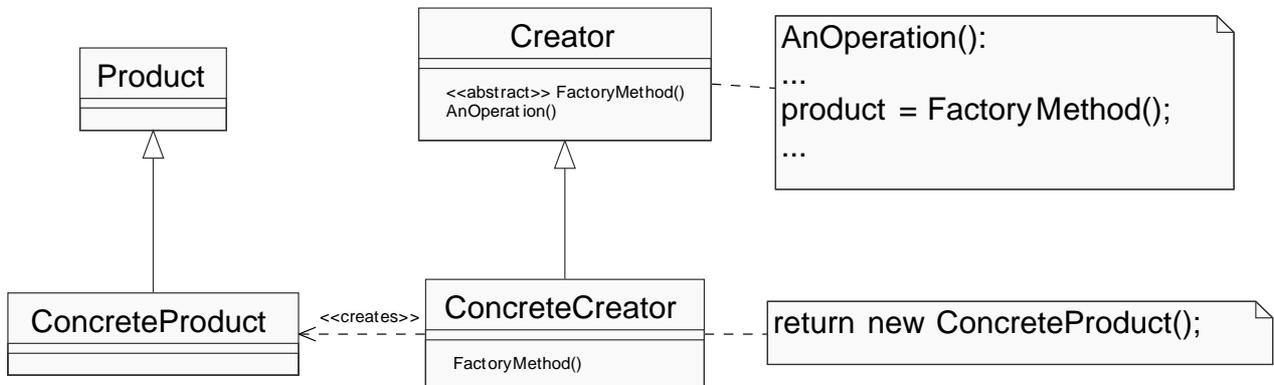
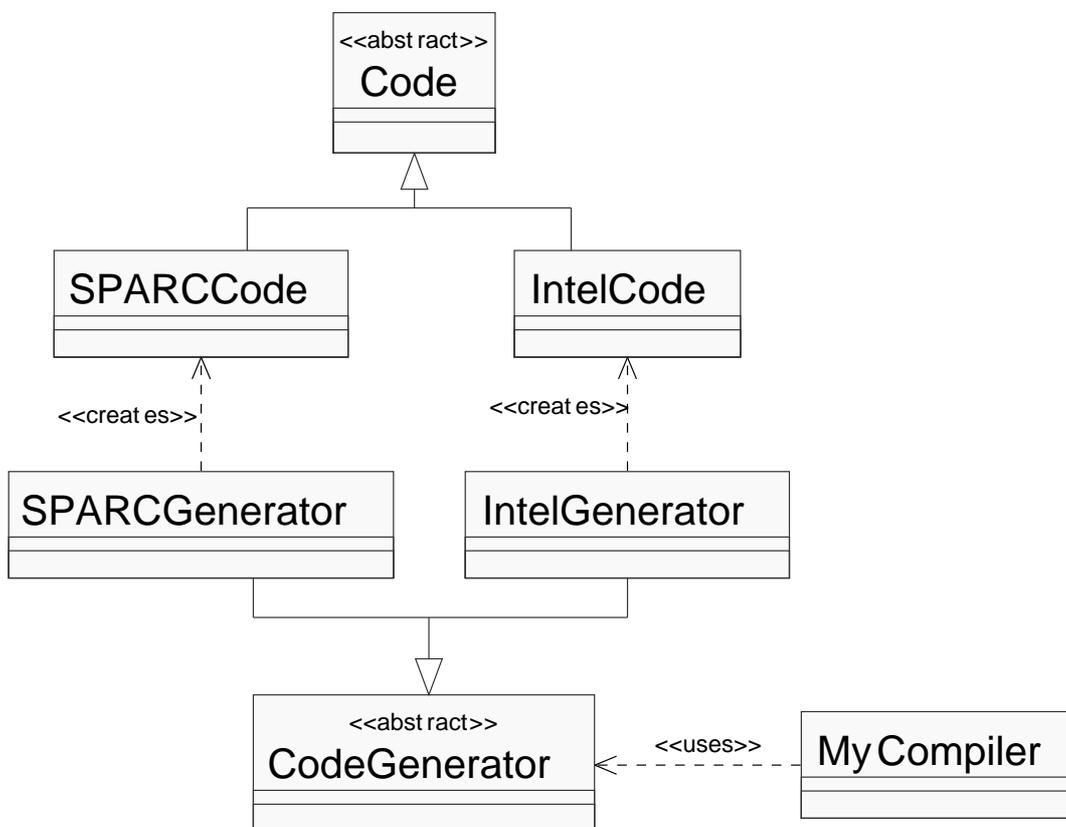


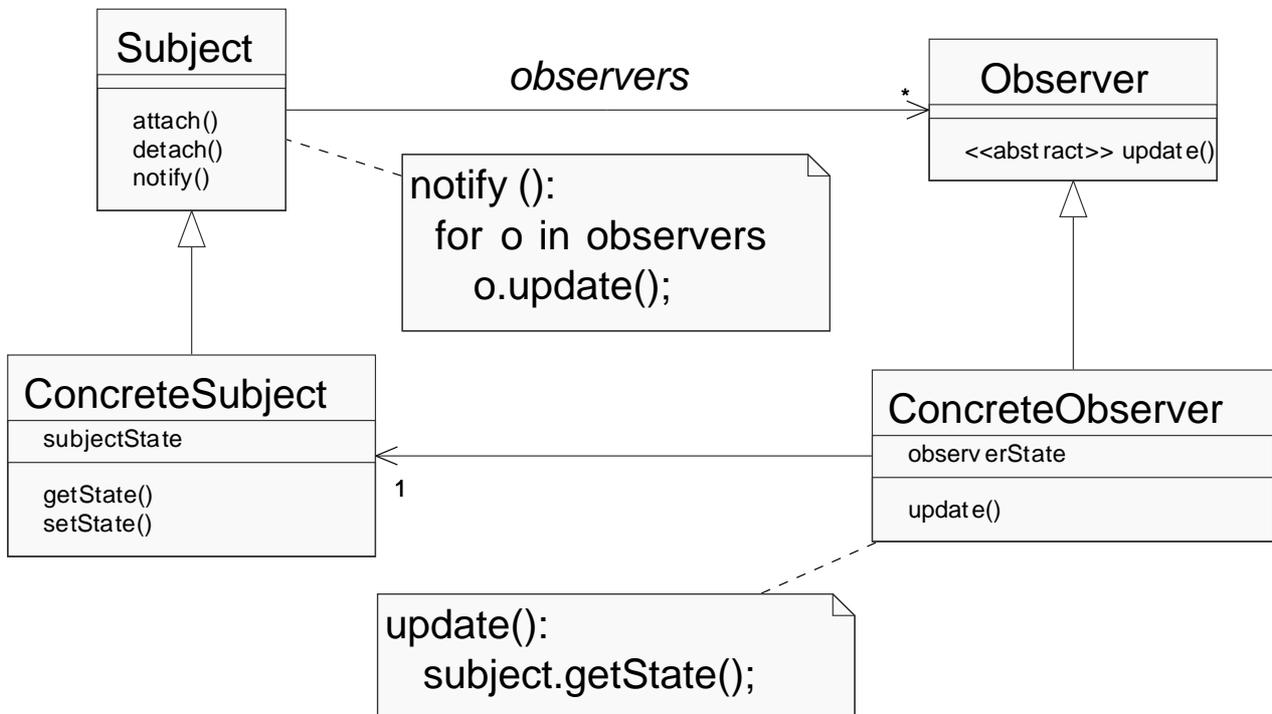
Das *Factory Method*-Pattern bietet die Schnittstelle zur Erzeugung eines Objektes an, wobei die Unterklassen entscheiden, von welchem Typ das konkrete Objekt ist. Die Fabrikmethoden entsprechen dabei typischerweise dem *Template Method*-Pattern.



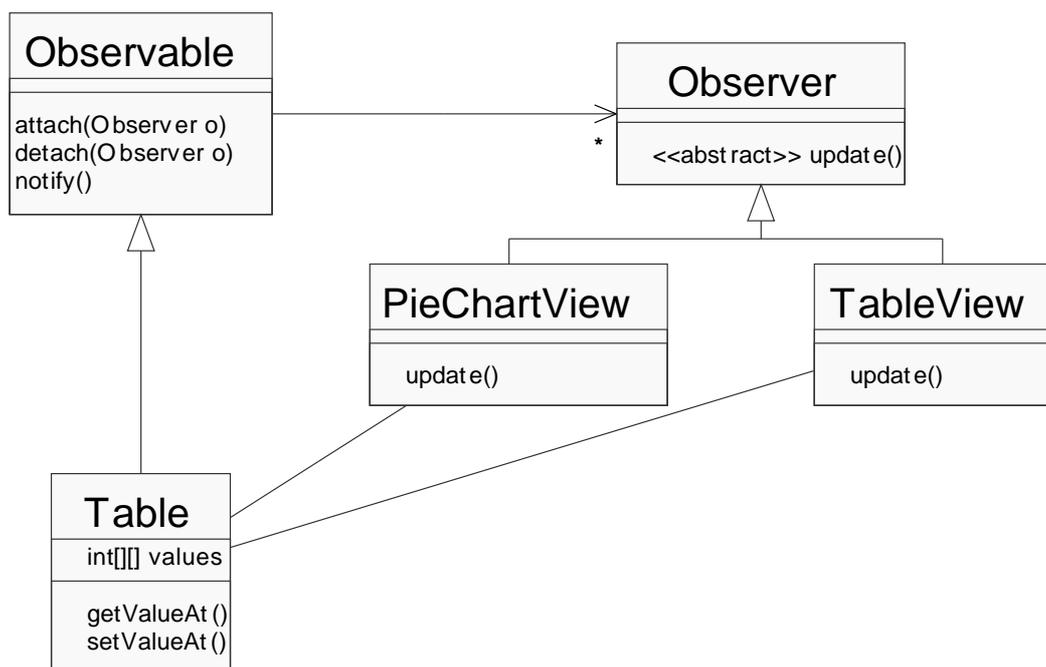
Die Applikation kann nicht vorhersagen, welche konkrete Subklassen einer abstrakten Klasse zur Laufzeit instantiiert werden muß. Sie kennt nur deren Schnittstellen. Das *Factory Method*-Pattern erlaubt es das Wissen darüber, welches konkretes Objekt erzeugt werden muß, an die Subklasse zu delegieren.

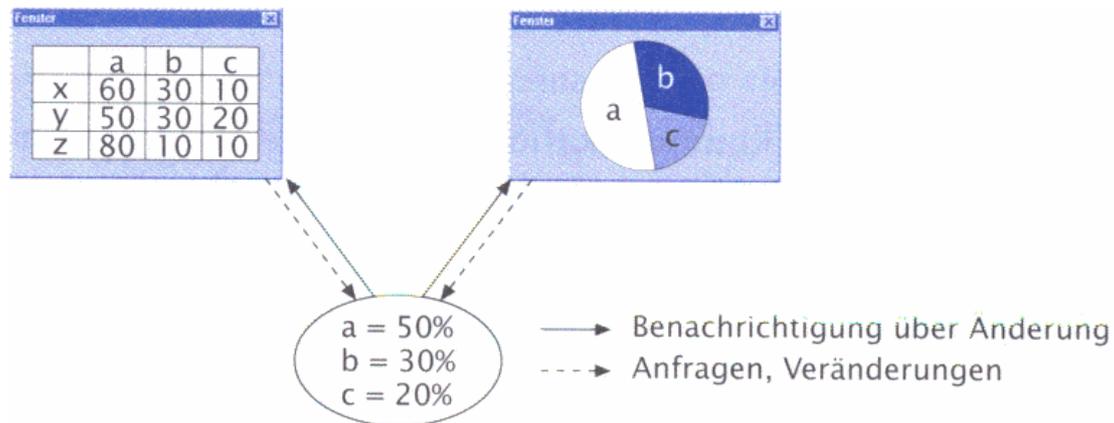


Das *Observer*-Pattern ermöglicht es, bei der Änderung des Zustands eines Objekts alle diejenigen Objekte zu informieren, die davon abhängig sein. *Observer* und *Observable* sind dabei nur lose aneinandergesetzt und können unabhängig voneinander geändert werden.

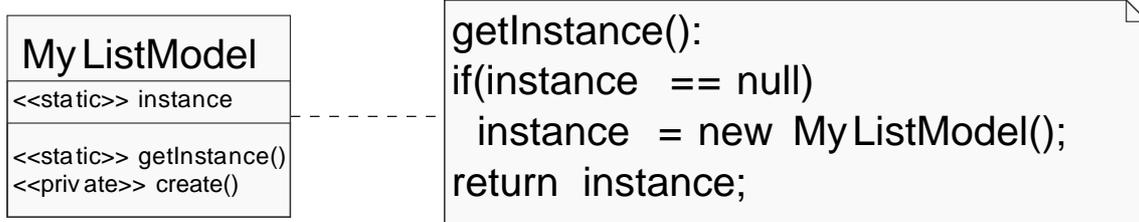


- Kommt beim Swing-API sehr häufig vor
- Vergleich zu Model-View-Controller
- Event notification
- update() kann ggf. parametrisiert sein



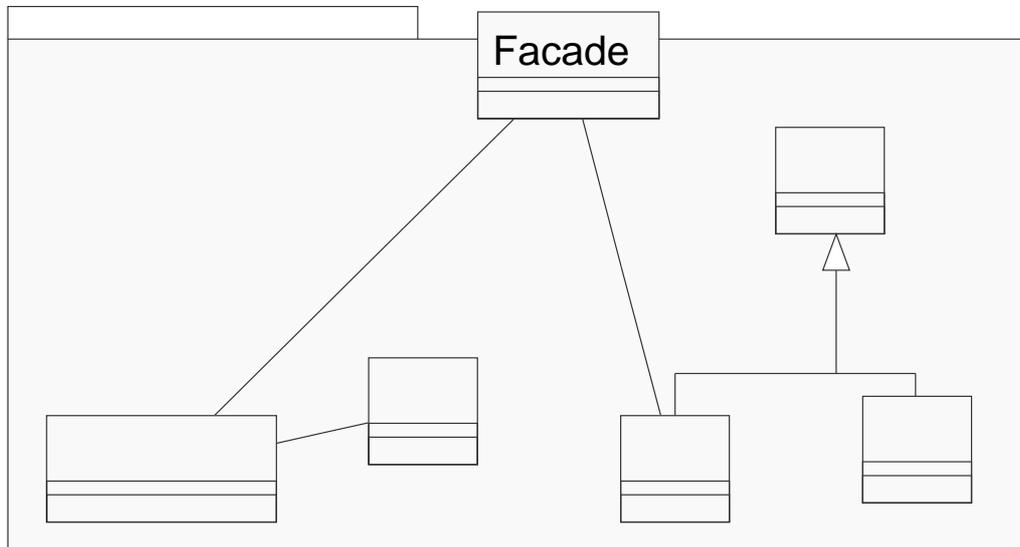


Das *Singleton–Pattern* stellt sicher, daß von einer Klasse nur eine Instanz existiert und ermöglicht den globalen Zugriff auf dieses Objekt. Es stellt somit eine Verbesserung des Konzepts globaler Variablen dar.

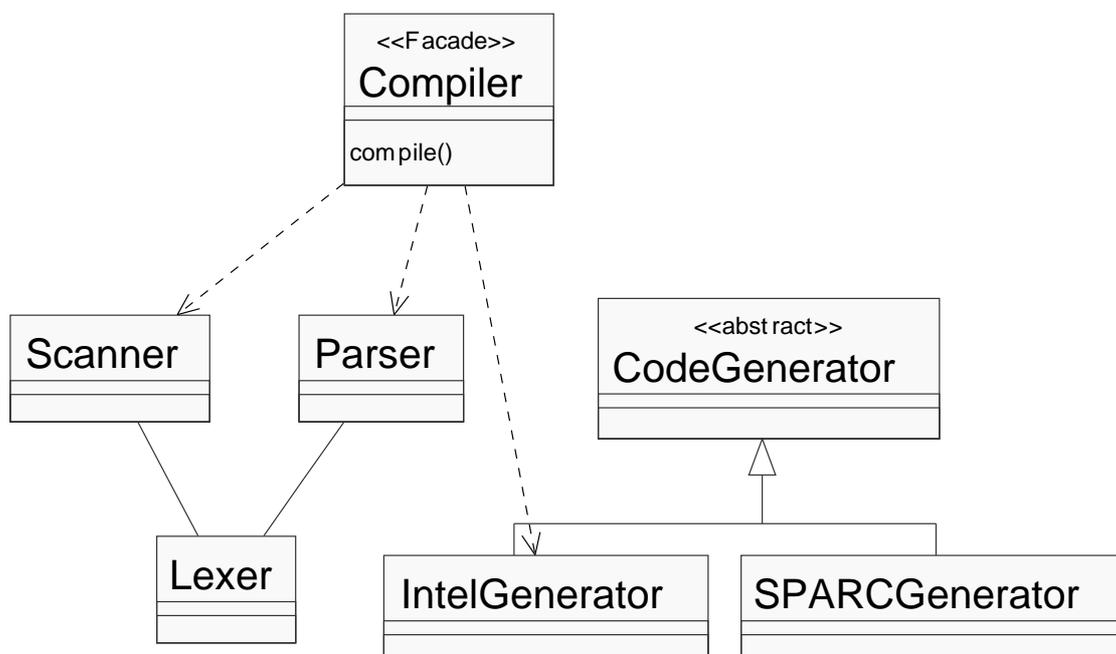


Auch eine spätere Verfeinerung durch Vererbung bleibt möglich, lediglich der Code der Zugriffsmethode muß angepaßt werden. Um zu verhindern, daß weitere Instanzen erzeugt werden, ist der Konstruktor der Klasse häufig `private`, bz. `protected`.

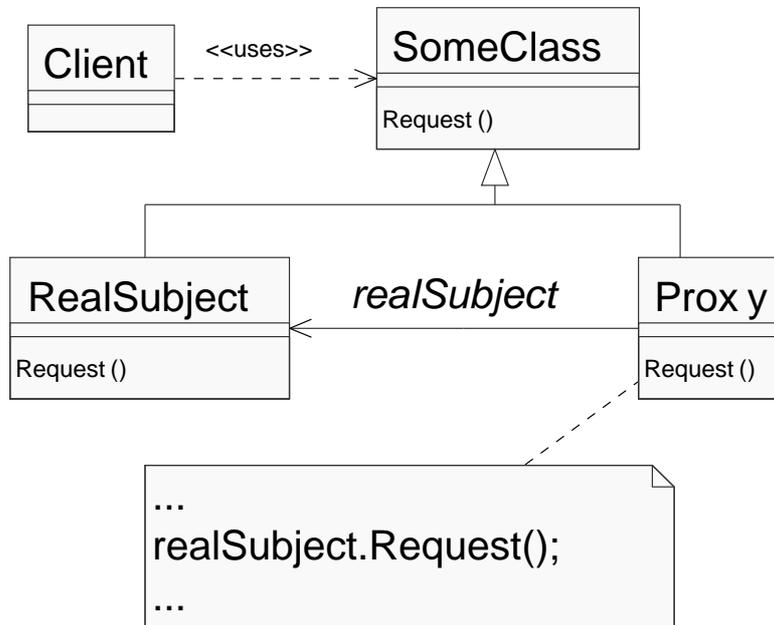
Das *Facade*-Pattern bietet eine einfache zentrale Schnittstelle zu einer Menge von Schnittstellen (Paketen) an. Dies ermöglicht es Pakete, bzw. Schichten nur lose zu koppeln. Klienten, denen die vereinfachte Fassade nicht ausreicht, müssen dahinter blicken.



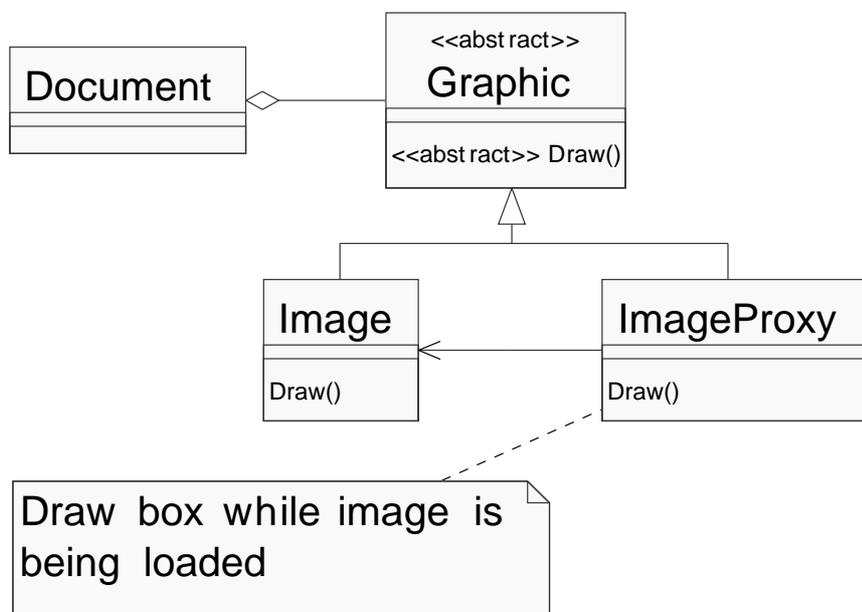
- Erleichtert separate Erstellung von Testtreibern
- Erleichtert Aufteilung der Arbeiten auf mehrere Teams
- Bessere Entkopplung, Aufteilung in Schichten



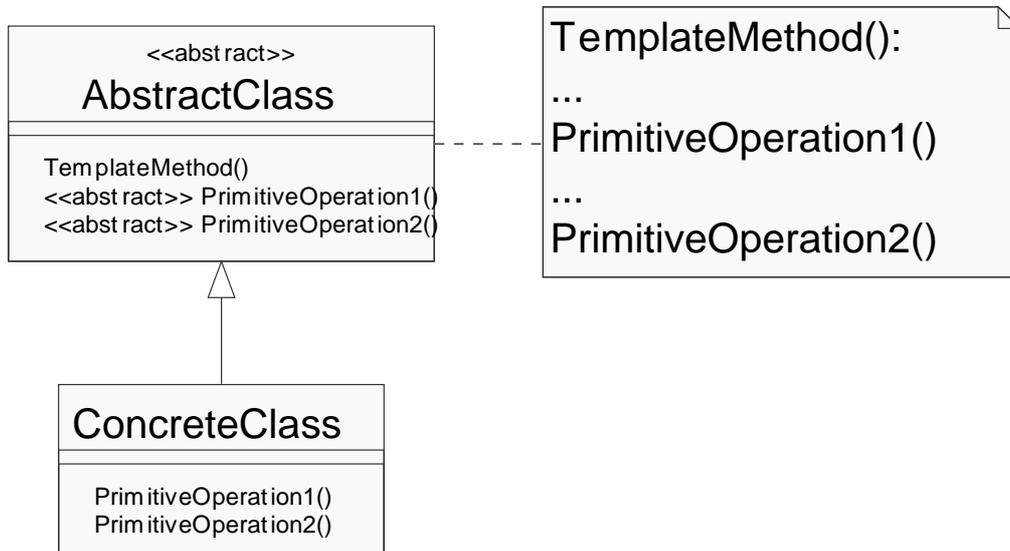
Durch das *Proxy-Pattern* (Proxy=Stellvertreter) ist es möglich den Zugriff auf ein Objekt mit Hilfe eines vorgelagerten Stellvertreter-Objekts zu kontrollieren. Der Stellvertreter kann dabei auch zusätzliche Funktionalität erlauben, z.B. den Zugriff von einem anderen Rechner aus.



- Nützlich für Caching, Erweiterungen, dynamische Programmierung
- Typischerweise für Fernaufrufe (CORBA, RMI)
- Zugriffsschutz auf das eigentliche Objekts



Das *Template-Method-Pattern* realisiert das Hollywood-Prinzip („Don't call us, we call you“)-Prinzip. Es definiert den rahmen eines Algorithmus in einer Operation und delegiert Teilschritte an Unterklassen.



- Kommt oft im Java-API vor, z.B. **MouseListener**, Spezialfall **MouseAdapter** mit vordefinierten leeren Methodenrumpfen
- Auslagerung von varianten Code, bzw. Zusammenfassung von invari-
antem Code

