Лекция 1

Что такое Java?

Java широко известна как новейший объектно-ориентированный язык, разработанным фирмой **Sun Microsystems** (сокращенно, **Sun**). Легкий в изучении и позволяющий создавать программы, которые могут исполняться на любой платформе без каких-либо доработок (**кросс-платформенность**).

Платформа — это программно-аппаратная среда, в которой происходит выполнение приложений, комбинация аппаратного обеспечения и операционной системы.

Объектно-ориентированный язык программирования — язык, построенный на принципах объектно-ориентированного программирования. В основе концепции объектно-ориентированного программирования лежит понятие объекта — некоей субстанции, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).

Например, объект человек может иметь поля имя, фамилия и методы есть и спать. Соответственно, в программе можем использовать операторы Человек.Имя:="Иван" и Человек.Есть(пища).

Программисты могут добавить к этому описанию, что язык похож на упрощенный C или C++ с добавлением **garbage collector'a** - автоматического сборщика "мусора" (Этот механизм автоматически подсчитывает количество ссылок на каждый объект Java. Когда на объект больше не указывает ни одна ссылка, он удаляется из памяти, освобождая ресурсы для программы).

Также известно, что Java ориентирована на Интернет, и самое ее распространенное применение - небольшие программы, называющиеся **апплеты**, которые запускаются в браузере и являются частью HTML-страниц.

Основные достоинства языка

- Наибольшая среди всех языков программирования степень переносимости программ.
- Мощные стандартные библиотеки.
- Встроенная поддержка работы в сетях (как локальных, так и Internet/Intranet).

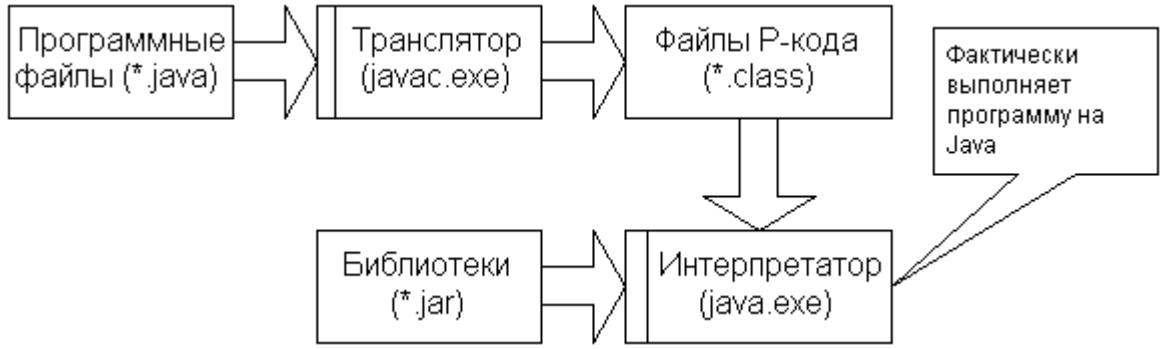
Основные недостатки

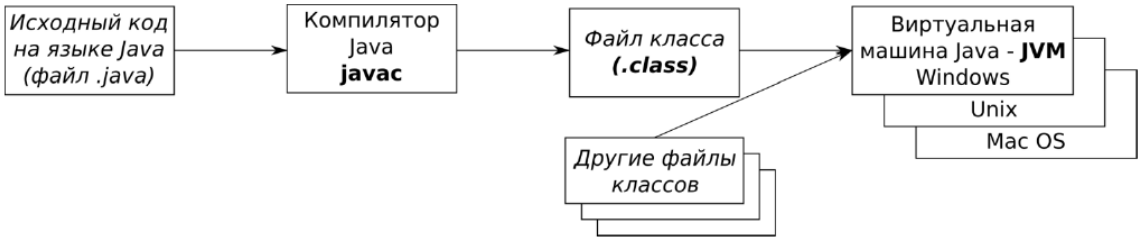
- Низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти (ОП).
- Большой объем стандартных библиотек и технологий создает сложности в изучении языка.
- Постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение.

Основные особенности

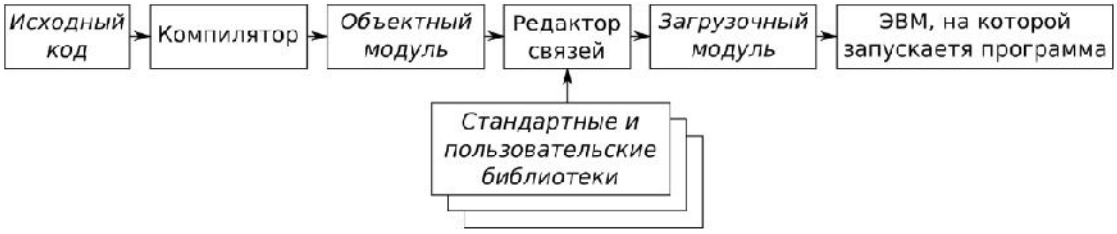
- Java является полностью объектно-ориентированным языком. Например, C++ тоже является объектно-ориентированным, но в нем есть возможность писать программы не в объектно-ориентированном стиле, а в Java так нельзя.

- Реализован с использованием интерпретации Р-кода (байт-кода). Т.е. программа сначала транслируется в машинонезависимый Р-код, а потом интерпретируется некоторой программой-интерпретатором (виртуальная Java-машина, JVM). Ее применение необходимо для обеспечения кросс-платформенности, а также для безопасности, однако создает определенные проблемы в вопросе производительности.
- Язык Java является компилируемым и интерпретируемым.
  Компиляция — преобразование программы, написанной на языке программирования, в программу на другом языке (как правило, машинном) путём последовательной замены операторов исходной программы эквивалентной последовательностью команд машинного языка. Вычислительная машина выполняет скомпилированную программу вместо исходной.
  Интерпретация — пооператорное выполнение исходной программы с помощью программы-интерпретатора. Интерпретатор анализирует каждый оператор исходной программы и непосредственно реализует эквивалентную последовательность команд машинного языка.
  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.
  Программа транслируется в байт-код компилятором **javac.exe**.
  Компилятор Java не генерирует машинные команды для процессора ЭВМ. Он создаёт промежуточный код, так называемый ***байт-код*** для виртуальной машины Java (**J**ava **V**irtual **M**achine).
  JVM — это виртуальный компьютер, размещенный в оперативной памяти реальной машины. Байт-код — программа, созданная компилятором на языке JVM. *В основе JVM лежит интерпретатор языка Java (точнее - байт-кода Java)*.
  Оттранслированная в байт-код программа имеет расширение **class** . Для запуска программы нужно вызвать интерпретатор **java.exe** , указав в параметрах вызова, какую программу ему следует выполнять. Кроме того, ему нужно указать, какие библиотеки нужно использовать при выполнении программы. Библиотеки размещены в файлах с расширением **jar** (в предыдущих версиях SDK использовались файлы ***.zip** и некоторые библиотеки все еще в таких файлах).





Процесс, необходимый для создания работающего приложения называется **жизненным циклом программы**. Для программ на Java он отличается от жизненного цикла программ на других языках программирования. Типичная картина жизненного цикла для большинства языков программирования выглядит примерно так.



Понятие среды выполнения Java

Среда выполнения **J**ava **R**untime **E**nvironment (**JRE**) поставляется в составе JDK. Виртуальная машина Java (**JVM**) является основой JRE. Функции JVM выполняет модуль ***java.exe*** — интерпретатор языка Java.

С 11 марта 1997 года компания Sun начала предлагать Java Runtime Environment, JRE (среду выполнения Java). По сути дела это минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки. Если пользователь хочет только запускать программы, это именно то, что ему нужно

Версии Java

Итак, впервые Java была объявлена 23 мая 1995 года. Основными продуктами, доступными на тот момент в виде бета-версий, были:

- Java language specification, JLS, спецификация языка Java (описывающая лексику, типы данных, основные конструкции, и т.д.);
- спецификация JVM;
- Java Development Kit, JDK - средство разработчика, состоящее в основном из утилит, стандартных библиотек классов и демонстрационных примеров.

Спецификация языка была составлена настолько удачно, что практически без изменений используется по сей день. Конечно, было внесено большое количество уточнений, более подробных описаний, были добавлены и некоторые новые возможности (например, объявление внутренних классов), однако основные концепции остаются неизменными.

Спецификация JVM предназначена в первую очередь для создателей виртуальных машин, а потому практически не используется Java-программистами.

JDK долгое время было базовым средством разработки приложений. Оно не содержит никаких текстовых редакторов, а оперирует только с уже существующими java-файлами. Компилятор представлен утилитой javac (java compiler) . Виртуальная машина реализована программой java. Для тестовых запусков апплетов есть специальная утилита appletviewer. Наконец, для автоматической генерации документации на основе исходного кода прилагается средство javadoc.

Первая версия содержала всего 8 стандартных библиотек:

- java.lang - базовая классы, необходимые для работы любого приложения (название - сокращение от language);
- java.util - многие полезные вспомогательные классы;
- java.applet - классы для создания апплетов;
- java.awt, java.awt.peer - библиотека для создания графического интерфейса пользователя (GUI), называется Abstract Window Toolkit, AWT.
- java.awt.image - дополнительные классы для работы с изображениями;
- java.io - работа с потоками данных (streams) и с файлами;
- java.net - работа с сетью.

Как видно, все библиотеки начинаются с java, именно они являются стандартными. Все остальные (начинающиеся с com, org и др.) могут меняться в любой версии без поддержки совместимости.

Обозначение версии состоит из трех цифр.

Первой пока всегда стоит 1. Это означает, что поддерживается полная совместимость между всеми версиями 1.x.x. То есть, программа, написанная на более старом JDK, всегда успешно выполнится на более новом. По возможности соблюдается и обратная совместимость - если программа откомпилирована более новым JDK, а никакие новые библиотеки не использовались, то в большинстве случаев старые виртуальные машины смогут выполнить такой код.

Вторая цифра изменилась от 0 до 4 (последняя на данный момент, версии 1.0−1.4). В каждой версии происходило существенное расширение стандартных библиотек (212, 504, 1781, 2130 и 2738 - количество классов и интерфейсов с 1.0 по 1.4), а также добавлялись некоторые новые возможности в сам язык. Менялись и утилиты, входящие в JDK.

Наконец, третья цифра означает развитие одной версии – нлмео версии поддержки платформы. В языке или библиотеках ничего не меняется, лишь устраняются ошибки, производится оптимизация, могут меняться (добавляться) аргументы утилит.

Спецификация Java 5.0 была выпущена в сентябре 2004 года. С этой версии изменена официальная индексация, вместо Java 1.5 правильнее называть Java 5.0. Внутренняя же индексация Sun осталась прежней — 1.x. Минорные изменения теперь включаются без изменения индексации, для этого используется слово «Update» или буква «u», например Java Development Kit 5.0

Update 22. Предполагается, что в обновления могут входить как исправления ошибок так и небольшие добавления в API, JVM.

Релиз версии Java 6 состоялся 11 декабря 2006 года. Изменена официальная индексация — вместо ожидаемой 6.0 версия значится как 6.

Релиз версии Java 7 состоялся 28 июля 2011 года. В этой версии, получившей название Java Standard Edition 7 (Java Platform, Standard Edition 7), помимо исправления большого количества ошибок было представлено несколько новшеств. Так, например, в качестве эталонной реализации Java Standard Edition 7 использован не проприетарный пакет JDK, а его открытая реализация OpenJDK.

Релиз версии Java 8 состоялся 19 марта 2014 года

Хотя с развитием версии 1.x ничего не удаляется, конечно, какие-то функции или классы устаревают. Они объявляются **deprecated**, и хотя они будут поддерживаться до объявления 2.0, пользоваться ими не рекомендуется.

Типы Java-приложений

Язык Java можно использовать для разработки программ следующих типов.

Автономное (самостоятельное) приложение – application

Для запуска приложения необходимо загрузить сначала реализацию виртуальной машины Java (Java Virtual Machine — **JVM**). В состав JDK включен интерпретатор языка Java — модуль Java.exe, в котором реализованы функции JVM.

Апплет – applet

Апплетом называется мини-приложение Java, работающее под управлением Web-браузера, в который встроена виртуальная Java-машина (JVM). Все наиболее популярные браузеры являются Java-совместимыми.

Технология апплетов поддерживается версией Java-платформы ***J2SE*** (***Java 2 S**tandard **E**dition*).

Комбинированное приложение

Комбинированное приложение Java может работать и как автономное приложение, и как апплет.

В приложениях может использоваться либо интерфейс командной строки, либо графический интерфейс пользователя (Graphical User Interface — GUI). В апплетах и комбинированных приложениях можно применять только GUI.

Сервлет – servlet

Сервлет является мини-приложением Java, выполняющимся, в отличие от апплета, на стороне Web-сервера. Сервлеты служат для формирования динамических HTML-страниц.

В настоящее время технология сервлетов применяется совместно с технологией ***JSP*** (***Java S**erver **P**ages* — серверных страниц Java). Страница JSP (HTML-код со специальной вставкой) обеспечивает автоматическое преобразование Java-кода в сервлет при первом обращении к ней клиента. Данные технологии поддерживаются версией Java-платформы ***J2EE*** (***Java 2 E**nterprise **E**dition*).

Мидлет – midlet

Java-приложение для ***M**icro **I**nformation **D**evices*, в том числе мобильных телефонов. Данная технология поддерживается версией Java-платформы ***J2ME*** (***J**ava **2** **M**icro **E**dition*).

Особенности

В современных объектно−ориентированных языках используются методы:

Наследование. Создание нового класса объектов путём добавления новых элементов (методов). В данный момент ОО языки позволяют выполнять множественное наследование, то есть объединять в одном классе возможности нескольких других классов.

Инкапсуляция. Сокрытие деталей реализации, которое позволяет вносить изменения в части программы безболезненно для других её частей, что существенно упрощает сопровождение и модификацию ПО.

Полиморфизм. При полиморфизме некоторые части (методы) родительского класса заменяются новыми, реализующими специфические для данного потомка действия. Таким образом, интерфейс классов остаётся прежним, а реализация методов с одинаковым названием и набором параметров различается. С полиморфизмом тесно связано позднее связывание.

Типизация. Позволяет устранить многие ошибки на момент компиляции, операции проводятся только над объектами подходящего типа.