

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

ПРОГРАММИРОВАНИЕ НА JAVA

Учебно-методическое пособие для вузов

Составитель
Ю.А. Крыжановская

Издательско-полиграфический центр
Воронежского государственного университета
2012

Содержание

Введение	4
Основные понятия.....	4
История Java	5
Особенности	6
Инструменты программирования	8
Основные категории программ.....	9
Лабораторная работа № 1. Приложение «Hello world!»	10
Лексика языка	11
Типы	13
Имена	14
Переменные.....	14
Модификаторы доступа.....	14
Лабораторная работа № 2. Использование операторов	15
Управление выполнением программы.....	18
Лабораторная работа № 3. Передача управления.....	21
Строки.....	23
Лабораторная работа № 4. Сравнение и соединение строк.....	23
Массивы.....	24
Исключения.....	25
Лабораторная работа № 5. Массивы и исключения	27
Пакеты	29
Объявление класса [1–9].....	29
Интерфейсы [1–9].....	32
Лабораторная работа № 7. Использование пакета JAVA.AWT	33
Лабораторная работа № 8. Пакеты java.io и java.lang [7–9]	41
Лабораторная работа № 9. Пакет java.util	43
Задания для самостоятельного выполнения.....	51
Список литературы	52

нако сервер не обязательно «знает», каков тип компьютера у клиента. В языке Java присутствуют средства для решения такого рода задач [2].

В 1995 году появился и новый «язык сценариев» Java Script, который является кроссплатформенным объектным языком сценариев для корпоративных сетей и Internet. Его код описывается в тексте HTML или подгружается из отдельных файлов (с расширением .js). Можно сказать, что в некотором смысле JavaScript является дополнением и Java, и HTML, позволяя создавать приложения, связывающие объекты и ресурсы на клиентской машине или на сервере.

Особенности

Кроссплатформенность

Довольно частой является ситуация, когда заказчикам требуется одинаковая функциональность на различных платформах при использовании процессоров с разной архитектурой, что приводит к необходимости переноса приложений. Эта задача не нова, однако далеко не во всех случаях получается осуществить перенос корректно, поскольку поддержка многих возможностей может существенно различаться. Для решения этой проблемы применяется механизм виртуальной машины. В данном случае для исполнения приложений, написанных с использованием Java, применяется среда Java Virtual Machine (JVM) [6]. Она же определяет многие свойства Java. Средства создания Java-приложений разработаны для различных платформ: Linux, Solaris, Windows, MacOS и т. д.

Объектная ориентированность

Этот язык изначально позиционировался как объектно-ориентированный, здесь практически все реализовано в виде объектов, включая потоки выполнения, потоки данных, работу с сетью, изображениями и пользовательским интерфейсом, обработку ошибок. Фактически приложение на Java представляет собой набор классов, описывающих новые типы объектов. Все принципы объектно-ориентированного программирования (ООП) используются здесь в полном объеме.

Особенностями объектной модели Java являются:

- Отказ от множественного наследования, которое могло излишне усложнить и запутать программу; здесь используется специальный тип «интерфейс».
- Наличие строгой типизации приводит к тому, что любые переменные и выражения имеют известный на момент компиляции тип, а это, в свою

очередь, упрощает выявление проблем. Однако поиск исключительных ситуаций (о них будет рассказано далее) во время исполнения программы требует сложного тестирования, то есть в данном случае повышение надежности кода будет достигаться за счет дополнительных усилий при его написании [4].

– Сходство с C/C++, поскольку на момент создания Java C++ считался основным инструментом разработки и незнакомый синтаксис мог осложнить внедрение, однако модель и идеология Java были построены в соответствии с поставленными целями (что в ряде случаев позволяет говорить о том, что Java – упрощенный вариант C++).

– Легкость в освоении и разработке, например за счет работы с памятью: в Java изначально был предусмотрен механизм автоматической сборки мусора. Кроме того, начиная с самой первой версии, в Java присутствует возможность создания многопоточных приложений. За счет использования многопоточности в интерактивном графическом приложении удастся достичь высокой производительности, что особенно актуально в распределенных приложениях, когда процессы сетевого обмена могут идти одновременно и асинхронно. При этом программа продолжает реагировать на ввод информации пользователем без неприятных задержек. Многопоточность поддерживается на уровне языка, а системные библиотеки могут быть использованы в такого рода приложениях. Приложение легко сопровождается и модифицируется. Система обеспечивает динамическую сборку программы, т. е. классы подгружаются по мере необходимости с любой точки сети, что позволяет сделать внесение изменений в приложения прозрачным для пользователя.

Сокращение цикла разработки приложений

Сокращение происходит за счет того, что система построена на основе интерпретатора. Исходный код программы на Java представляет собой текстовые файлы, создаваемые в текстовом редакторе или специализированном средстве разработки и имеющие расширение .java. Они подаются на вход Java-компилятора, транслирующего их в байт-код. Этот набор инструкций поддерживается JVM и является неотъемлемой частью платформы Java. Результат работы компилятора сохраняется в бинарных файлах с расширением .class. Java-приложение, состоящее из таких файлов, подается на вход виртуальной машины для исполнения. Первоначально байт-код каждый раз интерпретировался, что значительно замедляло работу приложений. Затем была предложена схема, называемая JIT-компиляцией (Just-In-Time), при применении которой для инструкции или набора инструкций в первый раз происходит компиляция соответствующего байт-кода с сохра-

нением скомпилированного кода в буфере, а затем содержимое буфера используется при повторном вызове. Несколько отличается механизм работы оптимизирующих JIT-компиляторов: так как компиляция инструкции обычно идет гораздо дольше ее интерпретации, время ее выполнения в первый раз при наличии JIT-компиляции может быть заметно больше, чем при чистой интерпретации, а, значит, выгоднее сначала запустить процесс интерпретации, а параллельно в фоновом режиме компилировать инструкцию. После окончания компиляции при последующих вызовах инструкции будет исполняться ее скомпилированный код, а до этого момента все вызовы будут интерпретироваться. Разработанная Sun виртуальная машина HotSpot осуществляет JIT-компиляцию только тех участков байт-кода, которые критичны к времени выполнения программы. При этом по ходу работы программы происходит оптимизация [7].

Безопасность

Вопрос безопасности, пожалуй, является наиболее важным. Поэтому при работе виртуальной машины применяется комплекс мер, связанных с работой с памятью и отсечением опасного кода на каждом этапе работы. При этом анализируется, подчиняются ли class-файлы общим правилам безопасности Java или были созданы злоумышленниками с помощью каких-то других средств, что позволяет интерпретатору в дальнейшем проверить каждое действие на допустимость. Следует отметить, что возможности классов, загруженных с локального диска или по сети, существенно различаются. Кроме этого механизм подписания апплетов и других приложений, загружаемых по сети, позволяет клиенту либо отказаться от работы с приложениями ненадежных производителей, либо сразу увидеть, что в программу внесены неавторизованные изменения, а специальный сертификат должен гарантировать, что пользователь получил код именно в том виде, в каком его выпустил производитель.

Инструменты программирования

Существует достаточно много сред, среди которых отметим:

- JDK (Sun Microsystems);
- NetBeans (Sun Microsystems);
- Eclipse.

Что касается первой среды – JDK (Java Developer Kit), то она представляет собой бесплатно распространяемое программное обеспечение от компании Sun, которая в настоящий момент перешла под руководство ком-