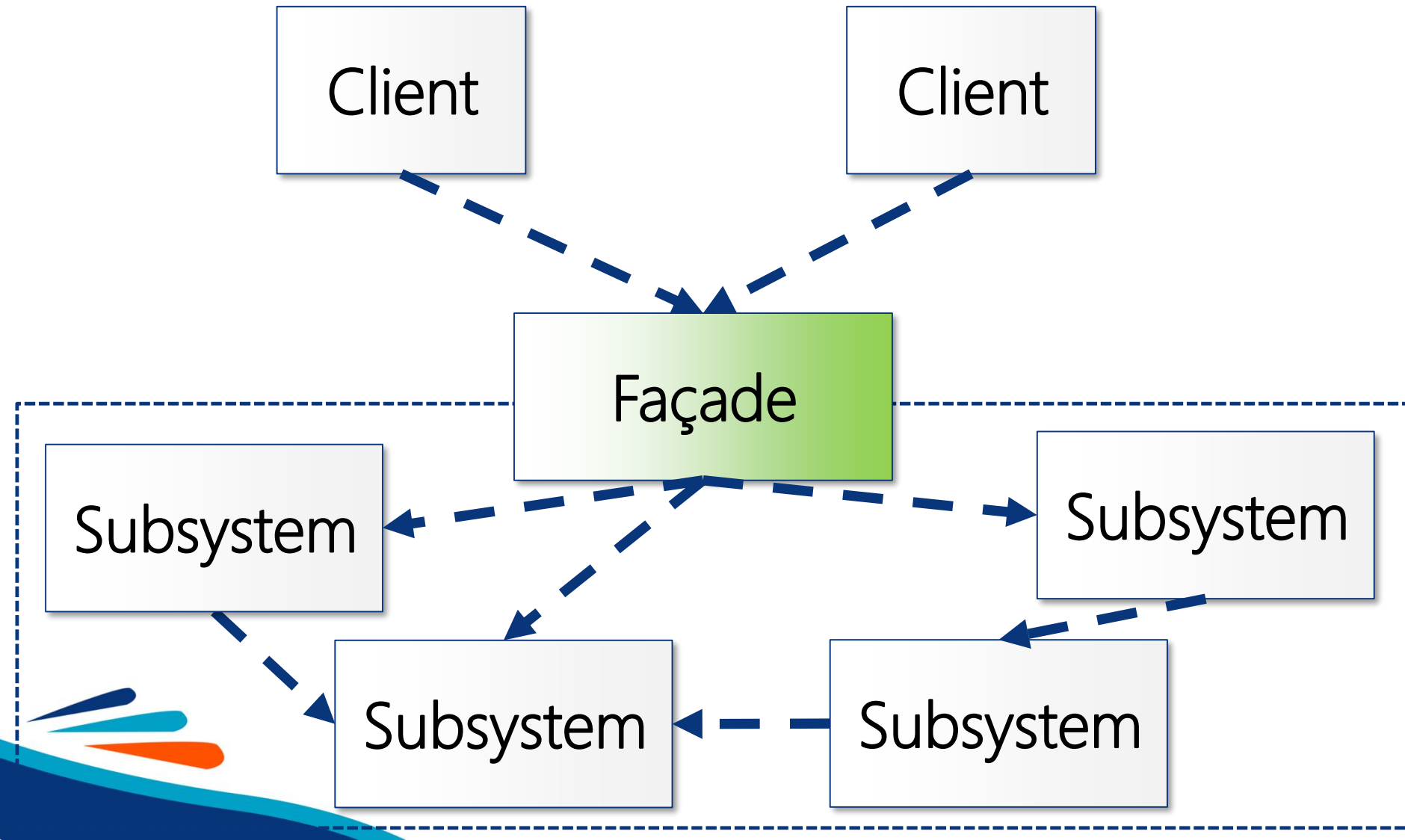
- ▶ *Provide a unified interface to a set of interfaces in a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use.*
- ▶ Outline
  - Isolate subsystems' intricacies from client
  - Make subsystem functionality reusable by clients
  - Loosely couple clients and subsystems
- ▶ Origin: Gang of Four



# Overview of Façade Pattern



# Overview of Façade Pattern



# Overview of Façade Pattern

- ▶ Client
  - Accesses Subsystems only through Façade
- ▶ Façade
  - Provides an entry point for Clients
  - Shields client from subsystem intricacies and complexities
  - Loosely couples Client from Subsystems
  - Makes Subsystems replaceable by other Subsystems without affecting the Clients
- ▶ Subsystems
  - Are left unchanged





# A Slight Word of Warning

- ▶ Do take a little of care in not “overfacading” everything
  - Overlaying
- ▶ Carefully consider situations where same subsystems appears in multiple façades
  - Versioning
  - Shared data contexts
  - ...
- ▶ Remember to implement the Disposable Pattern on your Façade if necessary!



