2.4. Объекты

И инженер, и художник должны хорошо чувствовать материал, с которым они работают. В объектно-ориентированной методологии анализа и создания сложных программных систем основными строительными блоками являются классы и объекты.

Объектом называют нечто обладающее состоянием, поведением и идентичностью. Структура и поведение схожих объектов определяет общий для них класс. Термины «экземпляр класса» и «объект» взаимозаменяемы. Понятия класса и объекта связаны, однако существует важное различие между ними. Объект обозначает конкретную сущность, определенную во времени и в пространстве. Класс определяет абстракцию существенного в объекте (его данные и поведение). Например, дом №2 по улице Лесной в нашем городе - объект класса House (Дом). Класс House определяет, что у дома должны быть высота, ширина, количество комнат. Объект этого класса - дома №2 по улице Лесной - может иметь высоту 20 футов, ширину 60 футов, 10 комнат. Класс - более общий термин, являющийся, по существу, шаблоном для объектов. Представьте себе, что класс - это проект дома, а объекты - это 5 построенных по проекту домов. ***Класс*** - это некое множество объектов, имеющих общую структуру и общее поведение.

Объектами могут быть осязаемые и видимые предметы (например, дом, цветок и т.п.).

Объекты могут быть осязаемыми, но иметь размытые физические границы: реки, туман или толпы людей.

Существуют такие объекты, для которых определены явные концептуальные границы, но сами объекты представляют собой неосязаемые события или процессы. Например, химический процесс на заводе можно трактовать как объект, так как он имеет четкую концептуальную границу, взаимодействует с другими объектами посредством упорядоченного и распределенного во времени набора операций и проявляет хорошо определенное поведение.

Объекты могут получаться из отношений между другими объектами. Два тела, например, сфера и куб, имеют, как правило, нерегулярное пересечение. Хотя эта линия пересечения не существует отдельно от сферы и куба, она все же является самостоятельным объектом с четко определенными концептуальными границами.

2.4.1. Состояние

Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.

К числу ***свойств*** объекта относятся присущие ему или приобретаемые им характеристики, черты, качества или способности, делающие данный объект самим собой. Например, для лифта характерным является то, что он сконструирован для поездок вверх и вниз, а не горизонтально. Перечень свойств объекта является, как правило, статическим, поскольку эти свойства составляют неизменяемую основу объекта. Однако в ряде случаев состав свойств объекта может изменяться. Примером может служить робот с возможностью самообучения. Робот первоначально может рассматривать некоторое препятствие как статическое, а затем обнаруживает, что это дверь, которую можно открыть. В такой ситуации по мере получения новых знаний изменяется создаваемая роботом концептуальная модель мира.

Все свойства имеют значения, которые могут изменяться. Эти значения могут быть простыми количественными характеристиками, а могут ссылаться на другой объект. Состояние лифта может описываться числом три, означающим номер этажа, на котором лифт в данный момент находится.

2.4.2. Поведение

Объекты не существуют изолированно, а подвергаются воздействию или сами воздействуют на другие объекты.

Поведение - это то, как объект действует и реагирует; поведение выражается в терминах состояния объекта и передачи сообщений (операций).

Иными словами, поведение объекта - это его наблюдаемая и проверяемая извне деятельность.

Операцией называется определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию. Например, клиент может активизировать операции append (добавить) и pop (изъять элемент) длятого, чтобы управлять объектом-очередью. Существует также операция length, которая позволяет определить размер очереди, но не может изменить это значение. В основном понятие ***сообщение*** совпадает с понятием ***операции*** над объектами, хотя механизм передачи различен. В объектно-ориентированных языках операции, выполняемые над данным объектом, называются ***методами*** и входят в определение класса объекта.

Передача сообщений (операции) - это одна часть уравнения, задающего поведение. Из нашего определения следует, что состояние объекта также влияет на его поведение. Рассмотрим торговый автомат. Мы можем сделать выбор, но поведение автомата будет зависеть от его состояния. Если мы не опустили в него достаточную сумму, скорее всего ничего не про-

изойдет. Если же денег достаточно, автомат выдаст нам желаемое (и тем самым изменит свое состояние).

Итак, поведение объекта определяется выполняемыми над ним операциями и его состоянием, причем некоторые операции имеют побочное действие: они изменяют состояние. Концепция побочного действия позволяет уточнить наше определение состояния: ***состояние объекта*** представляет суммарный результат его поведения.

Наиболее интересны те объекты, состояние которых не статично, а изменяется и запрашивается операциями.

Для примера опишем на языке C++ класс Queue (очередь):

```
class Queue {
public:
   Queue();
   Queue(const Queue&);
   virtual ~Queue();
   virtual Queue& operator=(const Queue&);
   virtual int operator==(const Queue&) const;
   int operator!=(const Queue&) const;
   virtual void clear();
   virtual void append(const void*);
   virtual void pop();
   virtual void remove(int at);
   virtual int length() const;
   virtual int isEmpty() const;
   virtual const void* front() const;
   virtual int location(const void*);
protected:
...
};
```

В определении класса используется обычная для C идиома ссылки на данные неопределенного типа с помощью void*, благодаря чему в очередь можно вставлять объекты разных классов. Эта техника небезопасна, клиент должен ясно понимать, с объектом какого класса он имеет дело. Кроме того, при использовании void* очередь не «владеет» объектами, которые в нее помещены. Деструктор ~Queue() уничтожает очередь, но не ее участников (параметризованные типы позволяют справляться с такими проблемами).

Так как определение Queue задает класс, а не объект, мы должны объявить экземпляры класса, с которыми могут работать клиенты:

```
Queue a, b, c, d;
```

Можно выполнить следующие операции над объектами:

```
a.append(&deb);
a.append(&karen);
a.append (&denise);
b = a;
a.pop();
```

Теперь очередь a содержит двух сотрудников (первой стоит karen), а очередь b - трех (первой стоит deb). Таким образом, очереди имеют определенное состояние, которое влияет на их будущее поведение. Например, одну очередь можно безопасно продвинуть (pop) еще два раза, а вторую - три.

С точки зрения класса объекта ***операция*** - это услуга, которую класс может предоставить своим клиентам. На практике типичный клиент совершает над объектами операции пяти видов (в разделе третей главы, который посвящен операциям, вы найдете несколько иную классификацию: *операции реализации, операции управления, операции доступа и вспомогательные операции*). Ниже приведены три наиболее распространенные операции:

- ***модификатор*** - операция, которая изменяет состояние объекта;
- ***селектор*** - операция, считывающая состояние объекта, но не меняющая состояния;
- ***итератор*** - операция, позволяющая организовать доступ ко всем частям объекта в строго определенной последовательности.

Кроме них существуют две универсальные операции. Они обеспечивают инфраструктуру, необходимую для создания и уничтожения экземпляров класса. Это:

- ***конструктор*** - операция создания объекта и/или его инициализации;
- ***деструктор*** - операция, освобождающая состояние объекта и/или разрушающая сам объект.

Операции могут быть не только методами класса, но и независимыми от объектов свободными подпрограммами. Свободные подпрограммы группируются в соответствии с классами, для которых они создаются. Это дает основание называть такие пакеты процедур утилитами класса. Например, для определенного выше класса Queue можно написать следующую свободную процедуру:

```
void copyUntilFound(Queue& from, Queue& to, void* item)
{
   while ((!from.isEmpty()) && (from.front() != item)) {
         to.append(from.front());
         from.pop();
         }
}
```

Смысл в том, что содержимое одной очереди переходит в другую до тех пор, пока в голове первой очереди не окажется заданный объект. Это операция высокого уровня, она строится на операциях-примитивах класса Queue.

Таким образом, можно утверждать, что все методы - операции, но не все операции - методы: некоторые из них представляют собой свободные подпрограммы.

Совокупность всех методов и свободных подпрограмм, относящихся к конкретному объекту, образует ***протокол*** этого объекта. Протокол, таким образом, определяет поведение объекта, охватывающее все его статические и динамические аспекты. В самых нетривиальных абстракциях полезно подразделять протокол на частные аспекты поведения, которые мы будет называть ***ролями***.

Объединяя определения состояния и поведения объекта, вводят понятие ответственности. ***Ответственность объекта*** имеет две стороны - знания, которые объект поддерживает, и действия, которые объект может исполнить. Они выражают смысл его предназначения и место в системе. Ответственность понимается как совокупность всех услуг и всех контрактных обязательств объекта. Таким образом, можно сказать, что состояние и поведение объекта определяют исполняемые им роли, а те, в свою очередь, необходимы для выполнения ответственности данной абстракции.

Большинство объектов исполняют в своей жизни разные роли, например:

1) банковский счет может быть в хорошем или плохом состоянии (две роли), и от этой роли зависит, что произойдет при попытке снятия с него денег;

2) для фондового брокера пакет акций - это товар, который можно покупать или продавать, а для юриста это знак обладания определенными правами;

3) в течение дня одна и та же персона может играть роль матери, врача, садовника и кинокритика.

Роли банковского счета являются динамическими и взаимоисключающими. Роли пакета акций слегка перекрываются, но каждая из них зависит от того, что клиент с ними делает. В случае персоны роли динамически изменяются каждую минуту.

Мы часто начинаем анализ задачи с перечисления разных ролей, которые может играть объект. Во время проектирования мы выделяем эти роли, вводя конкретные операции, выполняющие ответственности каждой роли.

2.4.3. Идентичность

Идентичность - это такое свойство объекта, которое отличает его от всех других объектов.

Не следует путать идентичность объекта с его ключевыми атрибутами, именем или номером.

2.4.4. Время жизни объекта

Началом времени существования любого объекта является момент его создания (отведение участка памяти), а окончанием - возвращение отведенного участка памяти системе.

Объекты создаются явно или неявно. Есть два способа создать их явно. Во-первых, это можно сделать при объявлении, тогда объект размещается в стеке. Во-вторых, можно разместить объект, выделив ему память из «кучи». В C++ в любом случае при этом вызывается конструктор, который выделяет известное ему количество правильно инициализированной памяти под объект.

Часто объекты создаются неявно. Так, передача параметра по значению в C++ создает в стеке временную копию объекта. Более того, создание объектов транзитивно: создание объекта тянет за собой создание других объектов, входящих в него. Переопределение семантики копирующего конструктора и оператора присваивания в C++ разрешает явное управление тем, когда части объекта создаются и уничтожаются. К тому же в C++ можно переопределять и оператор new, тем самым изменяя политику управления памятью в «куче» для отдельных классов.

При явном или неявном уничтожении объекта в C++ вызывается соответствующий деструктор. Его задача не только освободить память, но и решить, что делать с другими ресурсами, например, с открытыми файлами (Деструкторы не освобождают автоматически память, размещенную оператором new, программисты должны явно освободить ее).