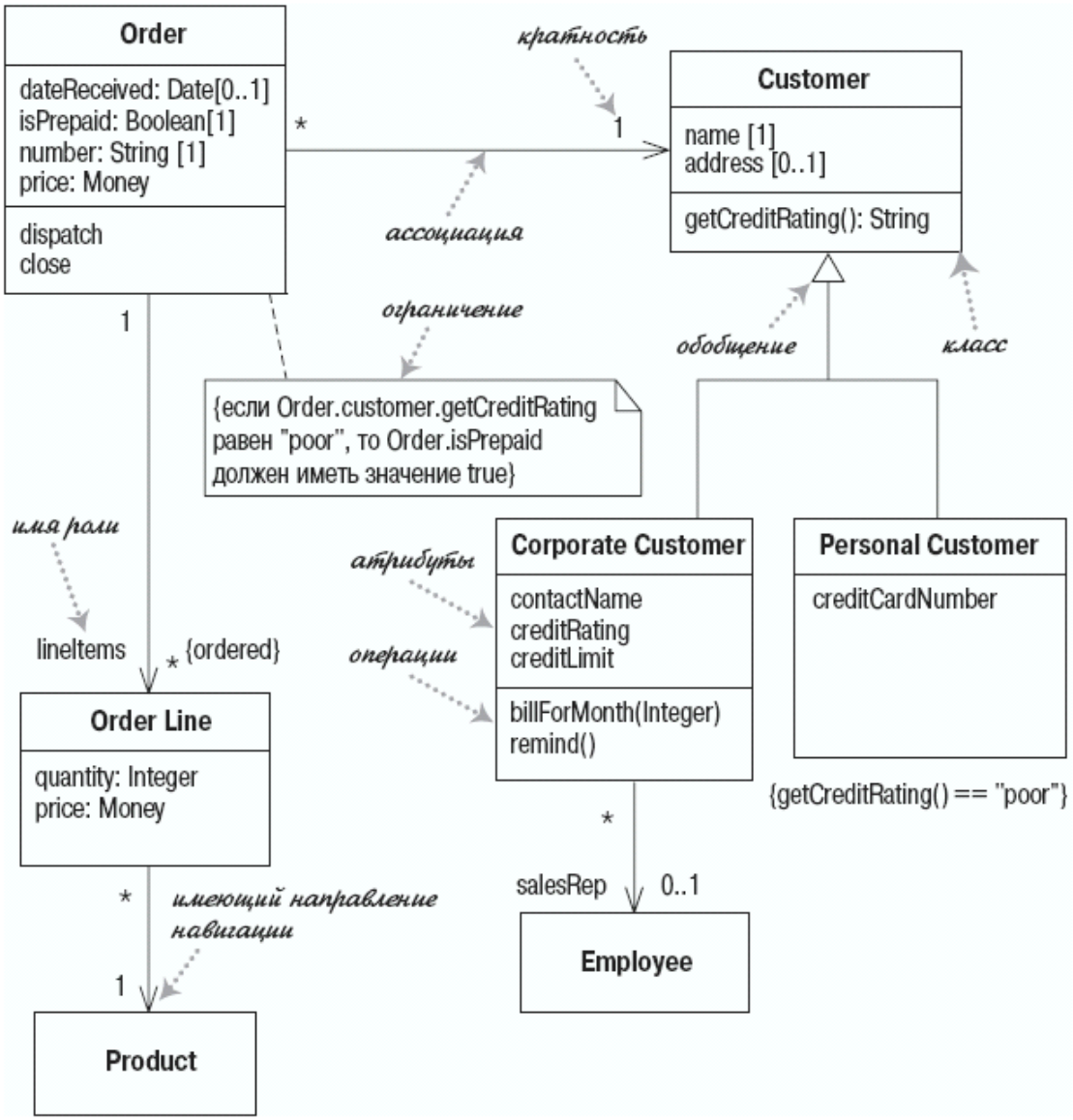
Диаграммы классов

Наиболее простой и самый распространенный тип диаграмм. Предназначены для описания типов объектов и отношений между ними.



Атрибут описывает свойство в виде строки текста внутри прямоугольника класса. Полная форма атрибута:

```
видимость имя: тип кратность = значение по умолчанию {строка свойств}
```

Например:

```
- имя: String [1] = "Без имени" {readOnly}
```

Обязательно только имя.

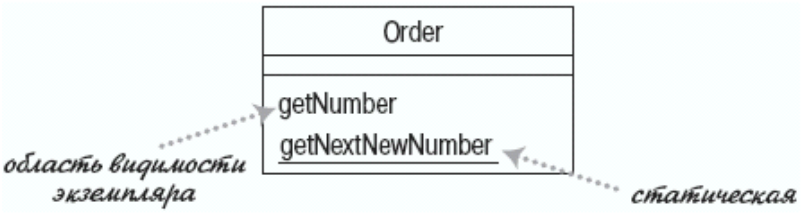
- Метка видимость обозначает, относится ли атрибут к открытым (+) (public) или к закрытым (-) (private). Другие типы видимости обсу-

- Имя атрибута – способ ссылки класса на атрибут – приблизительно соответствует имени поля в языке программирования.
- Тип атрибута накладывает ограничение на вид объекта, который может быть размещен в атрибуте. Можно считать его аналогом типа поля в языке программирования.
- Кратность рассмотрена на *стр. 65*.
- Значение по умолчанию представляет собой значение для вновь создаваемых объектов, если атрибут не определен в процессе создания.
- Элемент {строка свойств} позволяет указывать дополнительные свойства атрибута. В примере он равен {readOnly}, то есть клиенты не могут изменять атрибут. Если он пропущен, то, как правило, атрибут можно модифицировать. Остальные строки свойств будут описаны позже.

Кратность свойства обозначает количество объектов, которые могут заполнять данное свойство. Чаще всего встречаются следующие кратности:

- 1 (Заказ может представить только один клиент.)
- 0..1 (Корпоративный клиент может иметь, а может и не иметь единственного торгового представителя.)
- * (Клиент не обязан размещать заказ, и количество заказов не ограничено. Он может разместить ноль или более заказов.)

Статические члены



Ассоциации

Представление ассоциаций в виде свойств (но лучше так не делать):

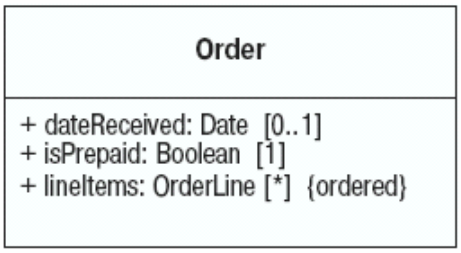
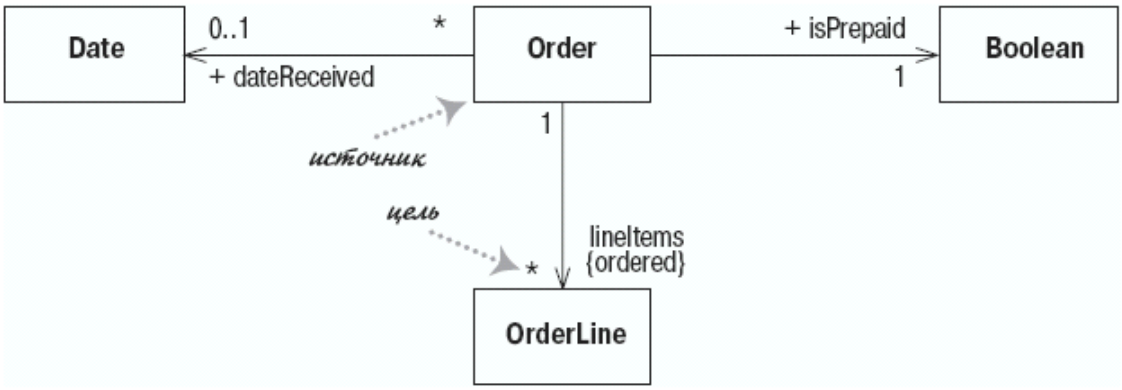


Рис. 3.2. Представление свойств заказа в виде атрибутов



Операции (operations) представляют собой действия, реализуемые некоторым классом. Существует очевидное соответствие между операциями и методами класса. Обычно можно не показывать такие операции, которые просто манипулируют свойствами, поскольку они и так подразумеваются.

Полный синтаксис операций в языке UML выглядит следующим образом:

```
видимость имя (список параметров) : возвращаемый тип {строка свойств}
```

- Метка `видимости` обозначает, относится ли операция к открытым (+) (public) или к закрытым (-) (private); другие типы видимости обсу-
- `Имя` – это строка.
- `Список параметров` – список параметров операции.
- `Возвращаемый тип` – тип возвращаемого значения, если таковое есть.
- `Строка свойств` – значения свойств, которые применяются к данной операции.

Параметры в списке параметров обозначаются таким же образом, что и для атрибутов. Они имеют вид:

```
направление имя: тип = значение по умолчанию
```

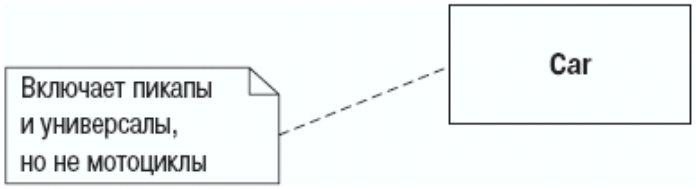
- Имя, тип и значение по умолчанию те же самые, что и для атрибутов.
- Направление обозначает, является ли параметр входным (`in`), выходным (`out`) или тем и другим (`inout`). Если направление не указано, то предполагается `in`.

Например, в счете операция может выглядеть так:

```
+ balanceOn (date: Date) : Money
```

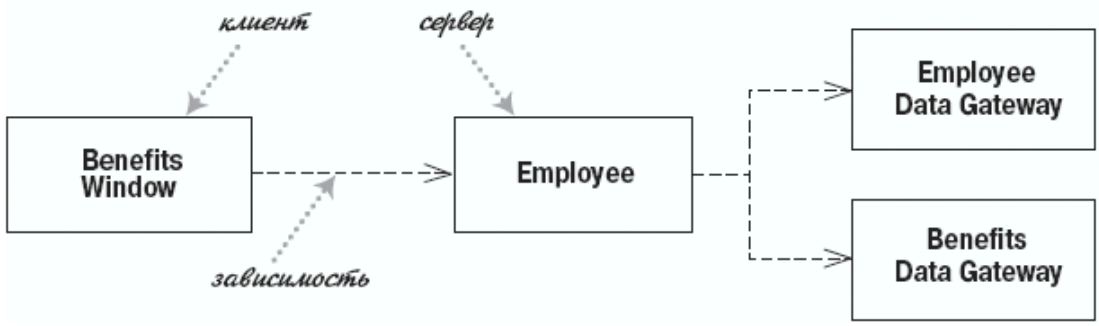
Примечания

Примечания – это комментарии на диаграммах. Примечания могут существовать сами по себе или быть связаны пунктирной линией с элементами, которые они комментируют (рис. 3.6). Они могут присутствовать на диаграммах любого типа.



Зависимости

Считается, что между двумя элементами существует **зависимость** (dependency), если изменения в определении одного элемента (**сервера**) могут вызвать изменения в другом элементе (**клиенте**). В случае классов зависимости появляются по разным причинам: один класс посылает сообщение другому классу; один класс владеет другим классом как частью своих данных; один класс использует другой класс в качестве параметра операции. Если класс изменяет свой интерфейс, то сообщения, посылаемые этому классу, могут стать недействительными.



Агрегация и композиция

Рис. 5.3. Агрегация

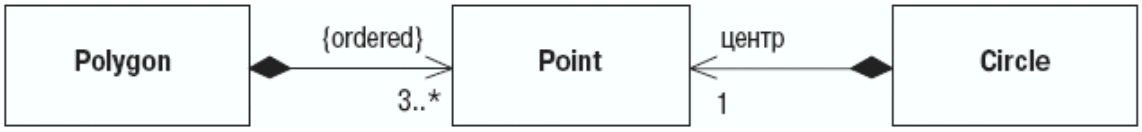
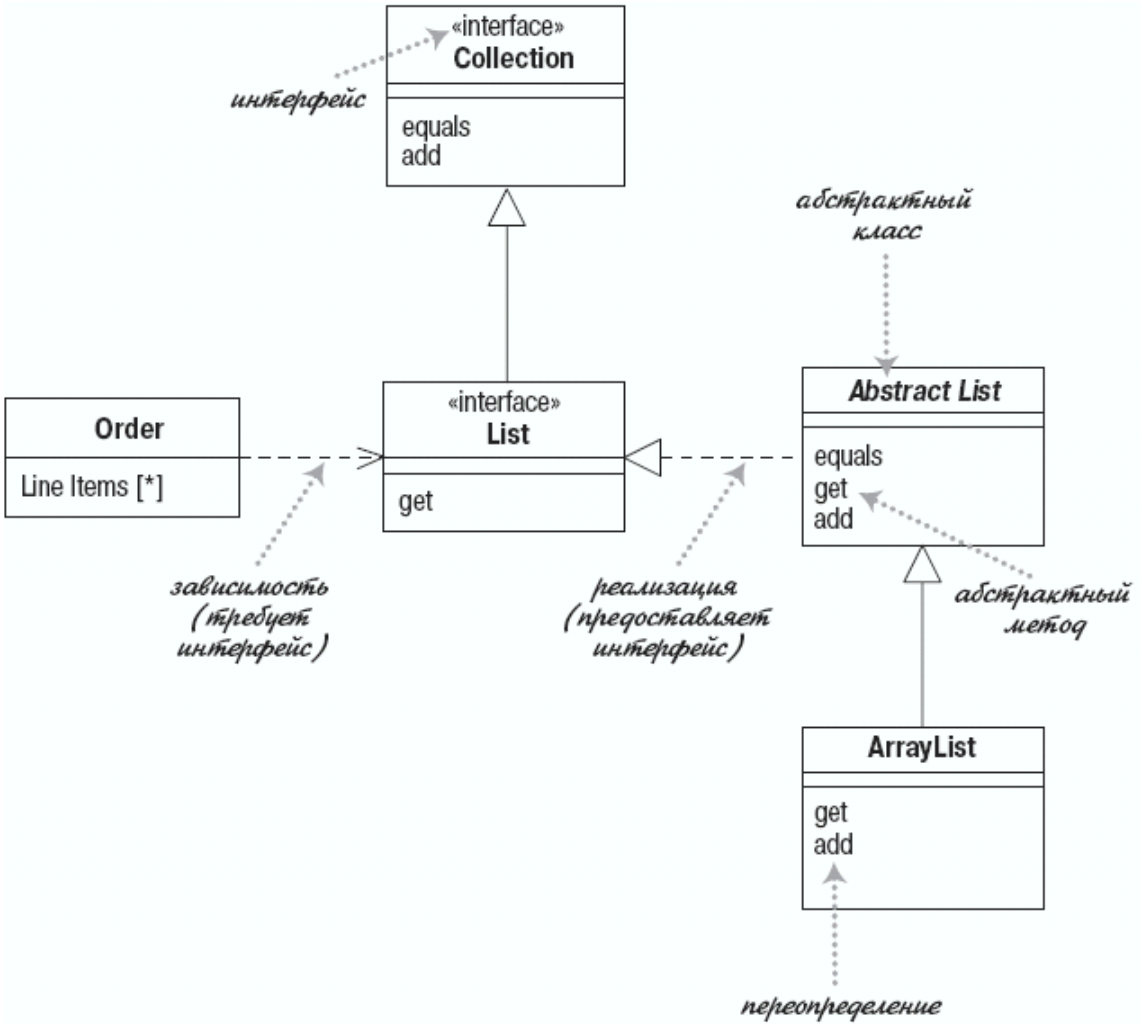
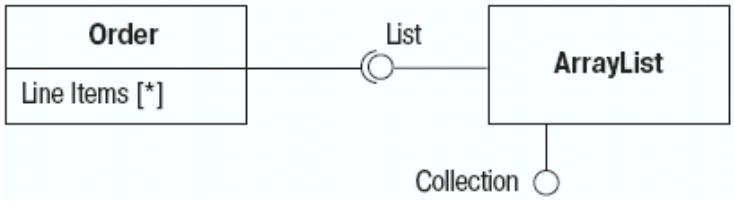
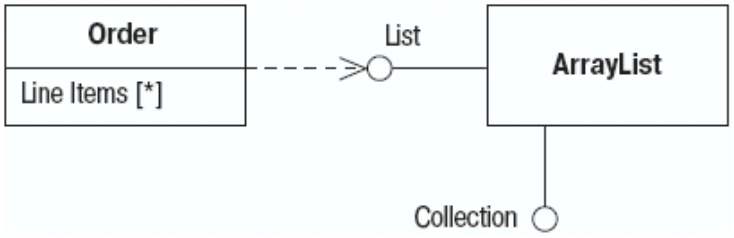


Рис. 5.4. Композиция

Наследование, реализация, абстрактные классы и интерфейсы







Ассоциативный класс:

Классы-ассоциации (association classes) позволяют дополнительно определять для ассоциаций атрибуты, операции и другие свойства, как показано на рис. 5.12. Из данной диаграммы видно, что `Person` (Личность) может принимать участие в нескольких совещаниях (`Meeting`). При этом необходимо каким-то образом хранить информацию о том, насколько внимательной была данная личность; это можно сделать, добавив к ассоциации атрибут `attentiveness` (внимательность).