Основы языка Java

Java – это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems, впоследствии приобретённой компанией Oracle. Java задумывался как универсальный язык программирования, который можно применять для различного рода задач. Дата официального выпуска Java – 23 мая 1995 года. С этого момента было издано множество различных версий языка, текущей версией является Java 18, вышедшая в марте 2022 года. На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования. К настоящему времени Java превратилась из универсального языка программирования в целую платформу, которая объединяет различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов. Кроме того, язык Java активно применяется для создания программного обеспечения для множества устройств: настольных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники.

Программы Java можно разделить на несколько основных категорий:

- приложение – аналог стандартной прикладной программы;
- серверное приложение – программа, предназначенная для выполнения на стороне сервера;
- апплет – специализированная программа с ограниченными возможностями, работающая под управлением браузера;
- сервлет – специализированная программа с ограниченными возможностями, работающая в сети Интернет на стороне сервера;
- мидлет – программа, запускаемая в мобильной среде;
- библиотека классов.

В настоящее время существует три редакции языка Java:

- Java SE (Java Platform Standard Edition) - это стандартная редакция Java, которая используется для разработки простых Java приложений;
- Java EE (Java Platform Enterprise Edition) - это редакция Java для разработки распределенных приложений масштаба предприятий;
- Java ME (Java Platform Micro Edition) - это редакция Java для разработки приложений для микрокомпьютеров.

Исходный код любой программы на языке Java представляется обычными текстовыми файлами, которые могут быть созданы в любом текстовом редакторе или специализированном средстве разработки и имеют расширение .java. Эти файлы подаются на вход Java-компилятора, который транслирует их в специальный байт-код, независимый от платформы. Результат работы компилятора сохраняется в бинарных файлах с расширением .class. Приложение, состоящее из таких файлов, подается на вход *виртуальной машине Java* (Java Virtual Machine, JVM), которая сама является приложением, и она начинает их исполнять – интерпретировать. Подобная

архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код. JVM обеспечивает интерфейс, который не зависит от операционной системы и аппаратных средств. Эта независимость дает Java-программам возможность выполняться на любом устройстве без необходимости внесения каких-либо изменений. Важной частью JVM является Just-in-time Compiler (JIT), который оптимизирует байт-код, уменьшая общее время, необходимое для компиляции байт-кода в машинный код. Минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, называется *средой исполнения Java* (Java Runtime Environment, JRE). Она состоит из виртуальной машины Java и библиотек Java-классов и не содержит компилятора и других средств разработки. Таким образом, при помощи JRE можно лишь запускать на выполнение Java-приложения, но не разрабатывать их. Для разработки приложений необходим *комплект разработчика Java* (Java Development Kit, JDK), который, помимо JRE, включает в себя компилятор, стандартные библиотеки классов Java, различные утилиты, документацию, примеры программного кода.

Особенности языка Java

При выпуске Java компания Sun Microsystems заложила пять парадигм потенциального успеха, которых разработчики языка придерживаются и в настоящее время:

- простота, объектная ориентированность и понятность;
- надёжность и безопасность;
- переносимость и независимость от платформы;
- высокая производительность;
- интрепретируемость, поточность и динамичность.

Разрабатывая новую технологию, авторы языка Java опирались на широко распространенный язык программирования C++. При этом они позаботились избавить программистов от наиболее распространенных ошибок, которые порой допускают даже опытные разработчики. Первое место среди таких ошибок занимает работа с памятью. Резервирование памяти в Java определяется исключительно JVM, а не компилятором или разработчиком – разработчик может лишь указать, что он хочет создать еще один новый объект. По этой причине указатели по физическим адресам отсутствуют принципиально. Для освобождения неиспользуемой памяти в Java был введен механизм автоматической сборки мусора (garbage collector). Сборщик мусора - это фоновый поток исполнения, который регулярно просматривает существующие объекты, и удаляет уже не нужные. Из программы никак нельзя повлиять на работу сборщика мусора - можно только с помощью стандартной функции явно инициировать его очередной проход. Таким образом, программист должен следить

лишь за тем, чтобы не оставалось ссылок на ненужные объекты, что существенно упрощает разработку программ. Также создатели языка Java отказались от множественного наследования – было решено, что оно слишком усложняет и запутывает программы. В языке используется альтернативный подход - специальный тип «интерфейс». При этом в Java, как и в C++, применяется строгая типизация – любая переменная и любое выражение имеет тип, известный уже на момент компиляции. Такой подход применен для упрощения выявления проблем, ведь компилятор сразу сообщает об ошибках и указывает их расположение в коде. Все эти изменения позволили обеспечить надёжность языка Java и его широкое распространение.

Лексика языка Java

Компилятор, анализируя текст программы, разделяет его на следующие части:

- лексемы – минимально значимые единицы языка;
- пробелы – символы, разбивающие текст программы на лексемы;
- комментарии – фрагменты текста, которые используются для ввода пояснений к программе.

С точки зрения компилятора, а точнее его части, отвечающей за лексический разбор, основная роль пробелов и комментариев - служить разделителями между лексемами, причем сами разделители далее отбрасываются и не влияют на компилированный код. В языке Java выделяют следующие виды лексем: идентификаторы, ключевые слова, литералы, разделители и операторы.

Идентификаторы - это имена, которые даются различным элементам языка для упрощения доступа к ним. Идентификатор состоит из букв и цифр и не может начинаться с цифры, длина идентификатора не ограничена. Символы, используемые в идентификаторах, включают в себя прописные и строчные буквы, а также знаки подчеркивания _ и доллара $. Знак доллара используется только при автоматической генерации кода (чтобы исключить случайное совпадение имен), либо при использовании каких-либо старых библиотек, в которых допускались имена с этим символом. Java-цифры включают в себя цифры 0-9. Для идентификаторов не допускаются совпадения с зарезервированными словами – это ключевые слова, булевские литералы true и false, null-литерал и специальный идентификатор var.

Ключевые слова, вместе с синтаксисом операторов и разделителей, образуют основу языка Java. В Java определено 53 ключевых слова: abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, exports, extends, final, finally, float, for, goto, if, impements, import, instanceof, int, interface, long, module, native, new, package, private, protected, public, requires, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while. Слова const и goto в языке Java не используются, но являются ключевыми – это сделано для того, чтобы компилятор мог правильно отреагировать на использование

этих распространенных в других языках слов. Напротив, литералы true, false и null часто считают ключевыми, однако это именно литералы.

Литералы позволяют задать в программе значения для числовых, символьных и строковых выражений, а также null-литералов. Всего в Java определены следующие виды литералов: целочисленный, дробный, логический, символьный, строковый и null-literal.

Целочисленные литералы позволяют задавать целочисленные значения в десятеричном, восьмеричном, двоичном и шестнадцатеричном виде. Десятеричный формат традиционен и ничем не отличается от правил, принятых в других языках. Запись восьмеричных чисел начинается с 0, разрешается использовать цифры 0-7. Запись двоичных чисел начинается с 0b или 0B (цифра 0 и латинская буква B в произвольном регистре), и разрешается использовать только цифры 0 и 1. Запись шестнадцатеричных чисел начинается с 0x или 0X (цифра 0 и латинская буква X в произвольном регистре), разрешается использовать цифры 0-9 и латинские буквы A-F в произвольном регистре. По умолчанию целочисленный литерал имеет тип int – чтобы ввести литерал типа long, необходимо в конце поставить латинскую букву L в произвольном регистре.

Дробные литералы представляют собой числа с плавающей десятичной точкой. Дробный литерал состоит из следующих составных частей: целая часть, десятичная точка, дробная часть, показатель степени (состоит из латинской буквы E в произвольном регистре и целого числа с опциональным знаком + или -) и окончание-указатель типа. Целая и дробная части записываются десятичными цифрами, а указатель типа имеет два возможных значения: латинская буква D (для типа double) или F (для типа float) в произвольном регистре.

Логические литералы представлены двумя значениям типа boolean: true (истина) и false (ложь).

Символьные литералы могут быть представлены в виде символа, заключенного в одинарные кавычки. Также позволяется специальная запись для описания символа через его код, записанный в восьмеричной или шестнадцатеричной форме. Для ввода значений в восьмеричной форме служит символ обратной косой черты, за которым следует трехзначное число, состоящее из цифр 0-7. Для ввода значений в шестнадцатеричной форме служит символ обратной косой черты, за которым следует латинская буква u в нижнем регистре и четырехзначное число, состоящее из цифр 0-9 и латинских букв A-F в произвольном регистре.

Строковые литералы могут быть представлены в виде последовательности символов, заключённой в двойные кавычки. Длина строковых литералов не ограничена. Управляющие символы и восьмеричная или шестнадцатеричная форма записи, определенные для символьных литералов, действуют точно так же и в строковых литералах. Строки могут располагаться только на одной строке исходного

кода – нельзя открывающую кавычку поставить на одной строке, а закрывающую – на следующей строке.

Null-литерал может принимать всего одно значение – null. Это литерал ссылочного типа, причем эта ссылка не ссылается ни на один объект.

Разделителями в языке Java называются следующие символы:

- ( ) – круглые скобки, используются для передачи списков параметров в определениях и вызовах методов, для обозначения операции приведения типов и предшествования операторов в выражениях, используемых в управляющих операторах;
- { } – фигурные скобки, используются для указания значений автоматически инициализируемых массивов, а также для определения блоков кода, классов, методов и локальных областей действия;
- [ ] – квадратные скобки, используются для объявления типов массивов, а также при обращении к элементам массивов;
- ; – точка с запятой, завершает операторы;
- , – запятая, разделяет последовательный ряд идентификаторов в объявлениях переменных;
- . – точка, используется для отделения имен пакетов от подпакетов и классов, а также для отделения переменной или метода от ссылочной переменной.

Операторы языка Java используются в различных операциях - арифметических, логических, битовых, операции сравнения, присваивания. Следующие 37 лексем являются операторами языка Java: = > < ! ~ ? : == <= >= != && || ++ -- + - * / & | ^ % << >> >>> += -= *= /= &= |= ^= %= <<= >>= >>>=

Пробелами называют все символы, разбивающие текст программы на лексемы. Это как непосредственно символ пробела, так и знаки табуляции и перевода строки. Они используются для разделения лексем, а также для оформления кода.

Комментарии не влияют на результирующий бинарный код и используются только для ввода пояснений к программе. Комментарии в Java бывают двух видов:

- строчные – начинаются с комбинации символов // и длятся до конца текущей строки;
- блочные – располагаются между комбинациями символов /* и */, могут занимать произвольное количество строк.

Кроме того, существует особый вид блочного комментария - комментарий разработчика. Он применяется для автоматического создания документации кода и записывается так же, как и блочный. Единственное различие в начальной комбинации символов – комментарий разработчика необходимо начинать с комбинации символов /**.