Абстрагирование создания объекта

Все, что мы видим и с чем можем взаимодействовать в пользовательском интерфейсе Lexi, – это визуальные глифы, скомпонованные в другие, уже невидимые глифы вроде строки (`Row`) и колонки (`Column`). Невидимые глифы содержат видимые – скажем, кнопку (`Button`) или символ (`Character`) – и правильно располагают их на экране. В руководствах по стилистическому оформлению много говорится о внешнем облике и поведении так называемых «виджетов» (widgets); это просто другое название таких видимых глифов, как кнопки, полосы прокрутки и меню, выполняющих в пользовательском интерфейсе функции элементов управления. Для представления данных виджеты могут пользоваться более простыми глифами: символами, окружностями, прямоугольниками и многоугольниками.

Мы будем предполагать, что имеется два набора классов глифов-виджетов, с помощью которых реализуются стандарты внешнего облика:

- набор абстрактных подклассов класса `Glyph` для каждой категории виджетов. Например, абстрактный класс `ScrollBar` будет дополнять интерфейс глифа с целью получения операций прокрутки общего вида, а `Button` – это абстрактный класс, добавляющий операции с кнопками;
- набор конкретных подклассов для каждого абстрактного подкласса, в которых реализованы стандарты внешнего облика. Так, у `ScrollBar` могут быть подклассы `MotifScrollBar` и `PMScrollBar`, реализующие полосы прокрутки в стиле Motif и Presentation Manager соответственно.

Lexi должен различать глифы-виджеты для разных стилей внешнего оформления. Например, когда необходимо поместить в интерфейс кнопку, редактор должен инстанцировать подкласс класса `Glyph` для нужного стиля кнопки (`MotifButton`, `PMButton`, `MacButton` и т.д.).

Ясно, что в реализации Lexi это нельзя сделать непосредственно, например, вызвав конструктор, если речь идет о языке C++. При этом была бы жестко закодирована кнопка одного конкретного стиля, значит, выбрать нужный стиль во время выполнения оказалось бы невозможно. Кроме того, мы были бы вынуждены отслеживать и изменять каждый такой вызов конструктора при переносе Lexi на другую платформу. А ведь кнопки – это лишь один элемент пользовательского интерфейса Lexi. Загромождение кода вызовами конструкторов для разных классов внешнего облика вызывает существенные неудобства при сопровождении. Стоит что-нибудь пропустить – и в приложении, ориентированном на платформу Mac, появится меню в стиле Motif.

Lexi необходим какой-то способ определить нужный стандарт внешнего облика для создания подходящих виджетов. При этом надо не только постараться избежать явных вызовов конструкторов, но и уметь без труда заменять весь набор виджетов. Этого можно добиться путем *абстрагирования процесса создания объекта*. Далее на примере мы продемонстрируем, что имеется в виду.

Фабрики и изготовленные классы

В Motif для создания экземпляра глифа полосы прокрутки обычно было достаточно написать следующий код на C++: