Операция `Child` возвращает потомка с заданным индексом (если таковой существует). Глифы типа `Row`, у которых действительно есть потомки, должны пользоваться операцией `Child`, а не обращаться к структуре данных потомка напрямую. В таком случае при изменении структуры данных, скажем, с массива на связанный список не придется модифицировать операции вроде `Draw`, которые обходят всех потомков. Аналогично операция `Parent` предоставляет стандартный интерфейс для доступа к родителю глифа, если таковой имеется. В Lexi глифы хранят ссылку на своего родителя, а `Parent` просто возвращает эту ссылку.

Паттерн компоновщик

Рекурсивная композиция пригодна не только для документов. Мы можем воспользоваться ей для представления любых потенциально сложных иерархических структур. Паттерн компоновщик инкапсулирует сущность рекурсивной композиции объектно-ориентированным способом. Сейчас самое время обратиться к разделу об этом паттерне и изучить его, имея в виду только что рассмотренный сценарий.

2.3. Форматирование

Мы решили, как *представлять* физическую структуру документа. Далее нужно разобраться с тем, как сконструировать *конкретную* физическую структуру, соответствующую правильно отформатированному документу. Представление и форматирование – это разные аспекты проектирования. Описание внутренней структуры не дает возможности добраться до определенной подструктуры. Ответственность за это лежит в основном на Lexi. Редактор разбивает текст на строки, строки – на колонки и т.д., учитывая при этом пожелания пользователя. Так, пользователь может изменить ширину полей, размер отступа и положение точек табуляции, установить одиночный или двойной междустрочный интервал, а также задать много других параметров форматирования.[1] В процессе форматирования это учитывается.

Кстати говоря, мы сузим значение термина «форматирование», понимая под этим лишь разбиение на строки. Будем считать слова «форматирование» и «разбиение на строки взаимозаменяемыми. Обсуждаемая техника в равной мере применима и к разбиению строк на колонки, и к разбиению колонок на страницы.

Таблица 2.2. Базовый интерфейс класса Compositor

Обязанность	Операции
что форматировать	`void SetComposition(Composition*)`
когда форматировать	`virtual void Compose()`

[1] У пользователя найдется что сказать и по поводу *логической* структуры документа: предложений, абзацев, разделов, глав и т.д. *Физическая* структура в общем-то менее интересна. Большинству людей не важно, где в абзаце произошел разрыв строки, если в целом все отформатировано правильно. То же самое относится и к форматированию колонок и страниц. Таким образом, пользователи задают только высокоуровневые ограничения на физическую структуру, а Lexi выполняет их.