Лабораторная работа № 11
Связи

Назначение и виды связей

Классы редко бывают изолированы, чаще всего они вступают в отношения друг с другом. Эти отношения показываются при помощи различного вида связей. Типы связей влияют на получаемый при генерации на основе диаграмм исходный код, поэтому необходимо рассмотреть связи и спецификации подробно.

В диаграмме классов используются следующие виды связей:

- unidirectional association (однонаправленная ассоциация);
- dependency (зависимость);
- association class (ассоциированный класс);
- generalization (наследование);
- realization (реализация).

Unidirectional Association

Одна из важных и сложных типов связи, которая используется в диаграмме классов. Данная связь показывает, что один класс включается в другой как атрибут, по ссылке или по значению (рис. 81).

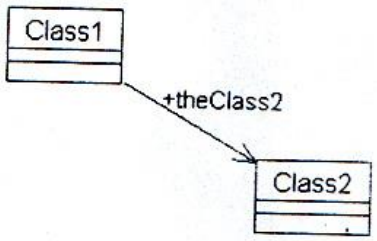

Рис. 81. Пример связи Unidirectional Association

Приведенный далее листинг показывает код C++, который был создан для показанной на рисунке связи.

```
class Class2;
class Class1
{public:
	Class2*theClass2;
};
```

Rational Rose создает код класса в зависимости от установленных спецификаций связи, поэтому рассмотрим влияние различных установок спецификаций на получаемый код. При нажатии правой кнопки мыши на связи активизируется контекстное меню, которое предоставляет быстрый доступ к установкам связи. Однако спецификации связи позволяют проделать то же самое при помощи вкладок диалоговых окон.

Активизируйте окно спецификаций при помощи контекстного меню или двойного нажатия мыши на стрелке ассоциации, при этом открывается вкладка General спецификаций (рис. 82).

Вкладка General

Показывает информацию об имени, стереотипе, родительском классе другую основную информацию о связи.

- Name (имя) задает имя связи. Для каждой связи может быть, хотя и не обязательно задано имя, которое одним словом или целой фразой указывает цель или семантику связи;
- Parent (родительский) указывает имя контейнера, которому принадлежит связь;
- Stereotype (стереотип) указывает стереотип;

- Role A/ Role B указывает имя роли, с которой один класс ассоциирует с другим;
- Element A/ Element B указывает имя класса, который ассоциирован с данной ролью.

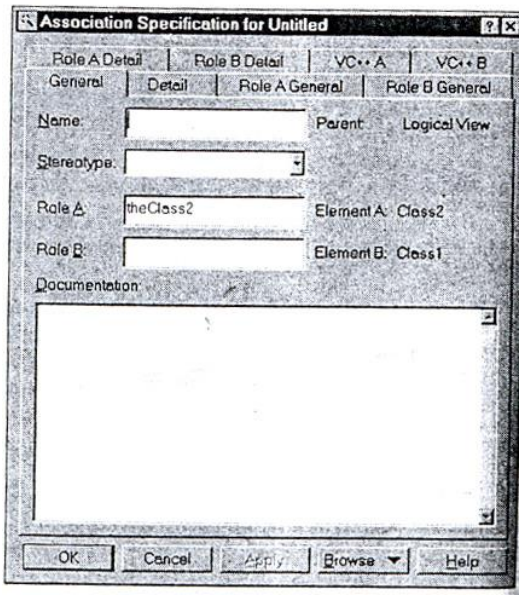

Рис.82. Вкладка General спецификаций Unidirectional Association

Вкладка Detail

Вкладка Detail указывает дополнительные свойства связи (рис. 83).

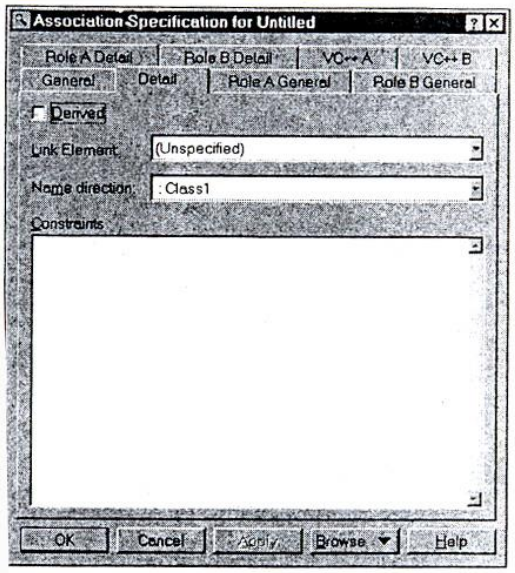

Рис.83.Вкладка Detail спецификаций Unidirectional Association

- Link Element указывает ассоциированный класс, если связь соединена с ним ассоциацией;
- Name указывает имя связанного класса;
- Constraints указывает выражение некоторого семантического условия, которое должно быть выполнено, в то время как система находится в устойчивом состоянии.

Вкладка Role General

Вкладка Role General отражает настройки переменной, которая будет включена в класс (рис. 84).

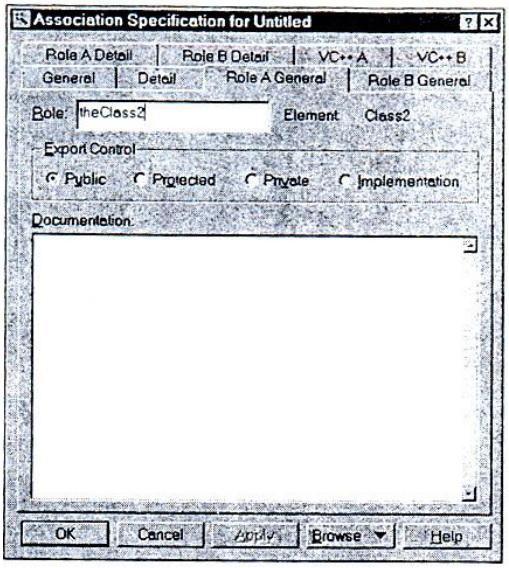

Рис. 84. Вкладка Role General спецификаций Unidirectional Association

Так как направление связи а нашем примере от Class1 к Class2, то будем заполнять вкладку Role A General. Эта вкладка имеет следующие поля:

- Role позволяет задавать имя переменной класса;
- Element показывает имя класса, для которого создается переменная;
- Export Control определяет доступ к данному элементу. В нашем случае установлен переключатель Public, поэтому переменная была создан в секции Public.

Вкладка Role Detail

Вкладка Role Detail детализирует установки для связи (рис. 85).

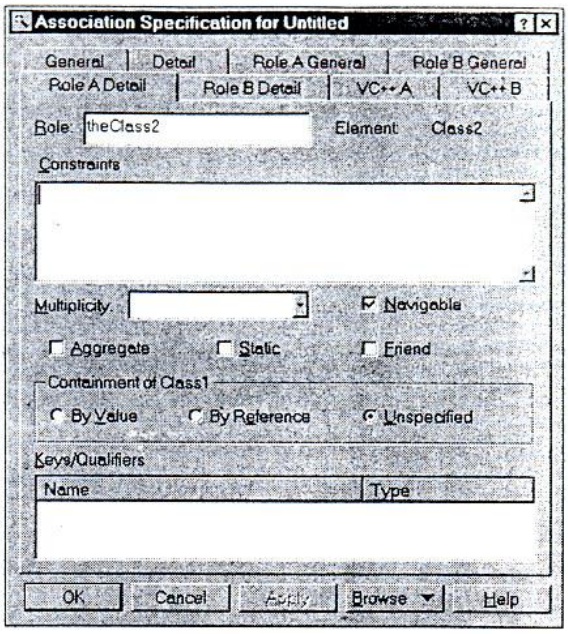

Рис. 85. Вкладка Role Detail A спецификаций Unidirectional Association

Здесь представлены следующие поля:

- Role позволяет задавать имя переменной класса;
- Element оказывает имя класса, для которого создается переменная;
- Constraints указывает выражение некоторого семантического условия, которое должно быть выполнено, в то время как система находится в устойчивом

состоянии. При задании ограничения оно будет показано на диаграмме в фигурных скобках. Причем на формируемом коде данное ограничение не будет отражаться.

- Multiplicity показывает, сколько ожидается создать объектов данного класса, может быть задано числом или буквой «n». Значение «n» указывает, что количество не лимитировано. Можно задать различные варианты ожидаемого количества или диапазон. Прячем, указанное число отображается рядом со стрелкой связи;
- Navigable показывает направление, в котором действует связь. При установке этого флажка связь приобретает вид стрелки, указывающей направление связи. Это поле напрямую влияет на создаваемый код класса, так как на какой класс будет направлена стрелка связи, тот и будет включен в другой. Для того чтобы изменить направление связи, достаточно снять флажок с вкладки Role A Detail и установить его во вкладке Role B Detail. При этом в случае, кода сняты флажки на обеих вкладках ни один элемент не будет включен в другой. При этом на диаграмме будет показана просто линия;
- Aggregate показывает, что один класс не просто использует, а содержит другой. Однако будьте внимательны, для того чтобы показать, что класс Class2 входит в класс Class1, необходимо установить этот флажок во вкладке Role B Detail. При этом стрелка связи на диаграмме приобретает ромб с обратной стороны стрелки (рис. 86).

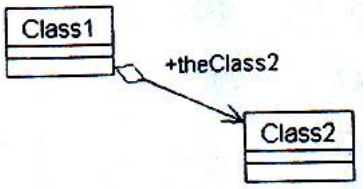

Рис. 86. Пример агрегирования класса

Агрегирование означает физическое включение связанного класса в другой класс. При этом генератором кода C++ Rational Rose создает код, приведенный на листинге.

```
Class Class2;
Class Class1
{public:
	Class2 theClass2;
};
```

- Static обозначает, что данный реквизит - общий для всех объектов данного класса. Причем, после инициализации к нему можно обращаться, даже если еще не было создано ни одного объекта класса. Static применяется для того чтобы переменные такого типа не тиражировались при создании нового объекта класса. Например, если необходимо точно знать, сколько объектов класса создано, то можно поручить самому классу следить за этим числом, создав в нем переменную типа static int iCnt, видимую во всех объектах класса;
- Friend определяет, что указанный класс является дружественным классом, то есть имеет доступ к защищенным методам и атрибутам класса.
- Key/Qualifiers атрибут, который идентифицирует уникальным образом единичный объект. На генерацию кода влияния не оказывает.

Association Class (ассоциированный класс)

Используйте данный тип связи для отображения свойства ассоциации. Свойства сохраняются в классе и соединяются связью Association (рис. 87). Этот тип связи не имеет своих спецификаций.

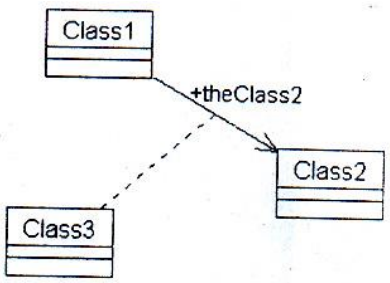

Рис.87. Пример использования Association Class

Dependency of instanties (зависимость)

Этот тип связи позволяет показать, что один класс использует объекты другого. Использование может осуществляться при передаче параметров или вызове операций класса. В этом случае генератор кода C++ Rational Rose включает заголовочный файл в класс, который использует операторы или объекты другого класса. Графически этот вид связи отражается пунктирной стрелкой (рис. 88).

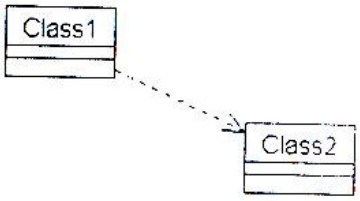

Рис. 88. Пример связи Dependency of instanties

Generalization

Данный тип связи позволяет указать, что один класс является родительским по отношению к другому, при этом будет создан код наследования класса. Пример такой связи показан на рис. 89.

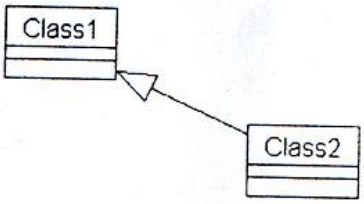

Рис.89 Пример связи Generalization

Далее показан листинг, полученный при генерации кода для показанного примера

```
#include “Class1.h”
class Class2: public Class1
{
};
```

Далее для работы будем пользоваться двумя видами связей — это Unidirectional Association, для агрегирования включения ссылок на классы, и Generalization, для создания иерархии наследования.