Лабораторная работа № 3.3

Генерация кода C++ на основе модели UML

Цель работы: Овладение технологией генерации кода на языке C++ по UML диаграммам классов и компонентов средствами *Rational Rose.*

Соответствующий генератор кода не включается по умолчанию – следует выбрать элемент меню **Add-lns → Add In Manager**, в одноименном диалоговом окне установить флажок **Rose C++** и закрыть окно щелчком на кнопке **ОК**.

Ниже раскрыты фазы процессов генерации кода.

1. Создание наборов свойств

При генерации кода учитываются свойства проекта в целом, а также свойства уровней классов, ролей, атрибутов и операций. К свойствам, регламентирующим характеристики проекта как такового, относятся имя файла проекта, названия контейнерных классов, используемых по умолчанию, и местоположение генерируемого кода. Свойства уровня класса обусловливают необходимость и способы создания конструкторов, деструкторов, конструкторов копии, операторов сравнения и методов **get/set**. Набор свойств роли определяет потребность в использовании методов **get/set**, признаки видимости методов и варианты применения того или иного контейнерного класса. Свойства разновидностей операций (***common***, ***virtual***, ***abstract***, ***static***, ***friend***) и придать статус "постоянной" (*constant*). *Rational Rose* предоставляет возможность создания любого количества наборов свойств, отвечающих существу проекта, и их редактирования. Для каждого класса генерируются два файла — файл заголовка (.h) и файл спецификации (.cpp). При работе над типичным проектом обязанности по формированию наборов свойств генерируемого кода распределяются между несколькими сотрудниками, а результаты используются всеми участниками группы. Вот некоторые примеры часто создаваемых наборов свойств: "виртуальный деструктор", "виртуальная операция", "абстрактная операция", "статическая операция".

Порядок выполнения

1. Выбрать элемент меню **Tools→Options**.
2. Перейти на вкладку **C++** диалогового окна **Options**.
3. В раскрывающемся списке Type установить требуемый тип набора свойств.
4. Щелкнуть на кнопке **Clone**, чтобы открыть диалоговое окно **Clone Property Set**.
5. Ввести наименование нового набора свойств и закрыть окно щелчком на кнопке **ОК**.

6. В списке **Model Properties** выбрать свойство, подлежащее модификации, и щелкнуть в пределах столбца **Value**.

7. Ввести новое значение свойства либо выбрать таковое с помощью раскрывающегося списка.

8. Повторить действия, перечисленные в п. п. 6, 7 для каждого свойства, которое должно быть изменено.

9. Щелкнуть на кнопке **Apply**, чтобы сохранить информацию.

10. Повторить действия, перечисленные в п.п. 3-9, с целью создания остальных наборов свойств.

11. Закрыть диалоговое окно **Options** щелчком на кнопке **ОК**.

Процесс создания набора свойств «Виртуальный деструктор» показан на рис. 1

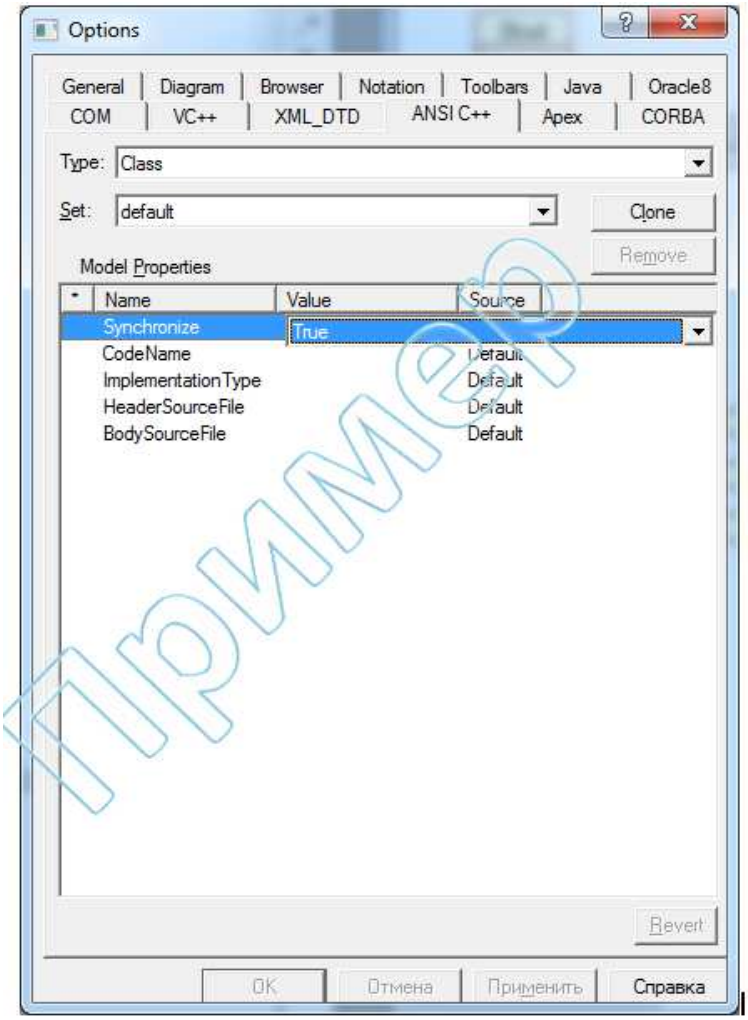

Рисунок 1. Создание набора свойств

2. Определение спецификаций компонентов

Rational Rose генерирует код, принимая во внимание номенклатуру созданных компонентов в совокупности с их стереотипами. Для каждого компонента без стереотипа система генерирует файл **.h**, содержащий

информацию объявления и определения соответствующего класса. Если компонент снабжен стереотипом **Package Specification**, генерируется файл **.h** с объявлением класса. Если же при этом существует надлежащий компонент со стереотипом **Package Body**, генерируется и файл **.cpp** с определением класса.

Порядок выполнения

1. Двойным щелчком на элементе дерева в окне **Browser**, представляющем диаграмму компонентов, открыть окно диаграммы.
2. Расположить курсор мыши над элементом диаграммы, отвечающим требуемому компоненту, и щелкнуть правой кнопкой, чтобы активизировать контекстное меню.
3. Выбрать элемент меню **Open Specification**.
4. Перейти на вкладку **General** диалогового окна **Component Specification**.
5. В поле **Stereotype** ввести значение стереотипа либо выбрать таковое с помощью раскрывающегося списка.
6. Закрыть диалоговое окно щелчком на кнопке **OK**.

Диалоговое окно **Component Specification** изображено на рис.2.

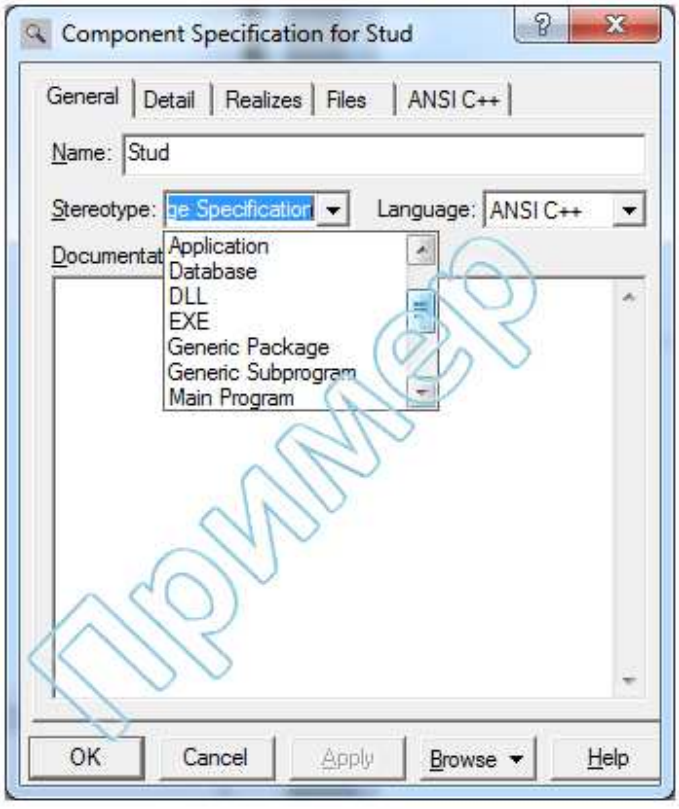

Рисунок 2. Спецификация компонента

Создание заголовка и тела компонента

1. Двойным щелчком на элементе дерева в окне **Browser**, представляющем диаграмму компонентов, открыть окно диаграммы.

2. Расположить курсор мыши над элементом диаграммы, отвечающим требуемому компоненту, и щелкнуть правой кнопкой, чтобы активизировать контекстное меню.

3. Выбрать элемент меню **Open Specification**.

4. Перейти на вкладку **General** диалогового окна **Component Specification**.

5. В раскрывающемся списке **Stereotype** выбрать значение стереотипа **Package Specification** для файла заголовка компонента либо значение **Package Body** — для файла, содержащего тело кода компонента.

6. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Пример диаграммы компонентов, элементы которой отвечают файлам **.h** и **.cpp**, приведен на рис. 3. Светлый компонент соответствует файлу заголовка, темный - файлу тела кода.

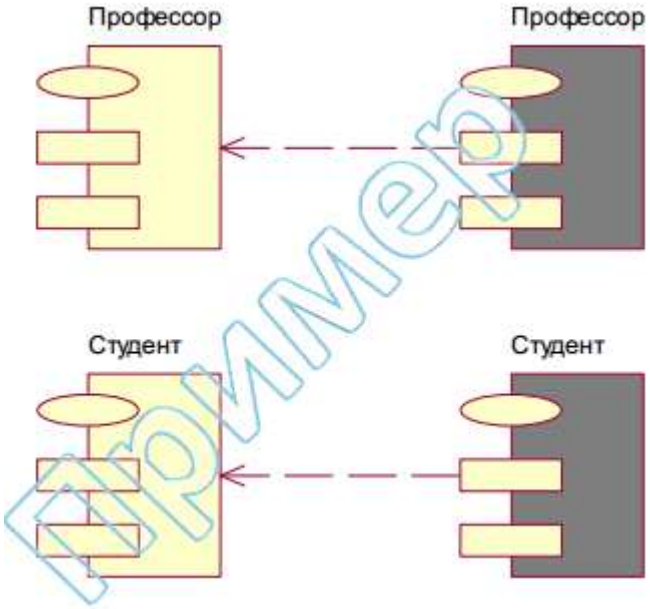

Рисунок 3. Уточненная диаграмма компонентов

3. Выбор языка программирования для компонентов

Как только компоненты, представляющие файлы **.h** и **.cpp**, созданы, им должен быть поставлен в соответствие язык программирования (C++). (Если языком, предлагаемым по умолчанию, является C++ - обратитесь к раскрывающемуся списку Default Language на вкладке Notation диалогового окна Options, активизируемого командой меню **Tools→Options**, - система автоматически выбирает опцию C++ для каждого компонента модели.)

1. Щелчком правой кнопки мыши указать компонент в дереве окна **Browser** либо на диаграмме компонентов и активизировать контекстное меню.

2. Выбрать элемент меню **Open Specification**.

3. Перейти на вкладку **General** диалогового окна **Component Specification**.

4. В раскрывающемся списке **Language** выбрать требуемую опцию (в данном случае – **C++).**

5. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Окно спецификации компонента "курс" показано на рис. 4.

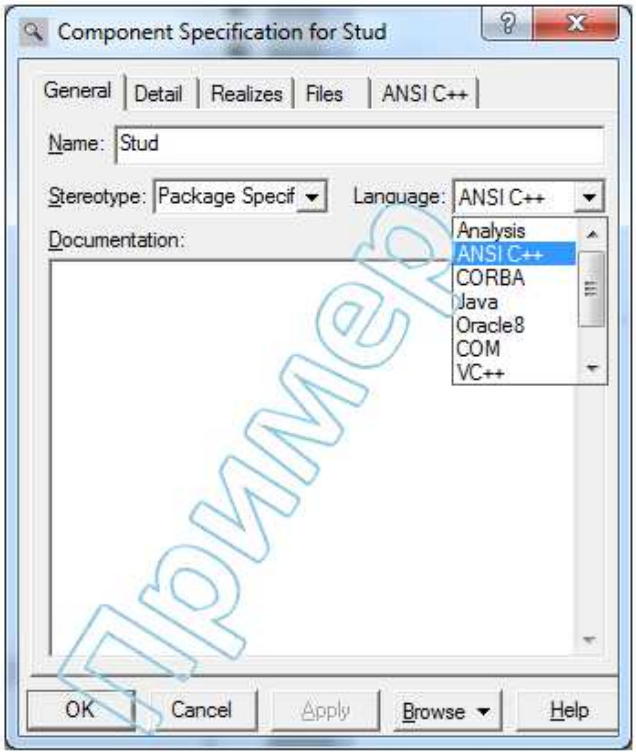

Рисунок 4. Выбор языка программирования для компонента

4. Отнесение классов к компонентам

После создания компонентов, представляющих файлы .h, с ними следует сопоставить те или иные классы модели.

1. Двойным щелчком на элементе дерева в окне Browser, представляющем диаграмму с компонентами файлов .h и .cpp, открыть окно диаграммы.

2. Выбрать класс в дереве окна Browser, перетащить в окно диаграммы и опустить на компонент, соответствующий требуемому файлу .h.

5. Связывание наборов свойств с элементами модели

Каждый элемент модели (например, класс, атрибут или роль) анализируется системой с целью выявления свойств, которыми должен обладать генерируемый код. Если элемент должен обладать свойствами,

отличными от тех, которые предусмотрены в наборе, предлагаемом по умолчанию, с элементом связывается тот или иной созданный набор свойств.

Щелчком правой кнопки мыши указать элемент модели в дереве окна **Browser** либо на диаграмме и активизировать контекстное меню.

1. Выбрать элемент меню **Open Specification**.
2. Перейти на вкладку **C++** диалогового окна спецификации элемента.
3. В раскрывающемся списке **Set** выбрать требуемый набор свойств.
4. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Поскольку адекватного набора свойств, подходящего для каждой комбинации элементов, заведомо не существует, при рассмотрении определенного элемента то или иное свойство можно переопределить — даже в том случае, если свойство входит в набор свойств, предлагаемый по умолчанию.

Переопределение свойства элемента модели

1. Щелчком правой кнопки мыши указать элемент модели в дереве окна **Browser** либо на диаграмме и активизировать контекстное меню.
2. Выбрать элемент меню **Open Specification**.
3. Перейти на вкладку **C++** диалогового окна спецификации элемента
4. В раскрывающемся списке Set выбрать требуемый набор свойств.
5. В списке **Model Properties** указать свойство, подлежащее модификации, и щелкнуть в пределах столбца **Value**.
6. Ввести новое значение свойства либо выбрать таковое с помощью раскрывающегося списка.
7. Повторить действия, перечисленные в п.п. 5, *6,* для каждого свойства, которое должно быть изменено.
8. Щелкнуть на кнопке **Override**.
9. Закрыть диалоговое окно щелчком на кнопке **ОК**.

6. Генерация кода

Код может быть сгенерирован для пакета в целом, для отдельного компонента либо группы компонентов. В качестве имени файла, в который помещается код, выбирается наименование пакета или компонента. Файл располагается в структуре каталогов, соответствующей поддереву Component View дерева Browser.

1. Щелчком выбрать пакет, компонент или группу компонентов в дереве окна **Browser** либо на диаграмме.
2. Выбрать элемент меню **Tools →C++ →Code Generation**.

3. Система осуществит генерацию кода и воспроизведет информацию о результатах в диалоговом окне **Code Generation Status**.

7. Анализ ошибок

Предупреждающие сообщения и информация об ошибках выводятся в окно протокола (**Log Window**, или **Output Window**). (Чтобы открыть окно протокола, достаточно выбрать элемент меню **View→Log**.) Если дизайн класса не завершен, система отобразит в окне предупреждающее сообщение. Подобная ситуация возникает в процессе итеративной разработки, когда классы не всегда реализуются в пределах одной отдельно взятой версии. Ниже перечислено несколько типичных сообщений об ошибках и предупреждениях, выводимых системой по мере генерации кода.

- **Error: Missing attribute data type. Void is assumed**. (Ошибка: отсутствует тип данных атрибута; подразумевается тип **void**.)
- **Warning**: **Unspecified multiplicity/cardinality indicators. One is assumed**. (Предупреждение: не заданы признаки множественности; подразумевается значение "один".)
- **Warning: Missing operation return type. Void is assumed**. (Предупреждение: отсутствует тип значения, возвращаемого операцией; подразумевается тип **void**.)

Окно **Output Window** показано на рис. 5.

Рисунок 5. Вид окна протокола работы *Rational Rose*

Вариант лабораторной работы выбирается по последней цифре зачетной книжки.

Вариант	Тема
1	Учет автомобилей на автостоянке и расчет прибыли.
2	Формирование чека для оплаты покупок в супермаркете.
3	Клиент сдает автомобиль в автосервис.
4	Покупатель покупает книгу в книжном магазине.
5	Клиент берет телевизор в пункте проката.
6	Пассажир приходит на регистрацию рейса в аэропорт.
7	Клиент снимает квартиру через агентство недвижимости.
8	Покупатель оформляет кредит на покупку товара.
9	Клиент заказывает печать фотографий и фотосувениров.
0	Клиент делает заказ на изготовление мебели.

Пример классов, атрибутов и операций:

Вариант: Пассажир бронирует билет на рейс у агента.

Классы: пассажир с атрибутами: Имя, фамилия, адрес, № паспорта, город вылета, город прилета.

с операциями: заказать, купить,

агент с атрибутами: фамилия, номер агента,

с операциями: бронировать, продать.

Вопросы для самопроверки

1. Какие диаграммы UML отражают статическую структуру проектируемой системы?
2. Какие диаграммы UML отражают динамическую структуру проектируемой системы?
3. Что такое поток событий, из каких элементов он состоит?
4. Назовите основные структурные компоненты *Rational Rose* и их функции.
5. Что содержит в себе представление вариантов использования в *RationalRose*?
6. Как переопределить свойство элемента модели?
7. Как выбрать язык программирования для компонента?
8. Как отнести класс к компоненту?
9. Как связать набор свойств с элементом модели?
10. Как определить или создать стереотип компонента?
11. Как сгенерировать код в *Rational Rose*?