Куно Пфистер

Component Pascal и Java: что лучше

Источник: ComputerWeek-Moscow, 1998, # 14.

Component Pascal — язык программирования нового поколения, сочетающий в себе гибкость динамических языков с надежностью статических. Он соотносится с Pascal так же, как Java с C и C++. Подобно Java, Component Pascal, вобравший в себя опыт создания языков последних десятилетий, позволяет успешно использовать приобретенные программистами навыки. Как и Java, он значительно лучше своих предшественников поддерживает разработку компонентного ПО. Однако в отличие от Java этот язык может применяться для реализации сложных проектов либо проектов, требующих определенного времени на подготовку специалистов.

Наиболее очевидное различие между двумя языками в их "внешнем" виде. С синтаксической точки зрения Component Pascal принадлежит к семейству Pascal, а Java — к семейству C. Но это различие чисто внешнее. С точки зрения "встроенной безопасности", Java и Component Pascal очень близки, тогда как C и даже оригинальный Pascal (например, в части записей с вариантами без информации о текущем типе) сравнительно ненадежны. В числе прочего безопасность подразумевает автоматическую сборку мусора. Этот механизм необходим во избежание неосвобождения памяти и, что более существенно, появления неинициализированных указателей.

Component Pascal более эффективен, чем Java. Он поддерживает статические (размещаемые в стеке) структуры данных и передаваемые по значению параметры (VAR). Степень эффективности этого языка зависит от конкретного приложения. Рассмотрим, например, такой метод в Component Pascal:

```
font.GetMeasures(ascender, descender, maxWidth)
```

Он возвращает в качестве выходных параметров три значения. В Java подобный эффективный способ отсутствует. В рамках эквивалентного Java-метода пришлось бы создавать, инициализировать и возвращать вспомогательный объект, содержащий эти три выходных значения. Для большинства случаев такой механизм слишком "тяжеловесен": он требует не только динамического выделения памяти, но и создания нового класса и файла этого класса. Другое решение могло бы состоять во введении трех отдельных методов, каждый из которых бы возвращал по одному параметру. Однако в этом случае вместо одного вызова потребовалось бы целых три. Наконец, VAR-параметры могут эмулироваться с помощью одноэлементных массивов. Последний вариант является явным злоупотреблением механизмом массивов и влечет за собой ненужные проверки выхода за границу массива, операции по контролю типов и требует выделения памяти под вспомогательные объекты. В Java выделение памяти осуществляется внутренними механизмами языка также при слиянии строк и генерации исключительных ситуаций.

Интенсивное использование динамических механизмов распределения памяти и сборки мусора особенно нежелательно в системах реального времени, поскольку требование высокой скорости работы сильно ограничивает выбор алгоритмов, причем за бортом остаются самые гибкие и эффективные. Наконец, в Component Pascal более эффективно реализованы массивы, что особенно важно для математических и других приложений, включающих сложные расчеты.

В отличие от Java, Component Pascal не имеет встроенной поддержки потоков на уровне языка, поскольку это ограничило бы переносимость: Java-потоки неодинаково ведут себя на различных платформах. Более того, реализованная в Java схема слишком ограниченна для систем реального масштаба времени: она распределяет вычислительные ресурсы между потоками с учетом не контрольных сроков завершения того или иного процесса, а присвоенных им приоритетов. Поскольку встраиваемые системы реального времени с ограниченными вычислительными ресурсами являются одной из важнейших областей приложения Component Pascal, это обстоятельство немаловажно. В операционной системе реального времени Portos (http://www.oberon.ch/rtos/portos) поддержка потоков реализована в специальной библиотеке.

В Component Pascal имеется ряд оригинальных элементов (таких, как явный атрибут NEW для дополнительно введенных методов, "пустые" методы EMPTY, записи типа LIMITED), которые предназначены специально для повышения контролируемости крупных компонентных программных систем, создаваемых посредством повторного использования компонентов. В Component Pascal реализована более удачная схема именования, позволяющая компилятору лучше контролировать согласованность компонентов и соответствующих им компонентных инфраструктур. Это совершенно необходимо для реализации универсальной стыковки независимо разрабатываемых компонентов.

Возможно, наиболее существенное различие — это степень сложности: описание языка Java содержит больше тридцати классов, включающих свыше 300 методов. Более того, все элементы языка (а иногда и стандартные библиотечные классы) чрезвычайно тесно взаимосвязаны, так что зачастую бывает невозможно разобраться в одном из них без глубокого понимания всех остальных. В результате применение Java "от случая к случаю" оказывается весьма трудным делом и требует существенных временных затрат.

По сравнению с Java язык Component Pascal значительно компактнее и проще. Это объясняется двумя обстоятельствами. Во-первых, несмотря на свои довольно скромные размеры, Component Pascal допускает поэтапное освоение и использование, его более сложные возможности могут вводиться в действие постепенно, по мере необходимости. Во-вторых, многие элементы, реализованные в Java на уровне языка, Component Pascal оставляет на долю библиотек. Один из примеров — с механизмом потоков и синхронизации — был приведен выше. В результате становится возможной более рациональная структуризация механизмов и абстракций по уровням, тогда как в Java сам язык и крупные библиотечные блоки образуют огромные концептуально единые "сети" понятий.