УДК 004.4

ИССЛЕДОВАНИЕ СОВРЕМЕННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ KOTLIN

Чернов Владимир Владимирович
ГВУЗ «Национальный горный университет», http://www.nmu.org.ua

Рассмотрен современный язык программирования Kotlin, компилируемый в байт-код для платформы JVM и в JavaScript. Указаны основные проблемы, которые решает Kotlin и не решает Java. В тезисах опубликованно исследование особенностей требований, которые кажутся наиболее важными для такого проекта.

Ключевые слова – статически типизированный объектно-ориентированный язык; язык программирования Kotlin; JavaScript; переносимый байт-код.

ВСТУПЛЕНИЕ

В последние годы назрела потребность в новом языке, компилируемом в переносимый байт-код для виртуальной машины Java. Появилось несколько проектов по созданию таких языков и один из них – Kotlin, статически типизированный объектно-ориентированный язык [1].

Разработка проекта Kotlin началась летом 2010 года, в июле 2011-го проект был официально анонсирован и на их официальном сайте размещено описание языка. Выпуск публичной бета-версии компилятора запланирован на начало 2012 года. Язык разрабатывается в компании JetBrains. Код проекта доступен под свободной лицензией Apache 2 [2].

Компания JetBrains создает инструменты для программистов, которые позволяют ускорить процесс написания кода, тестирования, сопровождения, общения с пользователями и т.д [3].

ОСНОВНАЯ ЧАСТЬЯ

Две основные проблемы, которые решает Kotlin и не решает Java, это:

1) Превентивная борьба с вездесущим NPE. Если в Java системе null значения легко разбегаются по всему проекту. То в Kotlin разработчик строго контролирует где у него могут быть null значения и минимизирует их распространение. Что сводит проверки на null к минимуму.

2) Свойства и указатели на функции. Без этого очень страдает как ORM так и многие другие движки на рефлексии - биндинг, IoC и т.п. Программисты вынуждены везде описывать свойства строкой, а не прямой ссылкой на свойство. Поэтому нет ни проверки что имя написано правильно, ни проверки на тип свойства во время компиляции.

Создавая язык, компания JetBrains руководствовалась рядом требований, которые кажутся наиболее важными для такого проекта. Рассмотрим подробнее каждый из них:

Совместимость с Java. Платформа Java – это прежде всего *экосистема*: кроме «официальных» продуктов компании Oracle, в нее входит множество проектов с открытым кодом: библиотек и фреймворков разного профиля, на базе которых строится огромное количество приложений. Поэтому для языка, компилируемого для этой платформы, очень важна совместимость с существующим кодом, который написан на Java. При этом необходимо, чтобы существующие проекты могли переходить на новый язык постепенно, то есть не только код на Kotlin должен легко вызывать код на Java, но и наоборот.

Статические гарантии корректности. Во время компиляции кода на статически типизированном языке происходит множество проверок, призванных гарантировать, что те или иные ошибки не произойдут во время выполнения. Например, компилятор Java гарантирует, что объекты, на которых вызываются те или иные методы, «умеют» их выполнять, то есть что в соответствующих классах эти методы реализованы [5]. К сожалению, кроме этого очень важного свойства, Java почти ничего не гарантирует. Это означает, что успешно скомпилированные программы завершаются с ошибками времени выполнения (вызывают исключительные ситуации). Ярким примером является разыменование нулевой ссылки, при котором во время выполнения вызывается исключение типа NullPointerException. Важным требованием к новому языку является усиление статических гарантий. Это позволит обнаруживать больше ошибок на этапе компиляции и, таким образом, сокращать затраты на тестирование.

Скорость компиляции. Статические проверки упрощают программирование, но замедляют компиляцию, и здесь необходимо добиться определенного баланса. Опыт создания языков с мощной системой типов (наиболее ярким примером является Scala) показывает, что такой баланс найти непросто: компиляция зачастую становится неприемлемо долгой. Вообще, такая характеристика языка, как время компиляции проекта, может показаться второстепенной, однако в условиях промышленной разработки, когда объемы компилируемого кода очень велики, оказывается, что этот фактор весьма важен ведь пока код компилируется, программист зачастую не может продолжать работу. В частности, быстрая компиляция является одним из важных преимуществ Java по сравнению с C++, и Kotlin должен это преимущество сохранить.

Лаконичность. Известно, что программисты зачастую тратят больше времени на чтение кода, чем на его написание, поэтому важно, чтобы конструкции, доступные в языке программирования,

позволяли писать программы кратко и понятно. Java считается многословным языком (ceremony language – «церемонный язык»), и задача Kotlin – улучшить ситуацию в этом смысле. К сожалению, строгие методы оценивания языков с точки зрения их лаконичности развиты довольно слабо, но есть косвенные критерии; один из них – возможность создания библиотек, работа с которыми близка к использованию *предметно-ориентированных языков* (Domain-Specific Language, DSL). Для создания таких библиотек необходима определенная гибкость синтаксиса в совокупности с *конструкциями высших порядков*; наиболее распространены функции высших порядков, то есть функции, принимающие другие функции в качестве параметров.

Доступность для изучения. Сложные статические проверки, гибкий синтаксис и конструкции высших порядков усложняют язык и затрудняют его изучение, поэтому необходимо в известной степени ограничивать набор поддерживаемых возможностей, чтобы язык был доступен для изучения. При разработке Kotlin учитывался опыт Scala и других современных языков, и слишком сложные концепции в язык не включались.

Инструментальная поддержка. Современные программисты активно используют различные автоматизированные инструменты, центральное место среди которых занимают *интегрированные среды разработки* (Integrated Development Environment, IDE). Десятилетний опыт, накопленный в компании JetBrains, показывает, что определенные свойства языка могут существенно затруднять инструментальную поддержку. При разработке Kotlin учитывается этот факт и создается IDE одновременно с компилятором [2].

ВЫВОД

За счет хорошей совместимости с java и возможности заменять старый код постепенно, в будущем Kotlin мог бы стать хорошей заменой java в больших проектах и удобным инструментом для создания небольших проектов с перспективой их развития. Простота языка и его гибкость дает разработчику больше возможностей для написания быстрого, но качественного кода. Проблем пока много, но разработчики языка активно с ними борются [4].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Язык Kotlin для платформы Java»/ Способ доступа: URL: http://jeeconf.com/archive/jeeconf-012/materials/kotlin/

2. Журнал «Открытые системы»,№9 2011, Способ доступа: URL: http://www.osp.ru/os/2011/09/13011550/h

3. «Kotlin – новый язык программирования от JetBrains» Способ доступа: URL: http://habrahabr.ru/post/124494/

4. «О языке Kotlin для Java-программистов» Способ доступа: URL: http://habrahabr.ru/post/150104/

5. Bloch J. Effective Java. Second Edition. – Prentice Hall, 2008.