3.7 Modell Management Sicht

- *Paket* (Package) = allgemeines Konstrukt zum Zusammenfassen von Modellierungselementen und weiteren Paketen
- Modellierungselemente
  - Klassen
  - Zustandsmaschinen
  - Anwendungsfalldiagramme
  - Interaktionsdiagramme

Pakete

- Jedes Element gehört zu genau einem Paket
- Es gibt anonymes *Wurzel Paket* (root package)
- Pakete lassen sich ineinander schachteln
  $\Rightarrow$ Baumstruktur

Aufteilung in Pakete

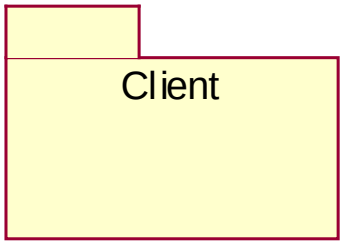
- Wie unterteilt man am besten in Pakete?
  - Zusammenfassung von Elementen, die semantisch eng zusammenhängen und viele Abhängigkeiten aufweisen
  - Unterteilung eines Systems in weitestgehend unabhängige und selbstständige Subsysteme
- **Ziel**: Abhängigkeiten minimieren

Pakete - Nutzen

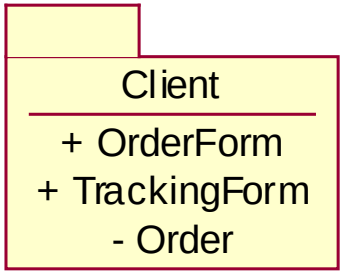
- Vorteile der Aufteilung in Pakete:
  - besseres Verständnis durch Betrachtung von Teilsystemen bzw. Systemausschnitten
  - Vermeidung von Namenskonflikten
  - Zugriffskontrolle für Elemente in Paketen, Kapselung
  - Vereinfachung der Testphase durch separates Testen von Paketen

Pakete

- **Graphische Darstellung**:



- Enthaltene Elemente graphisch oder textuell



Pakete - Sichtbarkeit

- Jedem Element innerhalb eines Paketes kann eine Sichtbarkeit zugeordnet werden:
  - *private* (-)
    Nur innerhalb des Paketes darf auf das Element zugegriffen werden
  - *protected* (#)
    Zugriff zusätzlich in abgeleiteten Paketen erlaubt
  - *public* (+)
    Zugriff aus allen anderen Paketen erlaubt

Pakete - Sichtbarkeit (Forts.)

- Den öffentlichen (public) Teil eines Paketes bezeichnet man auch als *Schnittstelle* des Paketes

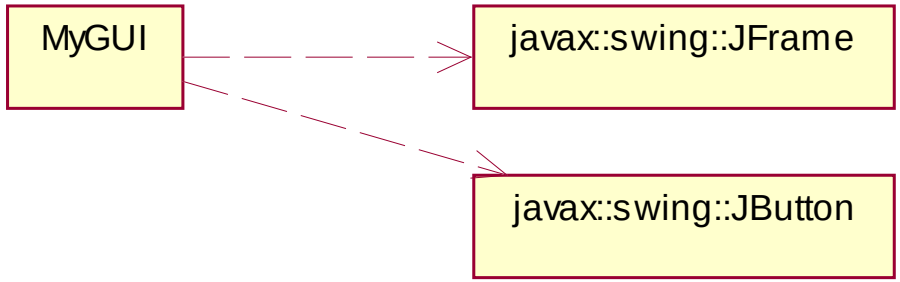
Pakete - Beziehungen

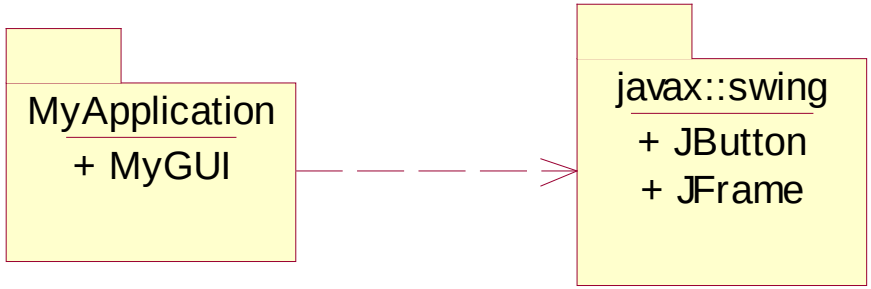
- Abhängigkeiten zwischen Paketen ergeben sich, wenn Abhängigkeiten zwischen den enthaltenen Elementen bestehen
- Mehrere Abhängigkeiten (auch verschiedener Art) werden zu einer einzigen Abhängigkeit abstrahiert

Pakete - Beziehungen (Forts.)

Beispiel:

Abhängigkeit von Klassen





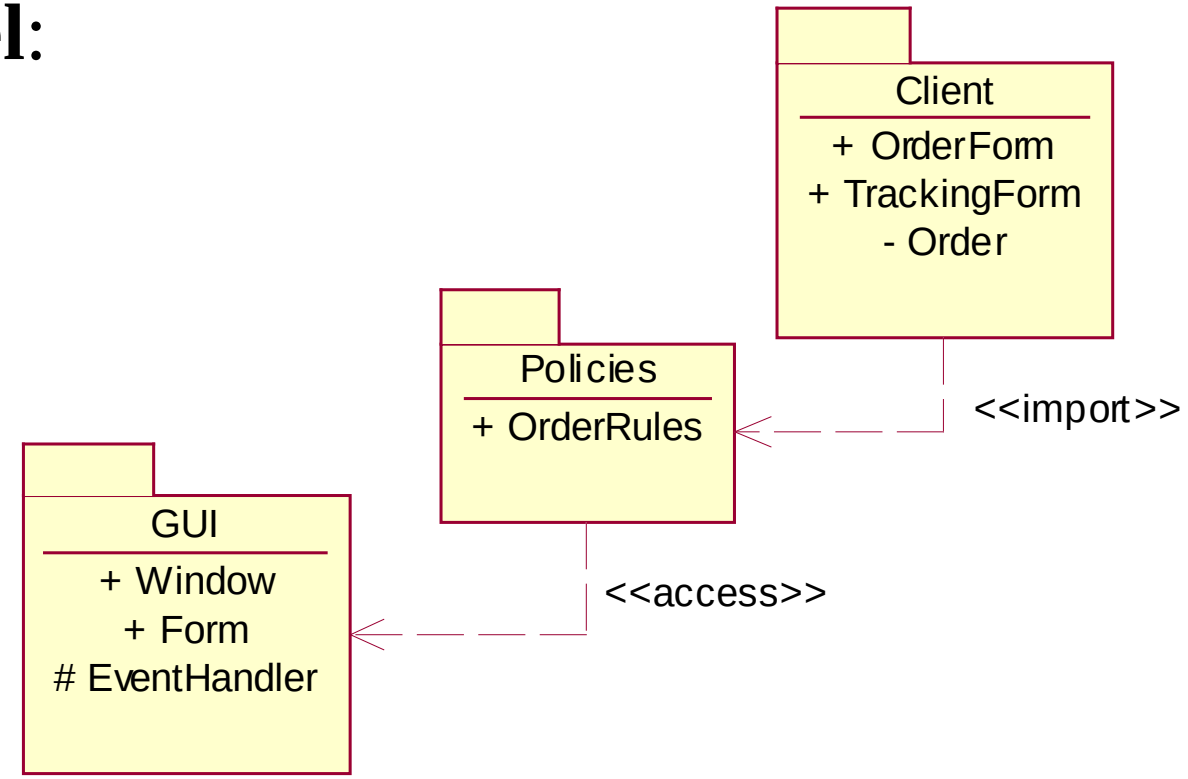
Abhängigkeit von Paketen

Pakete - Importieren/Exportieren

- Die öffentlichen Elemente eines Paketes sind von aussen zugreifbar; sie werden *exportiert*
- Zugriff von Elementen aus einem Paket auf Elemente aus anderen Paketen (*importieren*) muss explizit spezifiziert werden
  - Abhängigkeitsbeziehung mit Stereotyp `<<access>>` Zugriff durch Angabe des vollständigen Pfadnamens
  - Abhängigkeitsbeziehung mit Stereotyp `<<import>>` Angabe des einfachen Namens reicht

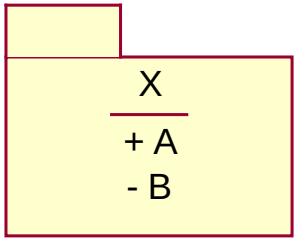
Pakete - Importieren/Exportieren (Forts.)

Beispiel:



Pakete - Zugriffsregeln/Sichtbarkeit

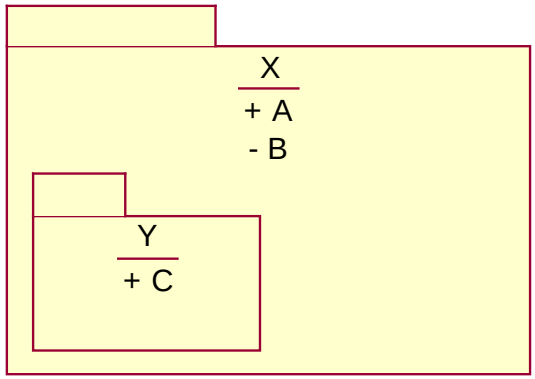
- Ein Element eines Paketes ist für alle anderen Elemente des Paketes sichtbar



  - A sieht B, B sieht A

Pakete - Zugriffsregeln/Sichtbarkeit (Forts.)

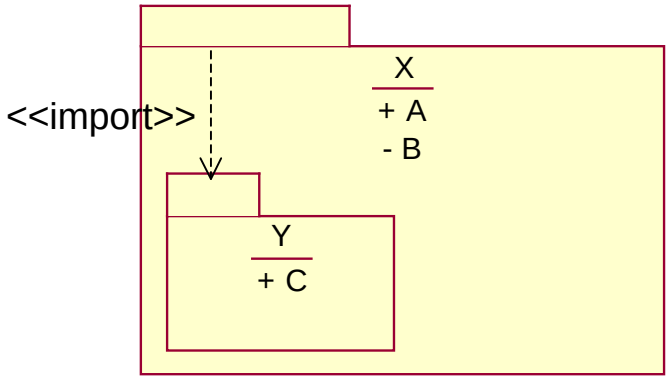
- Ein Element, welches sichtbar in einem Paket ist, ist auch in allen Unterpaketen sichtbar



  - C sieht A und B
  - A und B sehen C nicht

Pakete - Zugriffsregeln/Sichtbarkeit (Forts.)

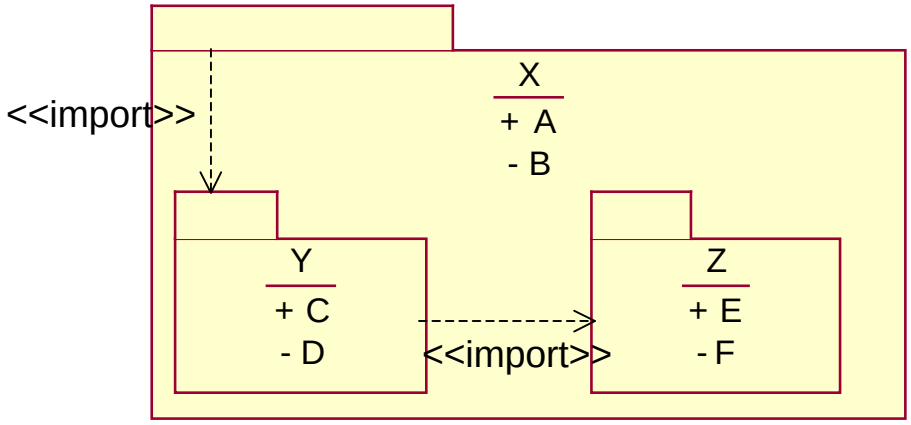
- Elemente eines Paketes können nur dann auf Elemente von Unterpaketen zugreifen, wenn diese importiert werden



  - A und B sehen C

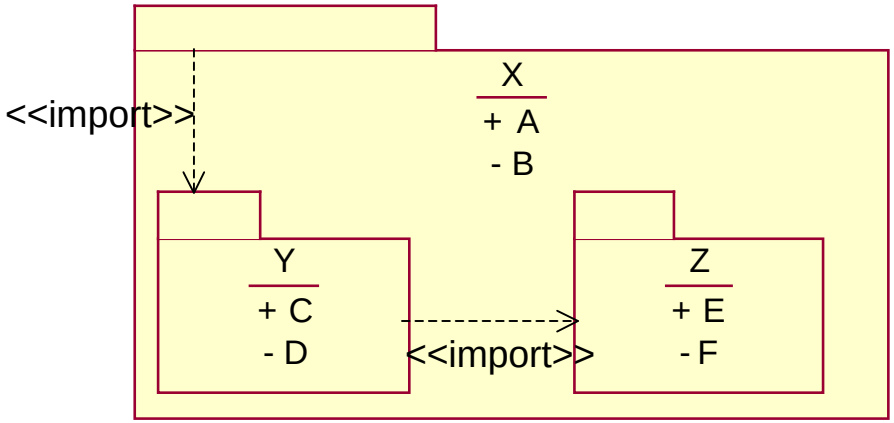
Pakete - Zugriffsregeln/Sichtbarkeit (Forts.)

- Importiert ein Paket ein anderes, so sind alle öffentlichen Elemente des importierten Paketes im importierenden Paket sichtbar
  - A und B sehen C, aber nicht D
  - C sieht E, aber nicht F



Pakete - Zugriffsregeln/Sichtbarkeit (Forts.)

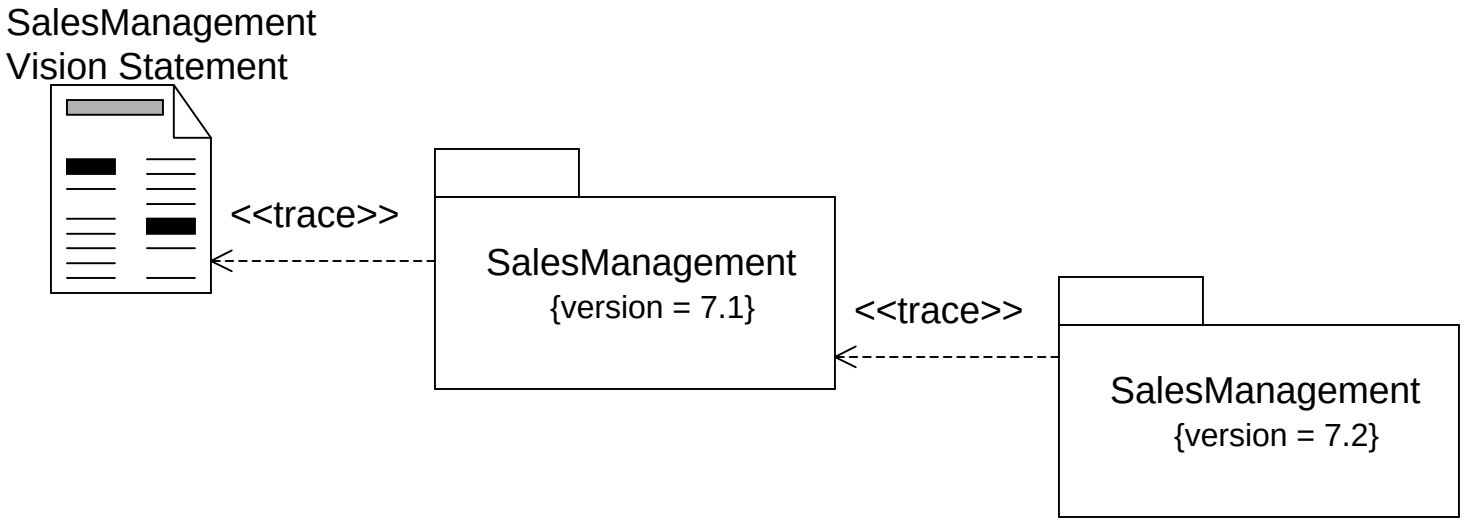
- Importieren ist *nicht* transitiv



  - A und B sehen C aber nicht E

Pakete - trace-Abhängigkeit

- Modellierung der Weiterentwicklung eines Elements durch `<<trace>>` Abhängigkeit

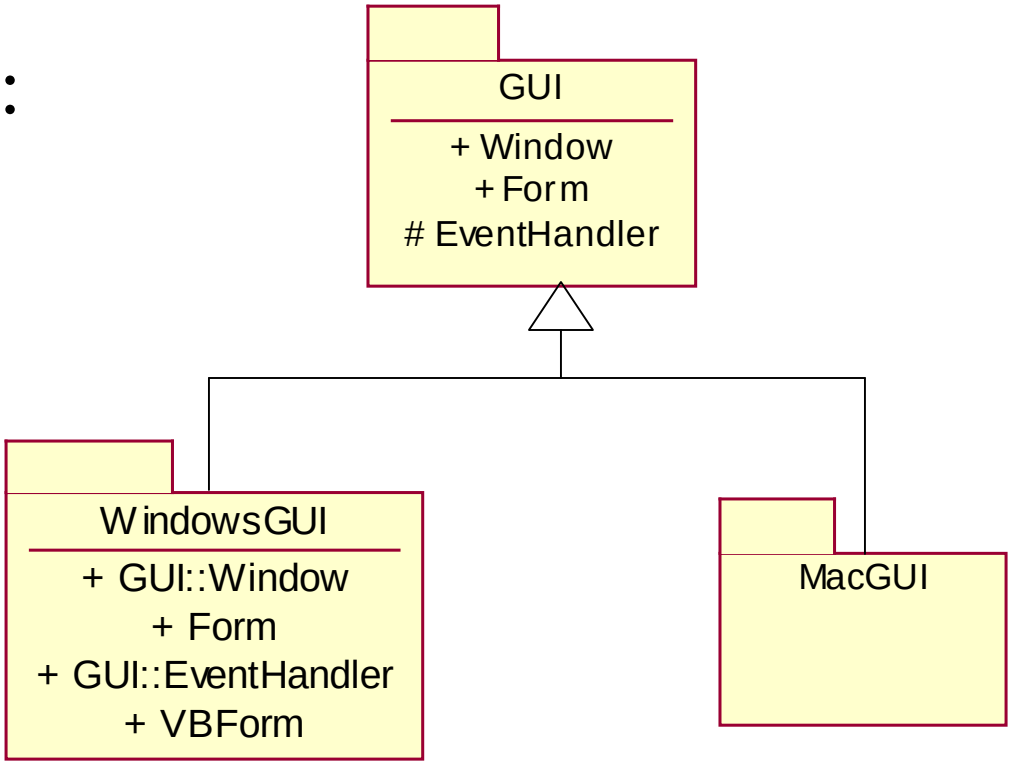


Pakete - Generalisierung

- Generalisierungsbeziehung bei Paketen:
  - Abgeleitete Pakete erben öffentliche (public) und geschützte (protected) Elemente des Oberpaketes
  - Elemente können in den abgeleiteten Paketen überschrieben werden

Pakete - Generalisierung (Forts.)

Beispiel:

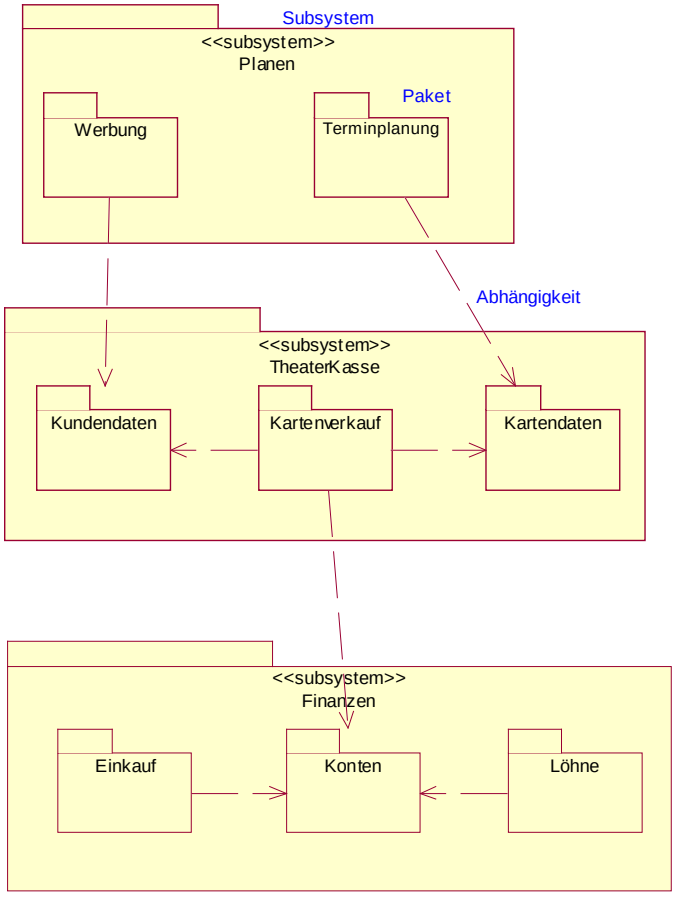


Pakete - Stereotypen

- `<<system>>`
  Paket, welches das ganze System enthält
- `<<subsystem>>`
  Teilsystem des ganzen Systems

Pakete - Stereotypen (Forts.)

Beispiel:



Pakete - Stereotypen (Forts.)

- `<<framework>>`
  Paket, welches hauptsächlich zusammen-arbeitende Entwurfsmuster enthält
- `<<stub>>`
  Paket, welches als Vertreter des öffentlichen Teils eines anderen Paketes fungiert

Pakete - Stereotypen (Forts.)

Beispiel: Stubs und Skeletons in CORBA

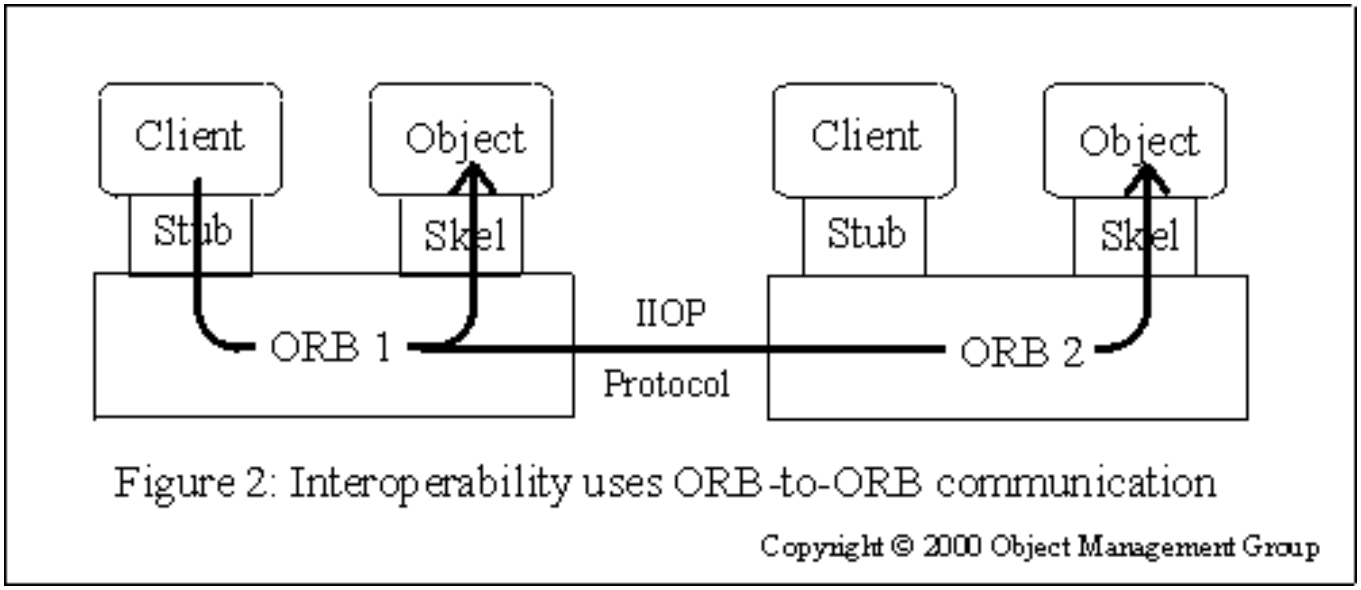


Figure 2: Interoperability uses ORB-to-ORB communication

Pakete - Stereotypen (Forts.)

- `<<facade>>`

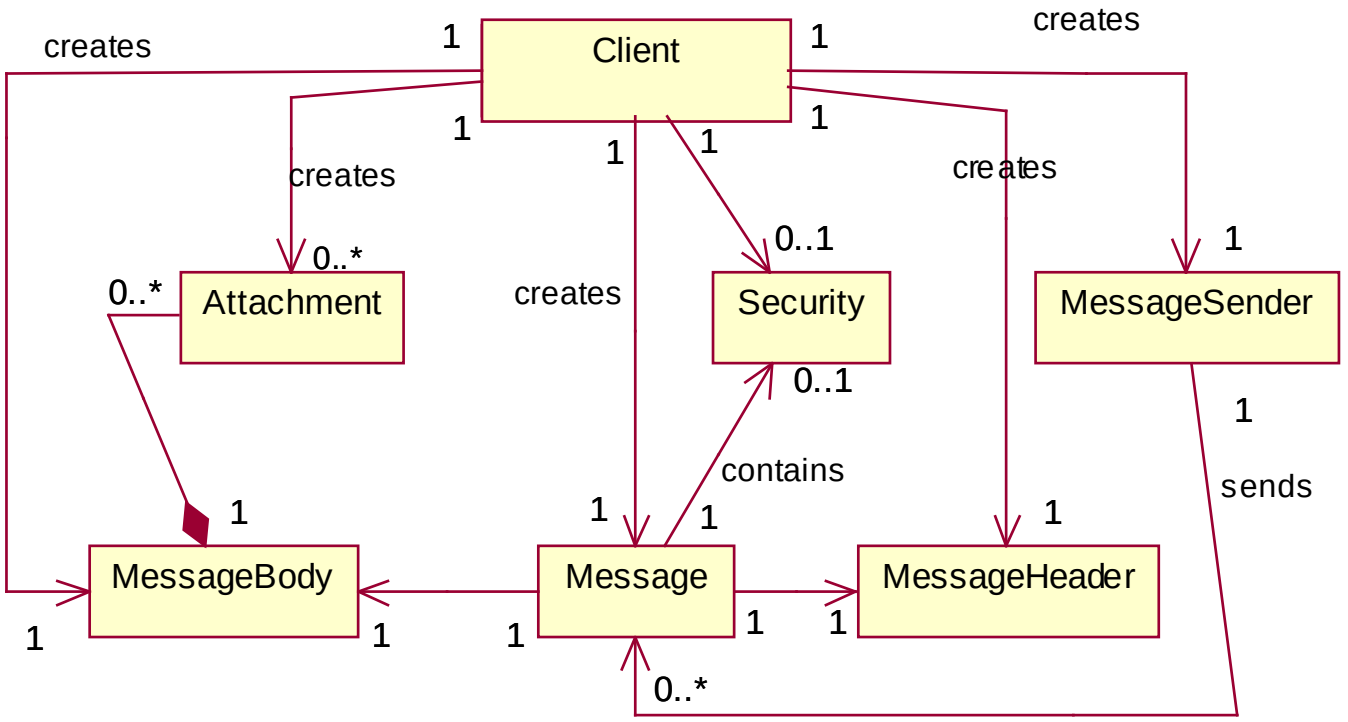
  Paket, welches vereinfachten Zugriff auf Elemente eines anderen komplexen Paketes zur Verfügung stellt

Entwurfsmuster - Facade

- *Facade* Entwurfsmuster
  - erleichtert den Zugriff auf eine Menge zusammengehöriger Objekte, indem ein Facade-Objekt zur Verfügung gestellt wird, welches deren Steuerung übernimmt
- Anwendung:
  - Vereinfachung der Benutzung einer Abstraktion
  - Verminderung der Abhängigkeiten zwischen benutzender Klasse und Abstraktion

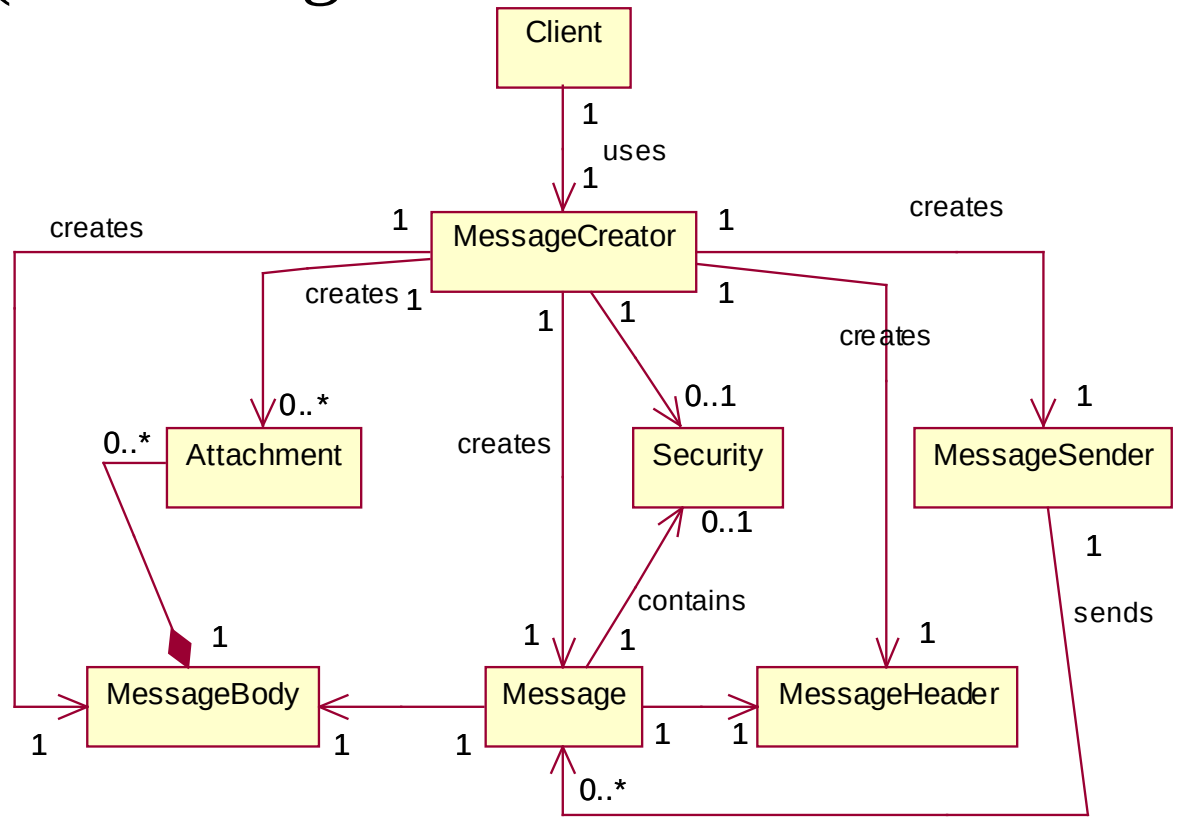
Entwurfsmuster - Facade (Forts.)

Beispiel (Zusammenstellung einer Email):



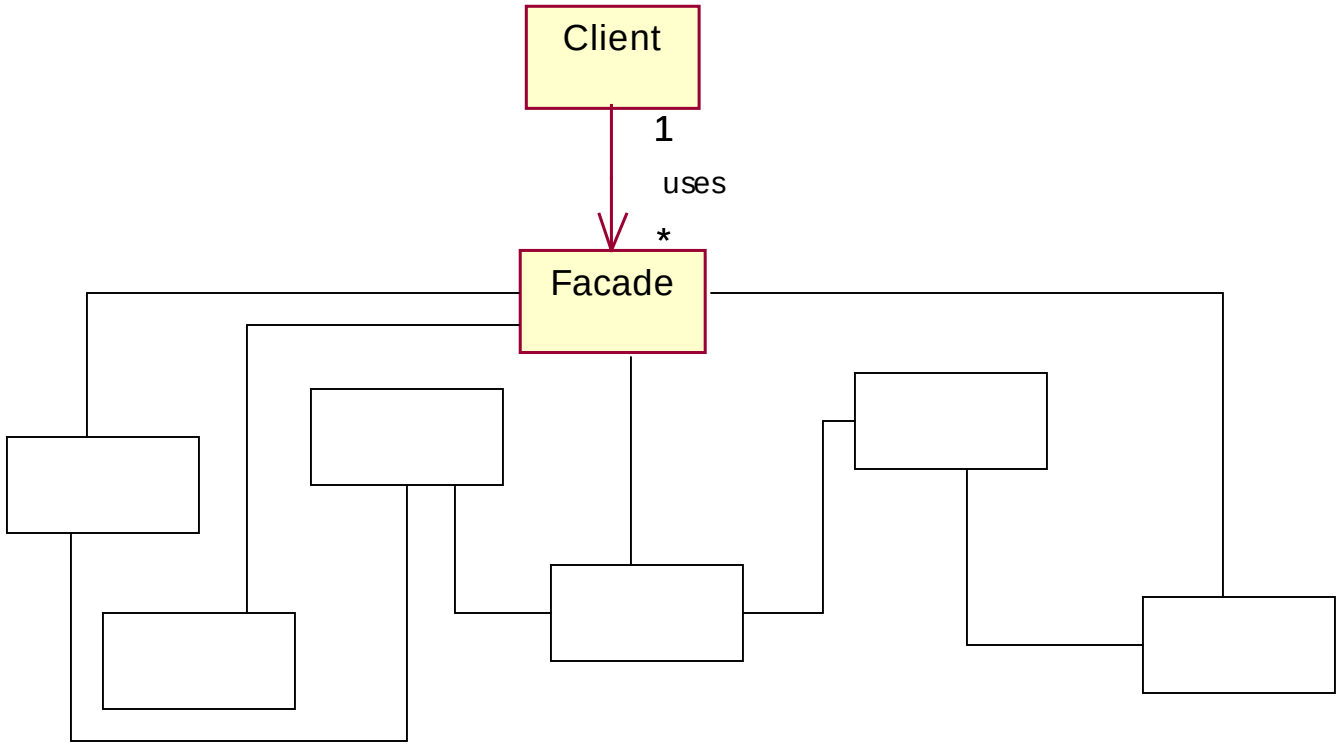
Entwurfsmuster - Facade (Forts.)

Beispiel (Erstellung einer Email mit Facade Objekt):



Entwurfsmuster - Facade (Forts.)

Abstrakte Modellierung:



Modell Management Sicht - Zusammenfassung

- Zusammenfassung und Strukturierung von Modellelementen durch *Pakete*
- Abhängigkeiten minimieren
- besseres Verständnis durch Betrachtung von Teilsystemen bzw. Systemausschnitten
- Zugriffskontrolle, Kapselung
- Vermeidung von Namenskonflikten