

могут изменяться во время выполнения и потому более динамичны. Почти все паттерны в какой-то мере используют наследование. Поэтому к категории «паттерны классов» отнесены только те, что сфокусированы лишь на отношениях между классами. Обратите внимание: большинство паттернов действуют на уровне объектов.

Порождающие паттерны классов частично делегируют ответственность за создание объектов своим подклассам, тогда как порождающие паттерны объектов передают ответственность другому объекту. Структурные паттерны классов используют наследование для составления классов, в то время как структурные паттерны объектов описывают способы сборки объектов из частей. Поведенческие паттерны классов используют наследование для описания алгоритмов и потока управления, а поведенческие паттерны объектов описывают, как объекты, принадлежащие некоторой группе, совместно функционируют и выполняют задачу, которая ни одному отдельному объекту не под силу.

Существуют и другие способы классификации паттернов. Некоторые паттерны часто используются вместе. Например, компоновщик применяется с итератором или посетителем. Некоторыми паттернами предлагаются альтернативные решения. Так, прототип нередко можно использовать вместо абстрактной фабрики. Применение части паттернов приводит к схожему дизайну, хотя изначально их назначение различно. Например, структурные диаграммы компоновщика и декоратора похожи.

Классифицировать паттерны можно и по их ссылкам (см. разделы «Родственные паттерны»). На рис. 1.1 такие отношения изображены графически.

Ясно, что организовать паттерны проектирования допустимо многими способами. Оценивая паттерны с разных точек зрения, вы глубже поймете, как они функционируют, как их сравнивать и когда применять тот или другой.

## 1.6. Как решать задачи проектирования с помощью паттернов

Паттерны проектирования позволяют разными способами решать многие задачи, с которыми постоянно сталкиваются проектировщики объектно-ориентированных приложений. Поясним эту мысль примерами.

### Поиск подходящих объектов

Объектно-ориентированные программы состоят из объектов. *Объект* сочетает данные и процедуры для их обработки. Такие процедуры обычно называют *методами* или *операциями*. Объект выполняет операцию, когда получает *запрос* (или *сообщение*) от клиента.

Посылка запроса – это *единственный* способ заставить объект выполнить операцию. А выполнение операции – *единственный* способ изменить внутреннее состояние объекта. Имея в виду два эти ограничения, говорят, что внутреннее состояние объекта *инкапсулировано*: к нему нельзя получить непосредственный доступ, то есть представление объекта закрыто от внешней программы.