в рамку предполагает, что это «что-то» имеется в единственном экземпляре. Мы могли бы приписать некоторую семантику декорации более одного объекта, но тогда пришлось бы вводить множество видов композиций с оформлением: оформление строки, колонки и т.д. Это не дает ничего нового, так как у нас уже есть классы для такого рода композиций. Поэтому для композиции лучше использовать уже существующие классы, а новые добавлять для оформления результата. Отделение декорации от других видов композиции одновременно упрощает классы, реализующие разные элементы оформления, и уменьшает их количество. Кроме того, мы избавлены от необходимости дублировать уже имеющуюся функциональность.

Паттерн декоратор

Паттерн декоратор абстрагирует *отношения между классами и объектами*, необходимые для поддержки оформления с помощью техники прозрачного обрамления. Термин «оформление» на самом деле применяется в более широком смысле, чем мы видели выше. В паттерне декоратор под ним понимается нечто, что возлагает на объект новые обязанности. Можно, например, представить себе оформление абстрактного дерева синтаксического разбора семантическими действиями, конечного автомата – новыми состояниями или сети, состоящей из устойчивых объектов, – тэгами атрибутов. Декоратор обобщает подход, который мы использовали в Lexi, чтобы расширить его область применения.

2.5. Поддержка нескольких стандартов внешнего облика

При проектировании системы приходится сталкиваться с серьезной проблемой: как добиться переносимости между различными программно-аппаратными платформами. Перенос Lexi на другую платформу не должен требовать капитального перепроектирования, иначе не стоит за него и браться. Он должен быть максимально прост.

Одним из препятствий для переноса является разнообразие стандартов внешнего облика, призванных унифицировать работу с приложениями на данной платформе. Эти стандарты определяют, как приложения должны выглядеть и реагировать на действия пользователя. Хотя существующие стандарты не так уж сильно отличаются друг от друга, ни один пользователь не спутает один стандарт с другим – приложения, написанные для Motif, выглядят не совсем так, как аналогичные приложения на других платформах, и наоборот. Программа, работающая более чем на одной платформе, должна всюду соответствовать принятой стилистике пользовательского интерфейса.

Наша проектная цель – сделать так, чтобы Lexi поддерживал разные стандарты внешнего облика и чтобы легко можно было добавить поддержку нового стандарта, как только таковой появится (а это неизбежно произойдет). Хотелось бы также, чтобы наш дизайн решал и другую задачу: изменение внешнего облика Lexi во время выполнения.