that cause a catastrophic failure in your API layer. You need
called in code bases outside the one your API classes are in
called in code bases outside the one your API classes are in
then, but you have discovered that those classes are being
then, but you have discovered that those classes are being
the right order they produce the desired code implementa-
the right order they produce the desired code implementa-
(transaction program interface). Each class has an interface
(transaction program interface). Each class has an interface
(
assembly) that we want to make into an API
assembly) that we want to make into an API
For our functional example, we have a group of classes that
For our functional example, we have a group of classes that

**Called by code outside the assembly**

**Control the order in which those classes are**

**Methods make up a process, but we need to**

**Problem:** We have several classes whose

**Figure 4.2: UML for Facade Pattern**

![](image)

of the entire subsystem structure.

of the entire subsystem structure.

class in its proper calling order encapsulating

class in its proper calling order encapsulating

- The facade could change access control to each
- In the facade, calling order is preserved, a transaction fails.
- The facade could change access control to each
- In the facade, calling order is preserved, a transaction fails.

No outside access to these classes would be permitted.
No outside access to these classes would be permitted.

Methods on the facade interface then could
Methods on the facade interface then could

Facade Pattern