Лекция – UML-диаграммы. Отношения между классами (продолжение)

Программная демонстрация отношений между классами

Ассоциация реализуется посредством *указателей* в качестве дополнительных членов-данных.

Указатель (*pointer*) позволяет изменить связь или сделать её нулевой.

Для ***однонаправленной*** ассоциации используется *односторонняя реализация*, т.е. когда указатель на класс конечного полюса ассоциации объявляется в классе её начального полюса.

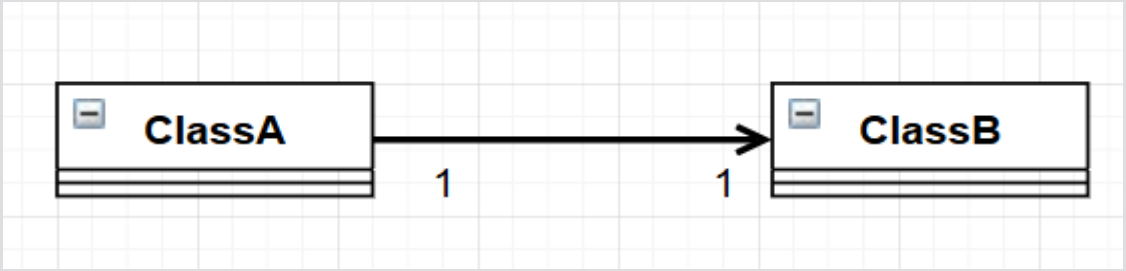
Если реализуется ассоциация в обоих направлениях, то применяется *двусторонняя реализация*, и ***указатели добавляются в оба класса***.

Такой подход обеспечивает быстрый доступ, но, если объект у одного из полюсов ***обновляется***, на втором полюсе также необходимо выполнить ***обновление***, чтобы связь объектов осталась *согласованной*.

Рассмотрим примеры программной демонстрации отношения ассоциация в зависимости от направленности и кратности.

Программная демонстрация отношений между классами

1. Пример кратности один-к-одному

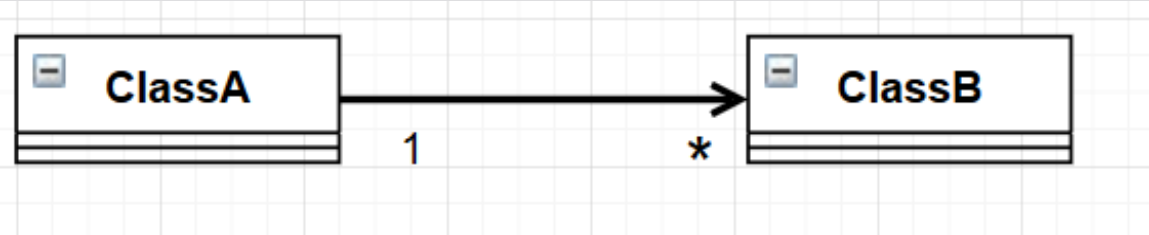


Такую связь можно представить следующим образом:

```
class ClassB { … };
class ClassA {ClassB* ptrClassB; … };
```

Программная демонстрация отношений между классами

2. Пример кратности один-ко-многим




```
class ClassB{};
class ClassA {ClassB ** ptrClassB;};
```

или

```
class ClassA {vector< ClassB* > dataPtrCB; };
```

Когда кратность полюса *много*, придётся объявить *указатель на указатель*, например, **** ptrClassB**, для организации двумерной структуры для хранения элементов или корректный *контейнерный класс*, например **vector< ClassB* > dataPtrCB**

Программная демонстрация отношений между классами

3. Пример кратности один к одному двунаправленной ассоциации

Организация отношения в классах будет выглядеть следующим образом.

```
class ClassB { … };
class ClassA {ClassB* ptrClassB; … };
class ClassA;
class ClassB {ClassA* ptrClassA; … };
```

Напомним, что в этом случае, если объект у одного из полюсов ***обновляется***, на втором полюсе также необходимо выполнить ***обновление***, чтобы связь объектов осталась *согласованной*.

Программная демонстрация отношений между классами

4. Пример кратности один-ко-многим двунаправленной ассоциации

Здесь, отношения формируются также как и в примере 2, за исключением класса ClassB.

class ClassB {ClassA* ptrClassA; … };

Когда полюс ассоциации должен быть привязан к **объектам в момент инициализации** и не подлежит дальнейшим изменениям, выбирается **явное объявление** или **ссылка.**

Ссылка (*reference*) получает своё значение в момент инициализации и (в отличие от указателя) не может быть изменена или сделана нулевой. Например, если объект класса **ClassA** должен быть привязан к объекту класса **ClassB** в момент его создания и не меняется в процессе работы программы, объявление будет такое: **class ClassB {ClassA & refClassA;**
…

Программная демонстрация отношений между классами

Агрегация реализуется посредством *указателей* так же, как ассоциация. Чаще всего, ***агрегация*** направлена от **класса *целого*** к **классам *частей***, поэтому части не имеют указателей на *целое*.

Внешним объектам разрешён доступ и к объектам *целого*, и **непосредственно** к объектам *частей*.

Композиция также рассматривается как ассоциация, но зависимость *частей* от *целого* обеспечивается дополнительно. Рассмотрим примеры в программной реализации. поддержки композиции.

1. Пример композиции с кратностью один к одному

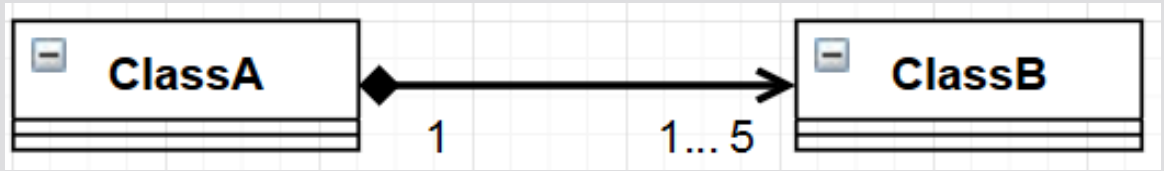

```
class ClassB { … };
class ClassA {ClassB objB; … };
```

Программная демонстрация отношений между классами

2. Пример композиции с кратностью один-ко-многим


```
class ClassB { … };
class ClassA {контейнер объектов<ClassB> objectsB; …
};
```

3. Пример композиции с ограниченной кратностью 5




```
class ClassB { … };
class ClassA { ClassB[5] objectsB; … };
```

Программная демонстрация отношений между классами

4.Пример композиции с кратностью один к одному с двунаправленной ассоциацией


```
class ClassA;
class ClassB {ClassA objClassA; … };
```

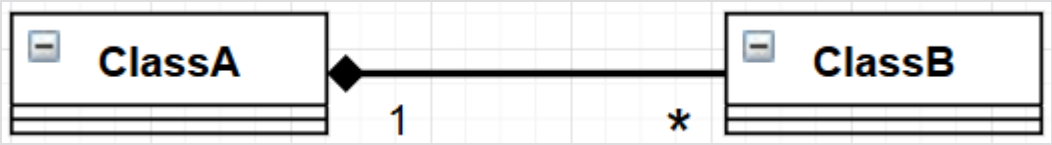
или

```
class ClassA;
class ClassB {ClassA & refClassA; … };


class ClassB;
class ClassA {ClassB objClassB; … };
```

Программная демонстрация отношений между классами

5.Пример композиции с кратностью0 один-ко-многим с двунаправленной ассоциацией




```
class ClassA;
class ClassB {ClassA objClassA; … };
```

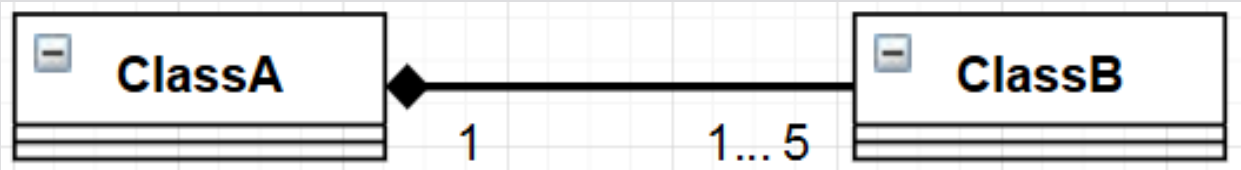
или

```
class ClassA;
class ClassB {ClassA & refClassA; … };

class ClassB;
class ClassA {контейнер объектов<ClassB> objectsB; … };
```

Программная демонстрация отношений между классами

6. Пример композиции с ограниченной кратностью с двунаправленной ассоциацией




```
class ClassA;
class ClassB {ClassA objClassA; … };
```

или

```
class ClassA;
class ClassB {ClassA & refClassA; … };


class ClassB;
class ClassA { ClassB[5] objectsB; … };
```

Программная демонстрация отношений между классами

Для кратности композиции *один-к-одному* или *ограниченной кратности*, объявление *частей* в классе *целого* производится явно.

Для кратности *один-ко-многим*, помимо контейнера объектов *частей*, можно пользоваться указателем на указатель (или массивом указателей) точно так же, как для обычной ассоциации.

Композиция при использовании указателей требует строгого контроля связей между объектами-участниками, поскольку удаление *целого* должно разрушать все его *части*.

Важно помнить, что при композиции внешние объекты получают доступ к любому объекту *части* **только** через объект *целого*, который в свою очередь вызывает методы классов *частей*.

Таким образом, в программе *части* всегда видны только своему *целому* и не видны **никаким** внешним объектам.