КОНТЕЙНЕРЫ В МЕТОДОЛОГИИ ПРОГРАММИРОВАНИЯ

Прыгунков М.О.

Волгоградский государственный технический университет

Волгоград, Россия

Объектно-ориентированные языки поддерживают три базовых принципа объектной технологии:

1) инкапсуляция

2) наследование

3) полиморфизм

Принцип инкапсуляции. В основе любого объектного языка лежит принцип инкапсуляции. Этим термином обозначается способность языка скрывать второстепенные детали реализации от пользователя объекта.

Принцип наследования. В объектно-ориентированных языках можно моделировать связь путём объявления базового класса и производного от него. Классы связываются посредством классического наследования («А есть В») или с помощью модели контейнеров («А содержит В»). В любом случаи наследование всегда предполагает, что один класс является специальной формой другого.

Отношение «А есть В»: классическое наследование. В классическом наследовании подкласс перенимает существующую функциональность базового класса.

Отношение «А содержит В»: контейнеры и делегирование. Другой способ наследования является реализация отношения «А содержит В» и известен также как модель контейнеризации/делегирования (containment/delegation).

Принцип полиморфизма. Полиморфизм позволяет связанным классам единым образом реагировать на одинаковые сообщения. Так же как и с наследованием, существуют две формы полиморфизма – классический и конкретный (adhoc) полиморфизм.

Классический полиморфизм поддерживается только теми языками, которые поддерживают классическое наследование. Оба принципа тесно связаны друг с другом.

Рассматривая возможность агрегации нескольких объектов в новый класс – контейнер, необходимо определить основы агрегации. Каждый объект обладает некоторым декларированным интерфейсом, с помощью которого с ним могут взаимодействовать другие объекты. Контейнер, имея в своём составе некоторое множество объектов, может предоставлять вовне весь интерфейс вложенных объектов или только его часть. Совокупность интерфейсов, которые использует контейнер у вложенного объекта, назовём ролью вложенного объекта. Несколько вложенных объектов могут играть одинаковые или различные роли в одном и том же контейнере.

Понятие роли объекта не менее важно для агрегации, чем формальные параметры подпрограмм в структурном программировании. Это положение исходит из того, что понятие роли вводит виртуальность объектов в контейнере. Действительно, можно с полным правом сказать, что контейнер агрегирует объекты, способные выступать в таких-то ролях, то есть, объектов, имеющих некоторую определённую совокупность интерфейсов.

Понятие роли динамично, поскольку оно позволяет непосредственно при работе системы объявить ролью некоторый набор интерфейсов и потребовать от системы список классов, объекты которых способны выступать в данной роли. Для примера можно рассмотреть класс, который имеет интерфейсы A, B, C, D и F. В роли «альфа» объединим интерфейсы A и B, в роли «бета» - интерфейсы B, C и F, в роли «гамма» - A, C и D и т.д. Понятно, что подклассы всегда могут выступать в тех ролях. Однако, благодаря тому, что полиморфизм является понятием независимым от наследования (обратное неверно), то справедливо будет и утверждение, что классы, не находящиеся в наследственной связи, тоже способны играть одну и ту же роль.

Простота и удобство механизма виртуальности объектов в контейнере позволяют существенно облегчить разработку сложных систем за счёт перехода от упрощенных опытных моделей к промышленным образцам. С другой стороны, на основе данного вида виртуализации можно моделировать и реальную эволюцию сложных систем, например, биологических, технических,

социальных. Однако, как будет отмечено ниже, такие требования не являются непреодолимыми, поскольку, с одной стороны, контейнеры существенно упрощают вложенные классы, в том числе и спецификации интерфейсов, а, с другой стороны, наследование контейнеров, как обычных классов, приводит и к наследованию функционала, который представим в виде ролей вложенных объектов. И, наконец, у контейнеров, как и любых других классов можно будет «наращивать» функциональность по мере необходимости, повышая возможности системы. Поэтому можно установить порядок разработки контейнера, определяя тем самым приоритеты реализации ролей, то есть интерфейсов вложенных классов.

Отметим, что на основе механизма сборки программ из объектов в процессе выполнения программ можно даже предложить способ создания новых типовых элементов построения программ, названных «кубиками» из уже созданных «кубиков».