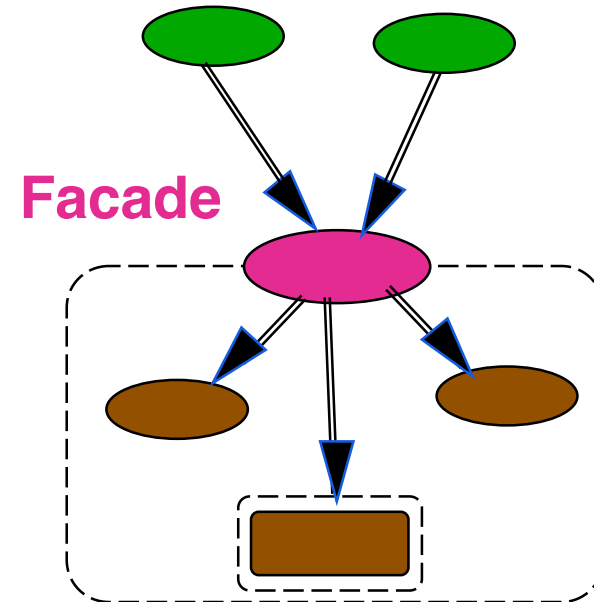
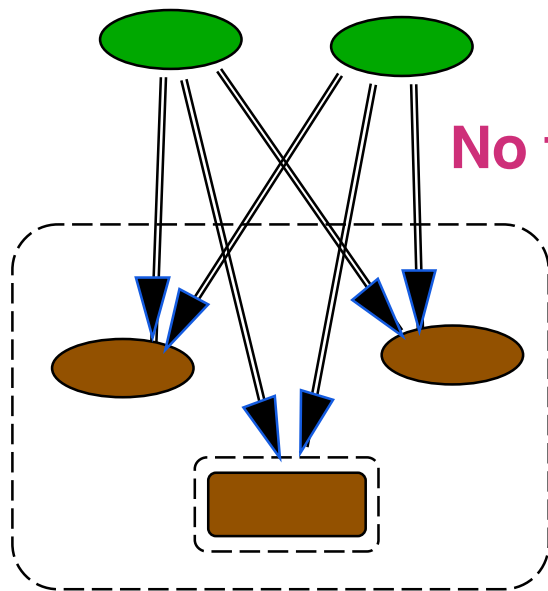


# Facade Pattern – Structural

- Intent
  - » **Provide common interface to a set of interfaces within system**
  - » **Define a higher level interface that makes the system easier to use for most common tasks**
- Motivation
  - » **Design goal is to minimize communication between client and subsystems of a system**
  - » **Facade provides a simplified interface to the more general facilities of a system**

# Facade Pattern – Diagram

Clients



Subsystem  
classes

## Facade – Applicability

- Need to provide a simple interface to set of complex subsystems
- Provide a simple default view

**As systems grow, classes become smaller more refined**

> **Better for reuse**

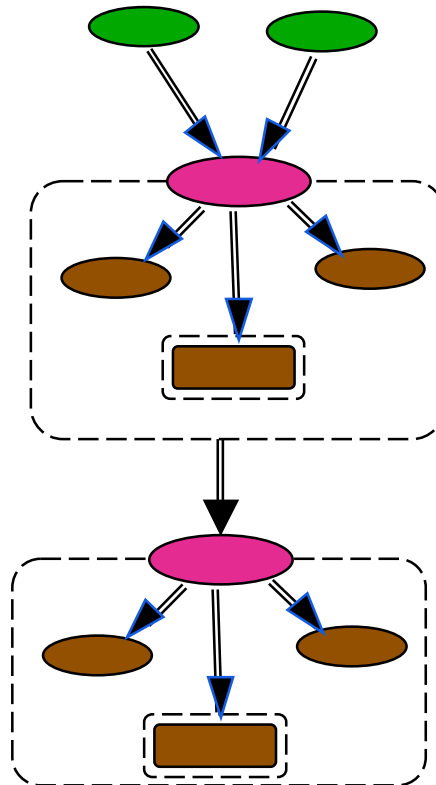
> **More difficult for clients to use**

- Decouple subsystems from clients

**Reduce implementation dependencies**

# Facade – Applicability – 2

- Layer subsystems
  - » Each layer has a single entry point
  - » Layers communicate only through Facade interface



# Facade – Compiler Example – Pseudocode

```
class COMPILER
  feature { NONE }
    nodeTree : NODE
    scanner : SCANNER // Individual subsystems
    parser : PARSE
    emitter : EMITTER
  feature
    compile is
      nodeTREE ← parser.parse ( scanner )
      emitter.output ( nodeTree )
    end
  end
end
```

# Facade – Compiler Participants

- Facade

- » **Compiler**

- > **Knows which subsystem classes are responsible for a request**
    - > **Delegates client requests to appropriate subsystem objects**

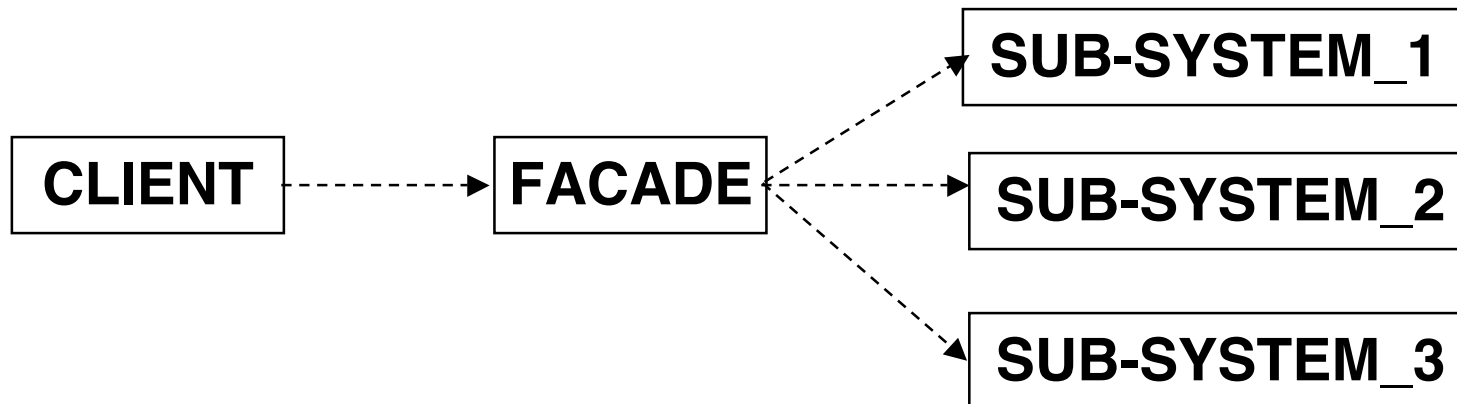
- Subsystems

- » **Scanner, Parser, StatementTypeNode(s), etc.**

- > **Implement system functionality**
    - > **Handle work assigned by Facade object**
    - > **Have no knowledge of the facade**
      - **Keep no references to it**

## Facade – Collaborations

- Clients communicate with the subsystem by sending requests to Facade
- Facade forwards requests to subsystem
  - » **Facade may have to translate its interface to subsystem interface (use Adapter)**
- Clients that use facade don't have direct access to the subsystems



# Facade – Consequences

- Benefits

- Shields clients from subsystem components**

- Reducing number of objects clients deal with**

- » Promotes weak coupling between subsystems and clients**

- Can vary components of subsystem without affecting clients**

- » Doesn't prevent expert clients from direct access to subsystems**

- Choice between ease of use and generality**



## Façade – Related Patterns

- Abstract Factory is used with Façade to provide an interface of creating subsystems independent of the subsystems.
- Mediator abstracts arbitrary communication between objects by centralizing functionality that does not properly belong to either of them. Instead of direct communication, objects go through the mediator
- Facade objects are often Singletons