Пример объектно-ориентированной разработки

Для иллюстрации унифицированного процесса рассмотрим фрагмент разработки. Поставим задачу — разработать оконный интерфейс пользователя, который будет использоваться прикладными программами.

Этап НАЧАЛО

Оконный интерфейс пользователя(WUI) — среда, управляемая событиями. Действия в среде инициируются функциями обратного вызова, которые вызываются в ответ на событие — пользовательский ввод. Ядром WUI является цикл обработки событий, который организуется менеджером ввода.

WUI должен обеспечивать следующие типы неперекрывающихся окон:

- простое окно, в которое может быть выведен текст;
- окно меню, в котором пользователь может задать вариант действий — выбор подменю или функции обратного вызова.

Идентификация актеров

Актерами для *WUI* являются:

- пользователь прикладной программы, использующей *WUI;*
- администратор системы, управляющий работой *WUI.*

Внешнее окружение *WUI* имеет вид, представленный на рис. 5.5.

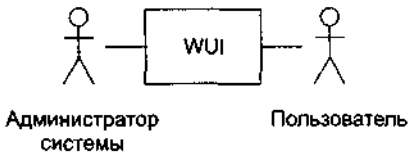


Рис. 5.5. Внешнее окружение WUI

Идентификация элементов Use Case

В WUI могут быть выделены два элемента Use Case:

- управление окнами;
- использование окон.

Диаграмма Use Case для среды WUI представлена на рис. 5.6.

Рис. 5.6. Диаграмма Use Case для среды WUI

Описания элементов Use Case

Описание элемента Use Case Управление окнами.

Действия начинаются администратором системы. Администратор может создать, удалить или модифицировать окно.

Описание элемента Use Case Использование окон.

Действия начинаются пользователем прикладной программы. Обеспечивается возможность работы с меню и простыми окнами.

Этап РАЗВИТИЕ

На этом этапе создаются сценарии для элементов Use Case, разрабатываются диаграммы последовательности (формализующие текстовые представления сценариев), проектируются диаграммы классов и планируется содержание следующего этапа разработки.

Сценарии для элемента Use Case Управление окнами

В элементе Use Case Управление окнами заданы три потока событий — три сценария.

1. Сценарий Создание окна.

Устанавливаются координаты окна на экране, стиль рамки окна. Образ окна сохраняется в памяти. Окно выводится на экран. Если создается окно меню, содержащее обращение к функции обратного вызова, то происходит установка этой функции. В конце менеджер окон добавляет окно в список управляемых окон WUI.

2. Сценарий Изменение стиля рамки.

Указывается символ, с помощью которого будет изображаться рамка. Образ окна сохраняется в памяти. Окно перерисовывается на экране.

3. Сценарий Уничтожение окна.

Менеджер окон получает указание удалить окно. Менеджер окон снимает окно с регистрации (в массиве управляемых окон WUI). Окно снимает отображение с экрана.

Развитие описания элемента Use Case Использование окон

Действия начинаются с ввода пользователем символа. Символ воспринимается менеджером ввода. В зависимости от значения введенного символа выполняется один из следующих вариантов:

при значении ENTER - вариант ОКОНЧАНИЯ ВВОДА;

при переключающем значении - вариант ПЕРЕКЛЮЧЕНИЯ;

при обычном значении - символ выводится в активное окно.

Вариант ОКОНЧАНИЯ ВВОДА:

при активном окне меню выбирается пункт меню. В ответ либо выполняется функция обратного вызова (закрепленная за этим пунктом меню), либо вызывается подменю (соответствующее данному пункту меню);

при активном простом окне выполняется переход на новую строку окна.

Вариант ПЕРЕКЛЮЧЕНИЯ.

При вводе переключающего символа:

ESC - активным становится окно меню;

TAB - активным становится следующее простое окно;

Ctrl-E - все окна закрываются и сеанс работы заканчивается.

Далее из описания элемента Use Case Использование окон выделяются два сценария: Использование простого окна и Использование окна меню.

На следующем шаге сценарии элементов Use Case преобразуются в диаграммы последовательности — за счет этого достигается формализация описаний, требуемая для построения диаграмм классов. Для построения диаграмм последовательности проводится грамматический разбор каждого сценария элемента Use Case: значащие существительные превращаются в объекты, а значащие глаголы — в сообщения, пересылаемые между объектами.

Диаграммы последовательности

Диаграммы изображены на рис. 5.7-5.11.

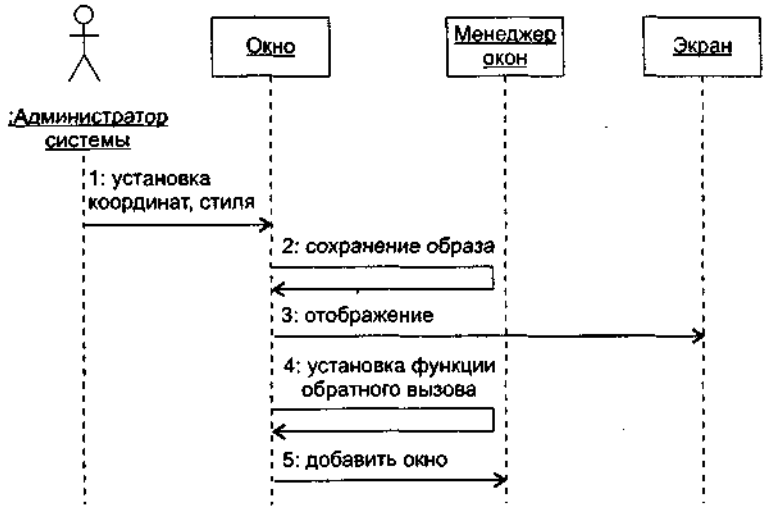


Рис. 5.7. Диаграмма последовательности Создание окна

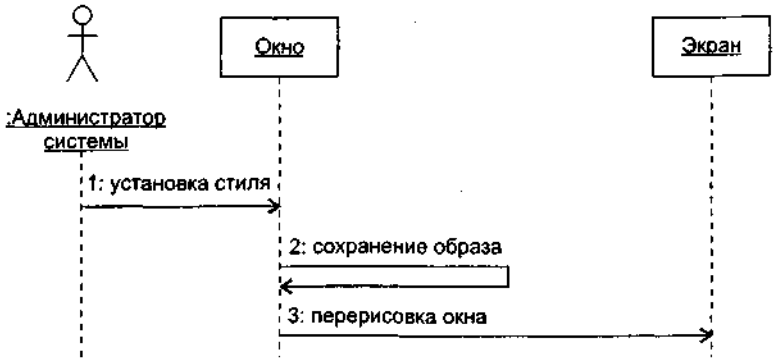
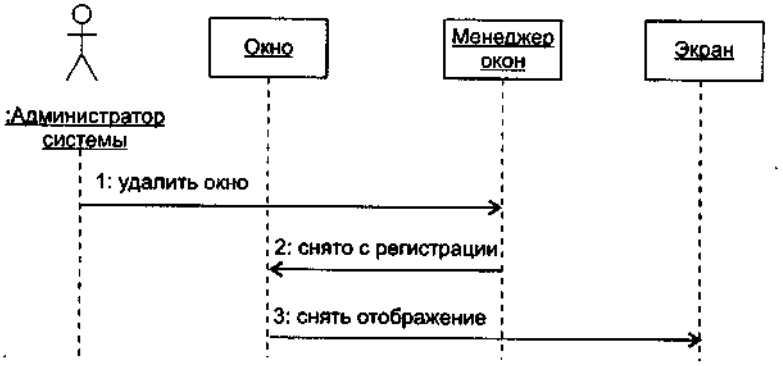


Рис. 5.8. Диаграмма последовательности Изменение стиля рамки



5.9. Диаграмма последовательности Уничтожение окна

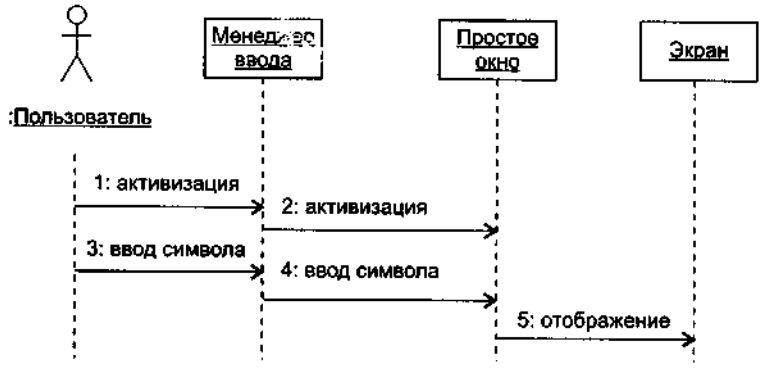


Рис. 5.10. Диаграмма последовательности Использование простого окна

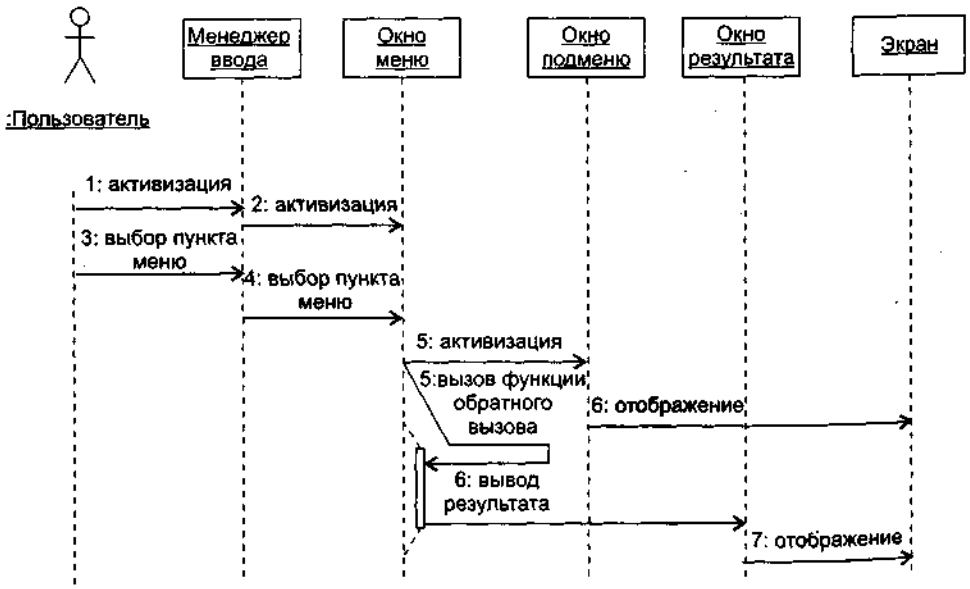


Рис. 5.11. Диаграмма последовательности Использование окна меню

Создание классов

Работа по созданию классов (и включению их в диаграмму классов) требует изучения содержания всех диаграмм последовательности. Проводится она в три этапа.

На первом этапе выявляются и именуются классы. Для этого просматривается каждая диаграмма последовательности. Любой объект в этой диаграмме должен принадлежать конкретному классу, для которого надо придумать имя. Например, резонно предположить, что объекту Менеджер окон должен соответствовать класс Window_Manager, поэтому класс Window_Manager следует ввести в диаграмму. Конечно, если в другой диаграмме последовательности опять появится подобный объект, то дополнительный класс не образуется.

На втором этапе выявляются операции классов. На диаграмме последовательности такая операция соответствует стрелке (и имени) сообщения, указывающей на линию жизни объекта класса. Например, если к линии жизни объекта Менеджер окон подходит стрелка сообщения добавить окно, то в класс Window_Manager нужно ввести си операцию add_to_list().

На третьем этапе определяются отношения ассоциации между классами — они обеспечивают пересылки сообщений между соответствующими объектами.

В нашем примере анализ диаграмм последовательности позволяет выделить следующие классы:

- Window — класс, объектами которого являются простые окна;
- Menu — класс, объектами которого являются окна меню. Этот класс является потомком класса Window;
- Menu_title — класс, объектом которого является окно главного меню. Класс является потомком класса Menu;
- Screen — класс, объектом которого является экран. Этот класс обеспечивает позиционирование курсора, вывод изображения на экран дисплея, очистку экрана;
- Input_Manager — объект этого класса управляет взаимодействием между пользователем и окнами интерфейса. Его обязанности: начальные установки среды WUI, запуск цикла обработки событий, закрытие среды WUI;
- Window_Manager — осуществляет общее управление окнами, отображаемыми на экране. Используется менеджером ввода для получения доступа к конкретному окну.

Для оптимизации ресурсов создается абстрактный суперкласс Root_Window. Он определяет минимальные обязанности, которые должен реализовать любой тип окна (а (посылка символа в окно, перевод окна в активное/пассивное состояние, перерисовка окна, возврат информации об окне). Все остальные классы окон являются его потомками.

Для реализации функций, определенных в сценариях, в классы добавляются свойства и операции. По результатам формирования свойств и операций классов обновляется содержание диаграмм последовательности.

Начальное представление иерархии классов WUI показано на рис. 5.12. Результаты начальной оценки качества проекта сведены в табл. 5.2.

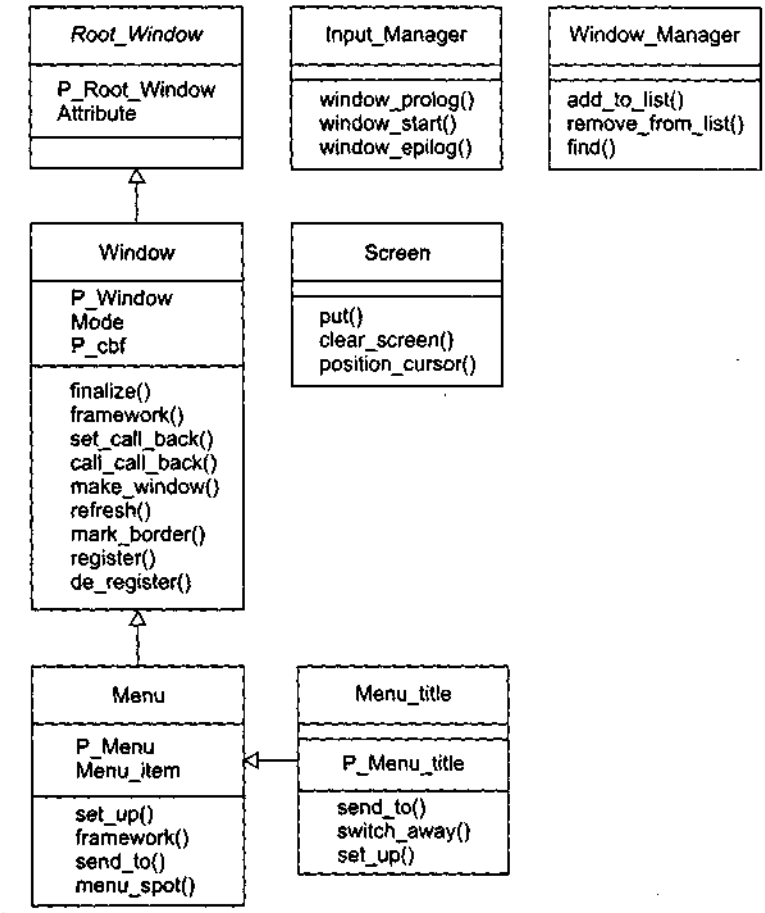


Рис. 5.12. Начальная диаграмма классов WUI

Таблица 5.2. Результаты начальпий оценки качества WUI

трика	**ut_ Manager**	**ndow_ Manager**	**een**	**ot_ Window**	**ndow**	**nu**	**nu_ title**	**еднее значен**
WMC	3	3	3	0	9	4	3	3,57
NOC	-	-	-	1	1	1	0	0,43
Метрики, вычисляемые для системы								
DIT	3							
NC	7							
NOM	25							

Отметим, что для упрощения рисунка на этой диаграмме не показаны существующие между классами отношения ассоциации. В реальной диаграмме они обязательно отображаются — без них экземпляры классов не смогут взаимодействовать друг с другом.

Планирование итераций конструирования

На данном шаге составляется план итераций, который определяет порядок действий на этапе конструирования. Цель каждой итерации — уменьшить риск разработки конечного продукта. Для создания начального плана анализируются элементы Us Case, их сценарии и диаграммы последовательности. Устанавливается приоритет их реализации. При завершении каждой итерации будет повторно вычисляться риск. Оценка риска может привести к необходимости обновления плана итераций.

Положим, что максимальный риск связан с реализацией элемента Use Case Управление окнами, причем наиболее опасна разработка сценария Создание окна, среднюю опасность несет сценарий Уничтожение окна и малую опасность — Изменение стиля рамки.

В связи с этими соображениями начальный план итераций принимает вид:

Итерация 1 — реализация сценариев элемента Use Case Управление окнами:

1. Создание окна.
2. Уничтожение окна.
3. Изменение стиля рамки.

Итерация 2 — реализация сценариев элемента Use Case Использование окон:

4. Использование простого окна.
5. Использование окна меню.

Этап КОНСТРУИРОВАНИЕ

Рассмотрим содержание итераций на этапе конструирования.

Итерация 1 — реализация сценариев элемента Use Case Управление окнами

Для реализации сценария Создание окна программируются следующие операции класса Window:

- framework — создание каркаса окна;
- register — регистрация окна;
- set_call_back — установка функции обратного вызова;
- make_window — задание видимости окна.

Далее реализуются операции общего управления окнами, методы класса Window_Manager:

- add_to_list — добавление нового окна в массив управляемых окон;
- find — поиск окна с заданным переключающим символом.

Программируются операции класса Input-Manager:

- window_prolog — инициализация WUI;
- window_start — запуск цикла обработки событий;
- window_epilog — закрытие WUI.

В ходе реализации перечисленных операций выясняется необходимость и программируется содержание вспомогательных операций.

1. В классе Window_Manager:

- write_to — форматный вывод сообщения в указанное окно;
- hide_win — удаление окна с экрана;
- switchAwayFromTop — подготовка окна к переходу в пассивное состояние;
- switch_to_top — подготовка окна к переходу в активное состояние;
- window_fatal — формирование донесения об ошибке;
- top — переключение окна в активное состояние;
- send_to_top — посылка символа в активное окно.

2. В классе Window:

- put — три реализации для записи в окно символьной, строковой и числовой информации;
- create — создание макета окна (используется операцией framework);
- position — изменение позиции курсора в окне;
- about — возврат информации об окне;
- switch_to — пометка активного окна;
- switch_away — пометка пассивного окна;
- send_to — посылка символа в окно для обработки.

Второй шаг первой итерации ориентирован на реализацию сценария Уничтожение окна. Основная операция — finalize (метод класса Window), она выполняет разрушение окна. Для ее обеспечения создаются вспомогательные операции:

- de_register — удаление окна из массива управляемых окон;

- remove_from_list (метод класса Window_Manager) — вычеркивание окна из регистра.

Для реализации сценария Изменение стиля рамки создаются операции в классе Window:

- mark_border — построение новой рамки окна;
- refresh — перерисовка окна на экране.

В конце итерации создаются операции класса Screen:

- dear_screen — очистка экрана;
- position_cursor — позиционирование курсора;
- put — вывод на экран дисплея строк, символов и чисел.

Результаты оценки качества первой итерации представлены в табл. 5.3.

Таблица 5.3. Оценки качества *WUI* после первой итерации

Метрика	**ut_ Manager**	**ndow_ Manager**	**'een**	**ot_ Window**	**ndow**	**еднее значение**
WMC	0,12	0,42	0,11	0	0,83	0,3
NOC	-	-	-	1	0	0,2
CBO	3	3	0	1	2	1,8
RFC	6	11	0	0	23	8
LCOM	3	0	5	0	0	1,6
CS	3/2	10/8	5/1	0/2	18/22	7,2/7
NOO	-	-	-	0	0	0
NOA	-	-	-	0	18	3,6
SI	-	-	-	0	0	0
OS_{AVG}	4	4,2	2,2	0	4,6	3
NP_{AVG}	0	1,3	1	0	2,4	0,9
Метрики, вычисляемые для системы						
DIT	1					
NC	5					
MOM	35					
LOC_{Σ}	148					

Итерация 2 — реализация сценариев элемента Use Case Использование окон

На этой итерации реализуем методы классов Menu и Menu_title, а также добавим необходимые вспомогательные методы в класс Window.

Отметим, что операции, обеспечивающие сценарий Использование простого окна, в основном уже реализованы (на первой итерации). Осталось запрограммировать следующие операции — методы класса Window:

- call_call_back — вызов функции обратного вызова;
- initialize — управляемая инициализация окна;
- clear — очистка окна с помощью пробелов;
- new_line — перемещение на следующую строку окна.

Для обеспечения сценария Использование окна меню создаются следующие операции.

1. В классе Menu:

- framework — создание каркаса окна-меню;
- send_to — обработка пользовательского ввода в окно-меню;
- menu_spot — выделение выбранного элемента меню;
- set_up — заполнение окна-меню именами элементов;
- get_menu_name — возврат имени выбранного элемента меню;
- get_cur_selected_detaits — возврат указателя на выбранное окно и функцию обратного вызова.

2. В классе Menu_title:

- send_to — выделение новой строки меню или вызов функции обратного вызова;
- switch_away — возврат в базовое окно-меню более высокого уровня;

- set_up — установки окна меню-заголовка.

Результаты оценки качества второй итерации представлены в табл. 5.4.

Таблица 5.4. Оценки качества WUI после второй итерации

трика	**ut_ Manager**	**ndow_ Manager**	**een**	**ot_ Window**	**ndow**	**nu**	**nu title**	**еднее значен**
WMC	0,12	0,42	0,11	0	0,98	0,33	0,27	0,32
NOC	-	-	-	1	1	1	0	0,4
CBO	3	3	0	1	2	2	3	2
RFC	6	11	0	0	27	9	12	9,4
LCOM	3	0	5	0	0	0	0	1,1
CS	3/2	10/8	5/1	0/2	22/22	28/24	11/12	11,3
NOO	-	-	-	0	0	2	3	0,7
NOA	-	-	-	0	22	6	0	4
SI	-	-	-	0	0	0,23	0,46	0,1
os_{we}	4	4,2	2,2	0	4,45	4,13	9	4,0
NP_{AVG}	0	1,3	1	0	2,18	4,63	1,67	1,5
Метрики, вычисляемые для системы								
DIT	3							
NC	7							
MOM	48							
LOC_Z	223							

Сравним оценки качества первой и второй итераций.

1. Рост системных оценок $LOC_{\sum}$, NOM, а также средних значений метрик WMC, RFC, CS, CBO и NOO — свидетельство возрастания сложности продукта.
2. Увеличение значения DIT и среднего значения NOC говорит об увеличении возможности многократного использования классов.
3. На второй итерации в среднем была ослаблена абстракция классов, о чем свидетельствует увеличение средних значений NOC, NOA, SI.
4. Рост средних значений OS_{AVG} и NP_{AVG} говорит о том, что сотрудничество между объектами усложнилось.
5. Среднее значение CBO указывает на увеличение сцепления между классами (это нежелательно), зато снижение среднего значения LCOM свидетельствует, что связность внутри классов увеличилась (таким образом, снизилась вероятность ошибок в ходе разработки).

Вывод: качество разработки в среднем возросло, так как, несмотря на увеличение средних значений сложности и сцепления (за счет добавления в иерархию наследования новых классов), связность внутри классов была увеличена.

В практике проектирования достаточно типичны случаи, когда в процессе разработки меняются исходные требования или появляются дополнительные требования к продукту. Предположим, что в конце второй итерации появилось дополнительное требование — ввести в WUI новый тип окна — диалоговое окно. Диалоговое окно должно обеспечивать не только вывод, но и ввод данных, а также их обработку.

Для реализации этого требования вводится третья итерация конструирования.

Итерация 3 — разработка диалогового окна

Шаг 1: Спецификация представления диалогового окна.

На этом шаге фиксируется представление заказчика об обязанностях диалогового окна. Положим, что оно имеет следующий вид:

1. Диалоговое окно накапливает посылаемые в него символы, отображая их по мере получения.
2. При получении символа конца сообщения (ENTER) полная строка текста принимается в функцию обратного вызова, связанную с диалоговым окном.

3. Функция обратного вызова реализует обслуживание, требуемое пользователю.
4. Функция обратного вызова обеспечивается прикладным программистом.

Шаг 2: Модификация диаграммы Use Case для WUI.

Очевидно, что дополнительное требование приводит к появлению дополнительного элемента Use Case, который находится в отношении «расширяет» с базовым г элементом Use Case Использование окон.

Диаграмма Use Case принимает вид, представленный на рис. 5.13.

Рис. 5.13. Модифицированная диаграмма Use Case для WUI

Шаг 3: Описание элемента Use Case Использование диалогового окна.

Действия начинаются с ввода пользователем переключающего символа, активизирующего данный тип окна. Символ воспринимается менеджером ввода. Далее пользователь вводит данные, которые по мере поступления отображаются в диалоговом окне. После нажатия пользователем символа окончания ввода (ENTER) данные передаются в функцию обратного вызова как параметр. Выполняется функция обратного вызова, результат выводится в простое окно результата.

Шаг 4: Диаграмма последовательности Использование диалогового окна.

Диаграмма последовательности для сценария Использование диалогового окна показана на рис. 5.14.

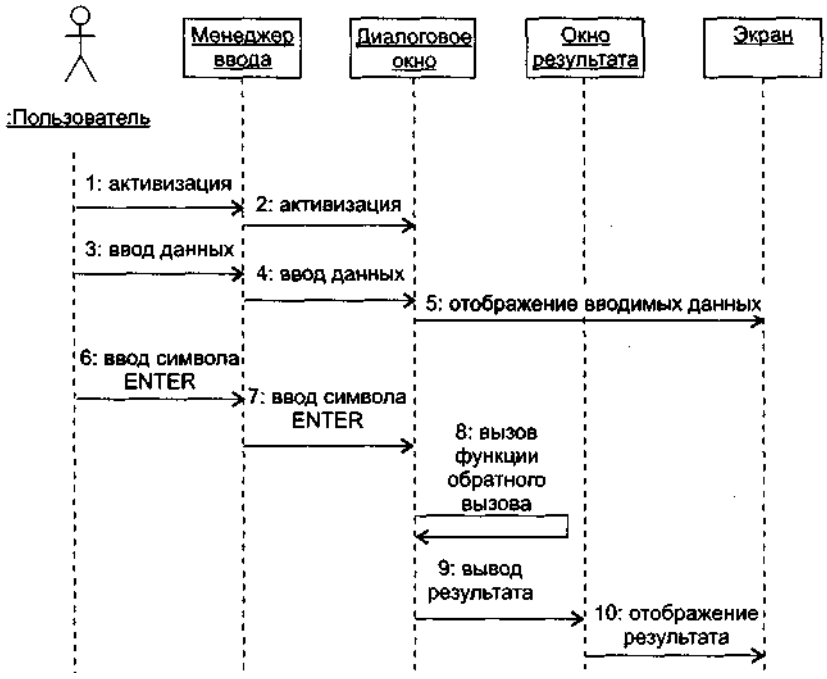


Рис. 5.14. Диаграмма последовательности Использование диалогового окна

Шаг 5: Создание класса.

Для реализации сценария Использование диалогового окна создается новый класс Dialog, который является наследником класса Window. Объекты класса Dialog образуют диалоговые окна.

Класс Dialog переопределяет следующие операции, унаследованные от класса Window:

- framework — формирование диалогового окна. Параметры операции: имя диалогового окна,

координаты, ширина окна, заголовок окна и ссылка на функцию обратного вызова. Операция создает каркас окна, устанавливает для него функцию обратного вызова, делает окно видимым и регистрирует его в массиве управляемых окон;

- send_to — обрабатывает пользовательский ввод, посылаемый в диалоговое окно. Окно запоминает символы, вводимые пользователем, а после нажатия пользователем клавиши ENTER вызывает функцию обратного вызова, обрабатывающую эти данные.

Конечное представление иерархии классов WUI показано на рис. 5.15. Результаты оценки качества проекта (в конце третьей итерации) сведены в табл. 5.5. Динамика изменения значений для метрик класса показана в табл. 5.6.

Таблица 5.5. Оценки качества WUI после третьей итерации

трика	**ut_ Manager**	**ndow Manager**	**een**	**ot Window**	**ndow**	**nu**	**nu-title**	**log**	**еднее значе**
WMC	0,12	0,42	0,11	0	0,98	0,33	0,27	0,23	0,3
NOC	-	-	-	1	2	1	0	0	0,5
CBO	3	3	0	1	2	2	3	2	2
RFC	6	11	0	0	27	9	12	7	9,1
LCOM	3	0	5	0	0	0	0	0	1
CS	3/2	10/8	5/1	0/2	22/22	28/24	11/12	24/14	2/10,6
NOO	-	-	-	0	0	2	3	2	0,9
NOA	-	-	-	0	22	6	0	0	3,5
SI	-	-	-	0	0	0,23	0,46	0,27	0,1
OS_{AVG}	4	4,2	2,2	0	4,45	4,13	9	11,5	4,9
NP_{AVG}	0	1,3	1	0	2,18	4,63	1,67	4	1,8
Метрики, вычисляемые для системы									
DIT	3								
NC	8								
NOM	50								
LOC_{Σ}	246								

Таблица 5.6. Средние значения метрик класса на разных итерациях

Метрика	**Итерация 1**	**Итерация 2**	**Итерация 3**
WMC	0,3	0,32	0,31
NOC	0,2	0,4	0,5
CBO	1,8	2	2
RFC	8	9,4	9,1
LCOM	1,6	1,1	1
CS	7,2/7	11,3/10,1	12,2/10,6
NOO	0	0,7	0,9
NOA	3,6	4	3,5
SI	0	0,1	0,14
OS_{AVG}	3	4,0	4,9
NP_{AVG}	0,9	1,5	1,8
DIT	1	3	3
NC	5	7	8
NOM	35	48	50
LOC_{Σ}	148	223	246

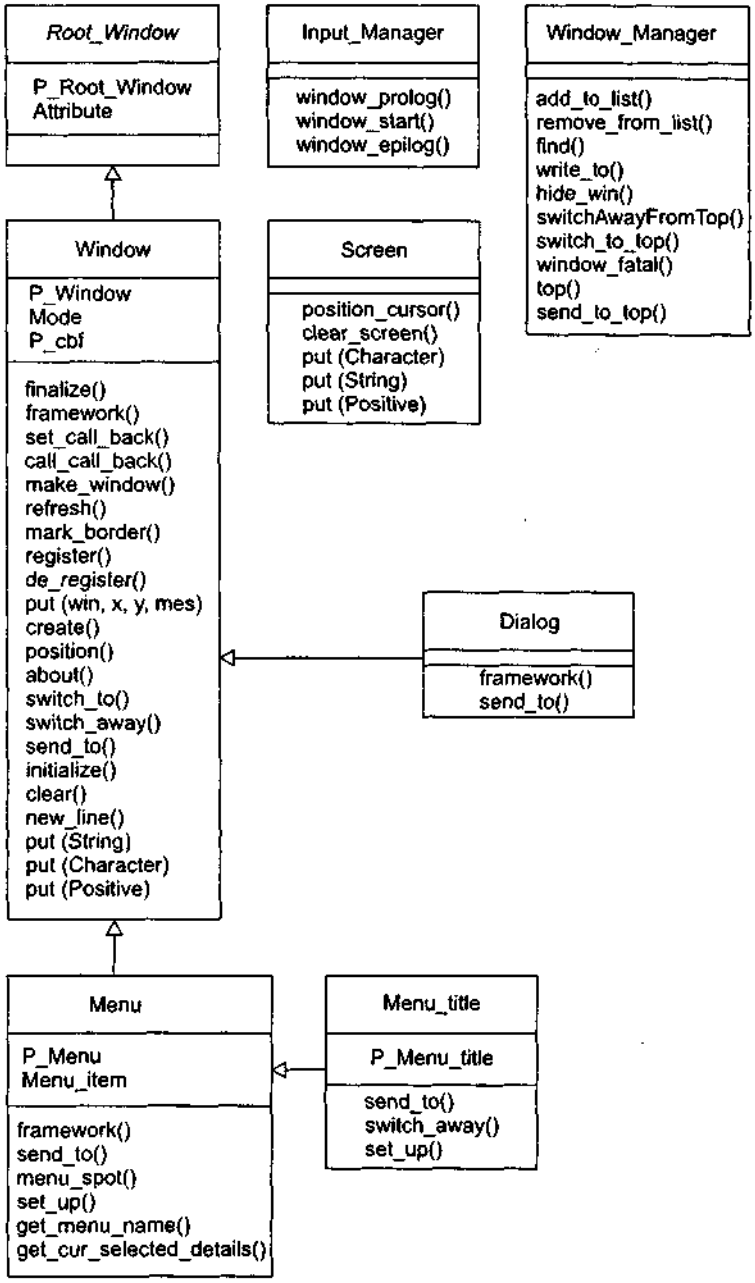


Рис. 5.15. Конечная диаграмма классов WUI

Сравним средние значения метрик второй и третьей итераций:

1. Общая сложность WUI возросла (увеличились значения LOC_{Σ}, NOM и NC), однако повысилось качество классов (уменьшились средние значения WMC и RFC).
2. Увеличились возможности многократного использования классов (о чем свидетельствует рост среднего значения NOC и уменьшение среднего значения WMC).
3. Возросла средняя связность класса (уменьшилось среднее значение метрики LCOM).
4. Уменьшилось среднее значение сцепления класса (сохранилось среднее значение CBO и уменьшилось среднее значение RFC).

Вывод: качество проекта стало выше.

На последней итерации рассчитаны значения интегральных метрик Абреу, они представлены в табл. 5.7. Эти данные также характеризуют качество проекта и подтверждают наши выводы.

Таблица 5.7. Значения метрик Абреу для WUI

Метрика	**Значение**
MHF	0,49
AHF	0,49

MIF	0,49
AIF	0,29
POF	0,69
COF	0,25

Метрические данные проекта помещают в метрический базис фирмы-разработчика, тем самым обеспечивается возможность их использования в последующих разработках.