17
Интерфейс программирования

Окна

Выбор текста

Выбор из меню

Окна просмотра

Выбор из списка

Прокрутка

Определение класса

Тестирование

Инспекторы

Сообщения об ошибках

Окно извещений

Отладчики

Классы, реализующие интерфейс программирования

В этой главе объясняется, как программист может добавлять новые классы в систему, а затем тестировать и отлаживать их поведение в среде программирования Smalltalk-80. Здесь представлен возможный сценарий того, как программист мог бы добавить в систему класс FinancialHistory. FinancialHistory рассматривался в первой части книги как пример создаваемого программистом класса. Его протокол и описание реализации приведены на форзаце в начале книги. Примерный сценарий не претендует на исчерпывающее описание всего интерфейса программирования системы Smalltalk-80. Это только предлог, чтобы дать необходимую мотивировку для описания в последующих главах разнообразных графических средств.

Пользователь и среда программирования Smalltalk-80 взаимодействуют через *растровый экран дисплея*, *клавиатуру* и *указывающее устройство*. Дисплей необходим для предоставления пользователю графической и текстовой информации. Клавиатура нужна для ввода текстовой информации в систему. Указывающее устройство применяется для выбора информации на экране дисплея. Система Smalltalk-80 использует непрямое указывающее устройство, называемое *мышью*. *Курсор* на экране показывает место, куда в данное время указывает мышь. Курсор перемещается по экрану, повторяя передвижение мыши по плоской ровной поверхности. Мышь имеет три кнопки[1], с помощью которых и осуществляется различного рода выбор.

Окна

Экран дисплея содержит одну или более прямоугольных областей, называемых окнами. Окна отображаются на сером фоне и могут накладываться друг на друга. У каждого окна есть заголовок, видимый в верхнем левом углу окна. На рис. 17.1 показан экран системы Smalltalk-80 с двумя перекрывающимися окнами. Они называются SystemBrowser (ОкноПросмотраСистемы) и Workspace (РабочееОкно). Эти два окна содержат только текст, другие окна могли бы содержать рисунки или текст вместе с рисунками.

На рис. 17.1 окно в верхней части экрана — *рабочее*. Оно содержит текст, который может редактироваться или выполняться. Окно, расположенное в нижней части экрана — *окно просмотра системы*. Оно позволяет просматривать и редактировать описания классов в системе. Стрелка справа внизу окна просмотра системы — это курсор. Он показывает текущее расположение мыши. Справа, в самом низу рисунка — маленький прямоугольник, содержащий три небольших, размещенных рядом овала; такой прямоугольник будет на каждом рисунке этой главы. Овалы обозначают три кнопки мыши. Когда нажимается одна из кнопок, соответствующий овал закрашиваться. Мы будем ссылаться на кнопки мыши как на *левую*, *среднюю* и *правую* кнопки, даже если они на некоторых мышах и не расположены в ряд.

Обычно на экране дисплея системы Smalltalk-80 размещается много разнообразной информации. Для того, чтобы выполнить некоторое действие, пользователь должен указать, какая часть видимой информации будет при этом задействована. Всякое действие, направляющее внимание к специфическому фрагменту информации на экране, называется *выбором*. Система обеспечивает визуальную обратную связь, показывая текущий выбор. Наиболее общий механизм обратной связи состоит в инверсии цвета прямоугольной области экрана, в которой черный цвет меняется на белый, а белый на черный. Чтобы начать работать в системе, надо выбрать одно из окон. Выбранное окно отмечается обращением цвета его заголовка и полностью отображается на экране, перекрывая любые другие окна. На рис. 17.1 выбранное окно — окно просмотра системы.

Окно можно выбирать, перемещая курсор в любую не перекрытую другими окнами часть прямоугольной области окна и нажимая в ней левую кнопку мыши. На рис. 17.2 выбирается рабочее окно. Обратите внимание, что для этого нажимается левая кнопка мыши. Когда выбор сделан, рабочее окно перекроет часть окна просмотра системы.

[1] Современные среды программирования Smalltalk-80 работают с любой мышью. — Примеч. ред. перев.

Выбор текста

Текстовый редактор системы Smalltalk-80 предоставляет возможность выбора текста и выполнения операций редактирования над выбранным текстом. Например, чтобы заменить в рабочем окне последовательность символов the standard на my special, надо выбрать старые символы, а затем набрать на клавиатуре новые символы.

Символы выбираются с помощью левой кнопки мыши. Курсор устанавливается в одном из концов выбираемого текста и нажимается левая кнопка мыши (см. рис. 17.3). Выбранный текст еще пуст — не содержит ни одного символа. Позиция пустого выбора показывается кареткой (перевернутым "v"). На рисунке каретка частично перекрыта курсором. Держа нажатой левую кнопку мыши, курсор перемещают к другому концу выбираемого текста. Выбранные символы отмечаются инвертированным прямоугольником (рис. 17.4).

Когда кнопка отпускается, выбор завершается (рис. 17.5). Если теперь набрать на клавиатуре символы, они заменят выбранные. После набора новых символов выбор пуст и каретка установлена за последним новым символом (рис. 17.6).

Выбор из меню

Другой вид выбора, используемый в интерфейсе пользователя, называется *выбором из меню*. Для выбора команды в одном из двух типов меню используются средняя и правая кнопки мыши. Когда нажимается одна из этих кнопок, соответствующее меню появляется на месте курсора. Меню, появляющееся при нажатии средней кнопки, содержит команды, относящиеся к содержимому выбранного окна. Когда окно содержит редактируемый текст (как, например, рабочее окно) команды меню относятся к работе с текстом. Меню, появляющееся при нажатии правой кнопки, содержит команды относящиеся к выбранному окну в целом. Меню средней кнопки может быть различно в разных окнах, а меню правой кнопки всегда одно и то же.

Символы могут быть удалены из фрагмента текста путем их выбора с последующим вызовом команды *cut* из меню средней кнопки. На рис. 17.7 символы special уже выбраны и нажата средняя кнопка мыши. Меню команд, связанных с содержимым окна открыто. Пока кнопка держится нажатой, курсор перемещается к необходимой команде из меню. Выбранная команда выполняется, когда кнопка отпускается. В этом примере удаляется выбранный текст (рис. 17.8).

Выбранный текст может трактоваться и выполняться как выражение языка Smalltalk-80. В меню средней кнопки есть две команды, выполняющие такую операцию: doIt и printIt. Выбор команды doIt просто выполняет выбранное выражение и игнорирует возникающее в результате значение. Выбор команды printIt выполняет выбранное выражение и печатает его значение на экране сразу за текстом выражения. Например, после набора в рабочем окне и последующего выбора выражения Time now, команда printIt выведет на экран новый экземпляр класса Time (рис. 17.9). Напечатанный результат становится текущим выбранным текстом (рис. 17.10).

Если курсор передвинуть за границы прямоугольника меню прежде, чем кнопка будет отпущена, ни одна команда из меню не выполнится.

Окна просмотра

Окно просмотра системы (браузер) — это окно, представляющее классы системы Smalltalk-80. В таком окне можно исследовать и изменять существующие в системе классы. С его помощью можно добавлять новые классы в систему. Окно просмотра состоит из пяти прямоугольных подокон (панелей). В верхней части — четыре панели, отображающие списки. В каждом списке может быть выбран один элемент списка. Выбранный элемент выделяется на экране инверсным изображением. Содержимое списков нельзя редактировать, из них можно только выбирать элементы. Ниже этих четырех панелей расположена панель, показывающая некоторый текст. Эта панель по своим свойствам

подобна рабочему окну и позволяет текст редактировать. Выборы, сделанные в четырех верхних списках определяют текст, который отобразится в нижней панели. Когда выбор сделан во всех четырех верхних списках, нижняя панель показывает метод Smalltalk-80. Этот метод находится в классе, определенном выборами из двух списков слева. Метод внутри выбранного класса определяется выборами из двух списков справа. Окно просмотра на рис. 17.11 показывает метод, выполняемый классом Rectangle в ответ на сообщение center.

Классы в системе собраны в категории. Крайняя левая панель окна просмотра показывает список категорий классов системы. Когда выбирается категория, классы этой категории отображаются в следующей панели справа. В нашем примере выбрана категория Graphics-Primitives (Примитивы-Графики). Эта категория содержит четыре класса. Когда выбирается один из этих классов, в следующей справа панели показывается список категорий сообщений этого класса.

Так как выбран класс Rectangle, отображаются категории сообщений, содержащиеся в его протоколе экземпляра. В нижней части второй панели слева, есть еще две прямоугольных области, помеченные как instance и class. Одна из их должна быть выбрана. Если выбирается class, то следующая панель справа содержит список категорий сообщений класса, если instance — список категорий сообщений экземпляра. Когда выбирается одна из категорий сообщений, в самой правой верхней панели отображается список имен сообщений из этой категории. Когда, наконец, выбирается одно из этих имен сообщений, соответствующий метод отображается в нижней панели окна просмотра. Отображенный метод можно редактировать и, если это необходимо, его старую версию можно заменять на новую, отредактированную.

Выбор из списка

Чтобы сделать выбор из списка, помещают курсор на выбираемый элемент, а затем нажимают и отпускают левую кнопку мыши. На рис. 17.12 в самой правой панели окна просмотра выбирается другой элемент, поэтому и другой метод появляется в нижней панели.

Если левая кнопка мыши нажимается и отпускается в то время, когда курсор расположен на ранее выбранном элементе, то выбор этого элемента отменяется (рис. 17.13).

Когда выбрана категория сообщений, но не выбрано ни одно из имен сообщений, в нижней панели содержится текст, описывающий различные синтаксические части метода (рис. 17.13). Этот текст можно заменить на текст нового метода, который затем можно добавить к системе. Новый метод будет добавлен к выбранной категории.

Если выбрана категория класса, но не выбран ни один из классов, то нижняя панель содержит текст, описывающий различные части определения класса. Этот текст задается в форме сообщения к классу (в данном случае к классу Object), для которого будет создаваться новый подкласс (рис. 17.14).

Прокрутка

Окно может оказаться не столь большим, чтобы показать всю необходимую информацию. Например, многие списки, просматриваемые в системе, достаточно длинны и не могут полностью поместиться на выделенном пространстве экрана. Окно может отобразить любую часть списка с помощью линейки прокрутки. Линейка прокрутки — это прямоугольная область, которая появляется слева от панели, содержащей курсор. Серый прямоугольник внутри линейки прокрутки указывает на то, какая часть всего списка видима в окне. Высота линейки прокрутки представляет длину всего списка, а часть линейки, занятая серым прямоугольником, — видимую часть списка.

Перемещая мышь внутри линейки прокрутки, можно просмотреть любую часть списка. Это процесс называется прокруткой или скроллингом. Когда курсор находится в правой половине линейки, он принимает форму указывающей вверх стрелки. Если нажимается левая кнопка мыши, элементы списка двигаются вверх и становятся видимыми нижние элементы. Когда курсор находится в левой половине линейки, он принимает форму указывающей вниз стрелки; теперь нажатие левой кнопки мыши передвигает список вниз и делает видимыми верхние элементы. Например, можно прокручивать крайний левый

список окна просмотра системы, чтобы увидеть категории, расположенные в списке выше (см. рис. 17.15).

Панели, содержащие текст, также могут прокручиваться, когда они слишком малы для того, чтобы показать весь текст сразу.

Определение класса

Новый класс можно добавить в систему, выбирая категорию класса и редактируя текст, описывающий части определения класса. Например, добавим класс FinancialHistory к категории классов с именем New Projects (Новые Проекты).

Редактирования текста в нижней панели самого по себе недостаточно для определения класса или метода. Чтобы указать, что редактирование завершено и определение класса или метода следует добавить к системе, необходимо вызвать меню средней кнопки текстовой панели и выполнить команду accept, (см. рис. 17.16 и 17.17).

Меню, которое появляется при нажатии средней кнопки мыши, — свое в каждой из панелей окна просмотра системы. В панели, показывающей классы в категории, такое меню включает строку definition. По этой команде выводится определение выбранного класса (см. рис. 17.18 и 17.19).

Определение класса затем можно изменить стандартными операциями редактирования текста, после чего для сохранения изменений необходимо вновь выполнить команду accept из меню средней кнопки. Например, можно было бы добавить новую переменную экземпляра всем прямоугольникам, добавляя имя этой новой переменной в соответствующее место определения класса Rectangle.

Другой пункт меню средней кнопки для списка классов — categories. Когда он выбирается, категории сообщений отображаются в нижней панели окна просмотра системы (рис. 17.20 и 17.21).

Новый класс имеет единственную, пустую категорию сообщений, называемую As yet unclassified (Пока еще неопределенная).

Разбиение на категории может быть изменено путем редактирования текста с последующим выбором команды accept (см. рис. 17.22 и 17.23). Обратите внимание на изменение в третьей панели слева на рис. 17.23. Здесь теперь три категории с именами transaction recording, inquiries и private.

После того, как новый класс добавлен к системе, можно добавлять методы, выбирая категории и редактируя шаблоны методов (рис. 17.24 и 17.25).

Обратите внимание на изменение в самой правой верхней панели на рис. 17.25. Имя сообщения нового метода добавилось в ранее существовавший список методов (в данном случае пустой) и стало текущим выбором.

Тестирование

После того, как все необходимые методы, определенные в главе 5 для класса FinancialHistory, добавлены в систему, можно создать экземпляры этого класса и протестировать их, посылая им сообщения. Сначала надо добавить в систему новую глобальную переменную, посылая сообщение at:put: словарю глобальных переменных с именем Smalltalk. Первый аргумент сообщения at:put: — имя, а второй — начальное значение новой глобальной переменной. Эта глобальная переменная будет использоваться для ссылки на тестируемый экземпляр (см. рис. 17.26).

Сообщения посылаются объекту HouseholdFinances посредством набора нужных выражений в рабочем окне и последующего их выполнения с помощью команды doIt или printIt (рис. 17.27). Несколько выражений можно выбирать и выполнять сразу. Выражения надо отделять друг от друга точкой (рис. 17.28).

Выбор команды printIt вместо команды doIt приводит к отображению результата выполнения выражения на экране непосредственно за текстом выражения (рис. 17.29 и 17.30).

Инспекторы

Инспектор — это окно для просмотра переменных экземпляра объекта. Чтобы вызвать инспектор, надо послать сообщения inspect объекту, переменные экземпляра которого требуется просмотреть (рис. 17.31).

После того, как сообщение inspect послано, система запрашивает у пользователя прямоугольную область, которую займет на экране инспектор. При этом изменится форма курсора, указывая на то, что должен быть задан верхний левый угол прямоугольной области (рис. 17.32).

Курсор перемещается в желаемую точку, а левая кнопка мыши нажимается и удерживается. Форма курсора вновь меняется, подсказывая, что нужно определить теперь правый нижний угол прямоугольной области. Пока левая кнопка мыши остается нажатой, предполагаемая новая прямоугольная область отображается на экране (рис. 17.33).

Когда кнопка отпускается, инспектор занимает выбранную прямоугольную область (рис. 17.34).

Заголовок инспектора — имя класса инспектируемого объекта.

Инспектор имеет две панели. В левой панели показывается список, содержащий self и имена переменных экземпляра объекта. При выборе одного из элементов этого списка, его значение выводится в правой панели (рис. 17.35 и 17.36). Текст, который появляется в правой панели — результат посылки сообщения printString выбранному объекту. Выбор элемента self, расположенного вверху списка, приводит к выводу самого инспектируемого объекта (рис. 17.37).

Сообщения об ошибках

Когда происходит ошибка, процесс, в котором она произошла, останавливается и для него создается окно просмотра. Остановленные процессы могут просматриваться двумя способами: с помощью *окна извещений* и с помощью *отладчика*. Окно извещений содержит простое описание процесса в момент ошибки. Отладчик обеспечивает более детализированный просмотр и, кроме того, предоставляет возможность изменять состояние остановленного процесса перед его возобновлением.

Как пример сообщения об ошибках, мы проследим процесс добавления и отладки нескольких новых методов в классе FinancialHistory. Следующие методы содержат несколько ошибок, которые мы “обнаружим” в процессе тестирования. Цель новых методов — выдавать итоговый отчет о состоянии экземпляра класса FinancialHistory.

```
report
    | reportStream |
    reportStream ← WriteStream on: (String new: 10).
    reportStream cr.
    reportStream nextPutAll: 'Expenses'.
    reportStream cr.
    self expenseReasons do:
       [:reason |
            reportStream tab.
            reportStream nextPutAll: reason.
            reportStream tab.
            reportStream nextPutAll: (self totalSpentFor: reason).
            reportStream cr].
    reportStream nextPutAll: 'Incomes'.
```

```
    reportStream cr.
    self incomeSources do:
      [:source |
          reportStream tab.
          reportStream nextPutAll: source.
          reportStream tab.
          reportStream nextPutAll: (self totalReceivedFrom: source).
          reportStream cr].
    ↑reportStream contents
```

incomeSources

```
    ↑incomes keys
```

expenditureReasons

```
    ↑expenditures keys
```

В класс FinancialHistory добавляется новая категория и новые методы (рис. 17.38).

После добавления новых методов, можно запросить отчет у экземпляра класса FinancialHistory, выполняя соответствующее выражение в рабочем окне (рис. 17.39). Однако, вместо печати отчета, на экране появится окно извещений (рис. 17.40).

Окно извещений

Окно извещений обеспечивает простое визуальное представление процесса, остановленного после ошибки. Его заголовок указывает на природу ошибки. Окно извещений создается посылкой объекту сообщения error:. Аргумент этого сообщения становится заголовком окна извещений. Так, окно извещений, показанное на рис. 17.40, информирует о том, что сообщение expenseReasons посылалось объекту, который его не понял. Список, видимый ниже заголовка, показывает часть состояния остановленного процесса.

Причина происшедшей ошибки очевидна из заголовка окна. В классе FinancialHistory определялось сообщение expenditureReasons, а не expenseReasons. Окно извещений и ошибочный процесс можно сбросить, выбирая команду close в меню правом кнопки мыши (рис. 17.41).

Орфографическая ошибка в методе report может быть исправлена в окне просмотра системы (рис. 17.42 и 17.43). После исправления ошибки, первоначальное выражение можно снова попытаться выполнить в рабочем окне (рис. 17.44 и 17.45).

Однако, возникает другое окно извещений. Причина следующей ошибки не так очевидна. Сообщение do: было послано объекту, который его не понял. Для того, чтобы лучше разобраться в происшедшем, нужно получить более подробную информацию об остановленном процессе, выбирая команду в меню средней кнопки.

Отладчики

Отладчик — это окно, представляющее остановленный процесс, которое описывает происшедшее более подробно, чем окно извещений. Когда из меню средней кнопки мыши окна извещений выбирается команда debug, создается отладчик для описания того же самого процесса (рис. 17.46). После выбора команды debug система запрашивает пользователя о размере прямоугольной области для отладчика. Прямоугольник определяется тем же самым способом, что и для инспектора (рис. 17.47).

Отладчик имеет шесть панелей. Верхняя панель показывает тот же самый список, который был виден в окне извещений. Этот список дает историю процесса, в котором произошла ошибка. Каждая строка соответствует сообщению, которое было послано, но ответ на которое еще не получен. Строка содержит имя класса получателя сообщения и имя сообщения, разделенные символами “>>”. Последняя видимая в списке строка, FinancialHistory>>report, указывает, что экземпляру класса FinancialHistory было послано сообщение report. Это произошло после того, как в рабочем окне выбрали выражение HouseholdFinances report, а затем в меню средней кнопки выбрали команду printIt. Когда

выбирается одна из строк верхней панели отладчика, вызываемый соответствующим сообщением метод отображается в нижней панели.

Когда метод отображается в панели, выделяется последнее посланное перед остановкой процесса сообщение. Рис. 17.48 показывает, что таковым было сообщение do:, которое посылалось результату выражения self expenditureReasons. Следующая вверх по списку строка Set>>do: указывает, что получателем сообщения do: был экземпляр класса Set. Вызываемый метод можно увидеть, выбирая в списке строку Set>>do:.

Рис. 17.49 показывает, что этот метод успел послать сообщение числу 1. Следующая строка вверх по списку, SmallInteger(Number)>>to:do:, информирует о том, что получателем сообщения был экземпляр класса SmallInteger. Когда метод, вызванный сообщением, располагается в суперклассе класса получателя, имя этого суперкласса заключается в круглые скобки и помещается непосредственно после имени класса получателя сообщения. В нашем примере метод to:do: был найден в классе Number.

Верхняя строка списка, SmallInteger(Object)>>doesNotUnderstand:, показывает последнее происшествие перед остановкой процесса — экземпляр класса SmallInteger получил сообщение doesNotUnderstand:. Это сообщение послано системой после того, как сообщение do: не было найдено ни в классе SmallInteger, ни в одном из его суперклассов. Сообщение doesNotUnderstand: вызвало метод, который приостановил процесс и создал окно извещений. Вторая строка сверху из списка, WriteStream(Stream)>>nextPutAll:, указывает, что неправильно истолкованное сообщение do: послано из метода nextPutAll: класса Stream. Рис. 17.50 показывает отладчик с выбранной именно этой строкой. Отображенный ниже метод информирует, что сообщение do: послали объекту с именем aCollection, который был аргументом сообщения nextPutAll:.

Нижние четыре панели отладчика используются для поиска значений переменных, используемых в методе. Они функционируют подобно двум инспекторам. Крайняя левая панель показывает список, состоящий из получателя сообщения (self) и имен его переменных экземпляра. Третья слева панель показывает имена аргументов сообщения и имена временных переменных. Когда в любом из этих списков выбирается имя, значение связанной с ним переменной отображается на панели справа от списка. Чтобы определить получатель сообщения do:, на рис. 17.50 выбран аргумент aCollection.

Источник происшедшей ошибки состоит в том, что экземпляр класса Stream ожидал получить в качестве аргумента сообщения nextPutAll: aCollection набор, а взамен получил число 700. Выбор следующей строки из списка верхней панели окна отладчика показывает, откуда появился этот аргумент. Он был результатом выполнения выражения self totalSpentFor: reason. На рис. 17.51 выбор в нижних панелях отладчика сделан так, чтобы отобразить значения переменной экземпляра expenditures и аргумента reason.

Когда текст выбирается и выполняется в методе, отображаемом в окне просмотра, имена переменных интерпретируются в контексте этого окна. Поэтому аргумент сообщения nextPutAll: может быть найден повторным выполнением выражения (self totalSpentFor: reason) и выдачей результата на экран (рис. 17.52).

Результат, как и ожидалось, равен 700 (см. рис. 17.53).

Метод report предполагал, что в reportStream будет добавлено символьное представление числа 700, но вместо этого туда попало само число. Эта ошибка устраняется посредством посылки этому числу сообщения printString. Исправление можно сделать прямо в отладчике (рис. 17.54). Теперь первоначальное выражение можно выполнить в рабочем окне снова и отчет будет там успешно напечатан (рис. 17.55).

Этим мы завершаем обзор интерфейса программирования системы Smalltalk-80. Способность поддерживать описанный тип взаимодействия с пользователем объясняет природу большинства графических классов, рассматриваемых в последующих главах.

Классы, реализующие интерфейс программирования

Протоколы всех классов, которые реализуют интерфейс программирования, не рассматриваются в этой книге. Они перечислены ниже в алфавитном порядке внутри категорий, представляющих общую или специфическую поддержку реализации описанных в этой главе окон.

Interface Framework and Support
(Структура интерфейса и его поддержка)
- Controller
- ControlManager
- Explainer
- MouseMenuController
- NoController
- StandardSystemController
- StandardSystemView
- View
- WindowingTransformation

Lists and List Selections
(Списки и выбор элементов)
- ListController
- ListView
- LockedListController
- SelectionInListController
- SelectionInListView
- TextList

Scrolling (Прокрутка)
- ScrollController

Confirmers and Prompters
(Подтверждения и подсказки)
- BinaryChoice
- BinaryChoiceController
- CRFillinTheBlankController
- FillinTheBlank
- FillinTheBlankController
- FillinTheBlankView

Browsers (Окна просмотра)
- Browser
- BrowserView
- MethodListBrowser

Menus and Other Selectable Items
(Меню и другие выбираемые элементы)
- ActionMenu
- BinaryChoiceView
- Button
- IndicatorOnSwitchController
- LockedSwitchController
- OneOnSwitch
- Switch
- Switch
- SwitchController
- SwitchView

Text and Code Support
(Поддержка работы с текстом и кодом)
- AlwaysAcceptCodeController
- CodeController
- CodeView
- OnlyWhenSelectedCodeController
- OnlyWhenSelectedCodeView
- ParagraphEditor
- StringHolder
- StringHolderController
- StringHolderView
- TextController
- TextView

Inspectors (Инспекторы)
- ContextInspector
- DictionaryInspector
- Inspector
- InspectorView

Error Reporting-Notifiers and
Debuggers
(Извещение об ошибках и отладка)
- Debugger
- NotifierController
- NotifierView
- ProcessHandle
- SyntaxError

В этой книге также не описывались системные файлы, окна системной информации, окна проектирования и окна управления изменениями.

File Access (Доступ к файлам)
- FileList
- FileModel

Transcript (Окна информации)
- TextCollector
- TextCollectorController
- TextCollectorView

Projects (Проектирование)
- Project
- ProjectController
- ProjectView

Change Management
(Управление изменениями)
- ChangeController
- ChangeList
- ChangeListController
- ChangeListView
- Change
- ChangeScanner

ChangeSet
ClassChange
ClassCommentChange
ClassDefinitionChange
ClassOtherChange
ClassRelatedChange
MethodChange
MethodDifinitionChange
MethodOtherChange
OtherChange