МОДЕЛЬ И ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA

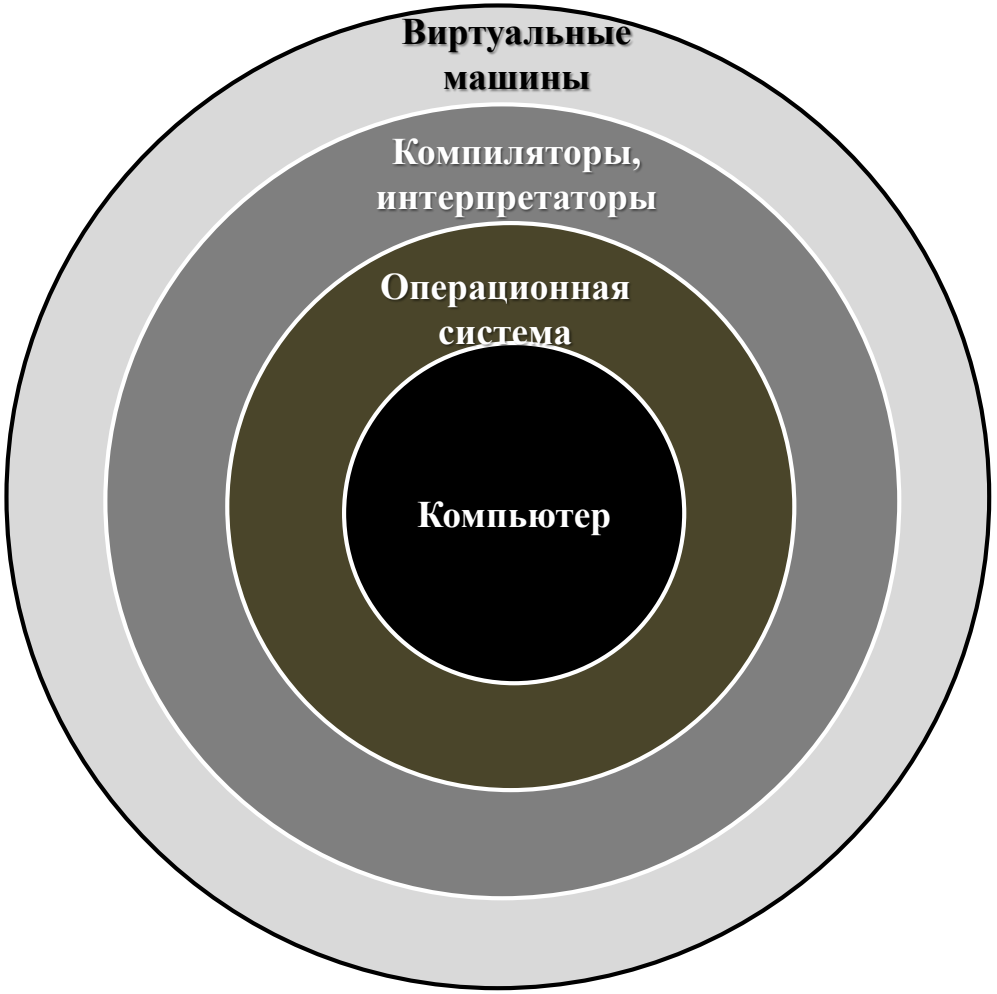
Кофе на работе – это напиток,
который пьют, когда хотят есть.
А. Калинин

Языки программирования

Содержание

- Методы реализации языков программирования
  - Интерфейс виртуальных компьютеров
  - Компиляция
  - Чистая интерпретация
  - Смешанный подход
- Модель Java
  - Проблемы эффективности и безопасности
  - Независимость модели от языка
- Обзор языка Java

Интерфейс виртуальных компьютеров



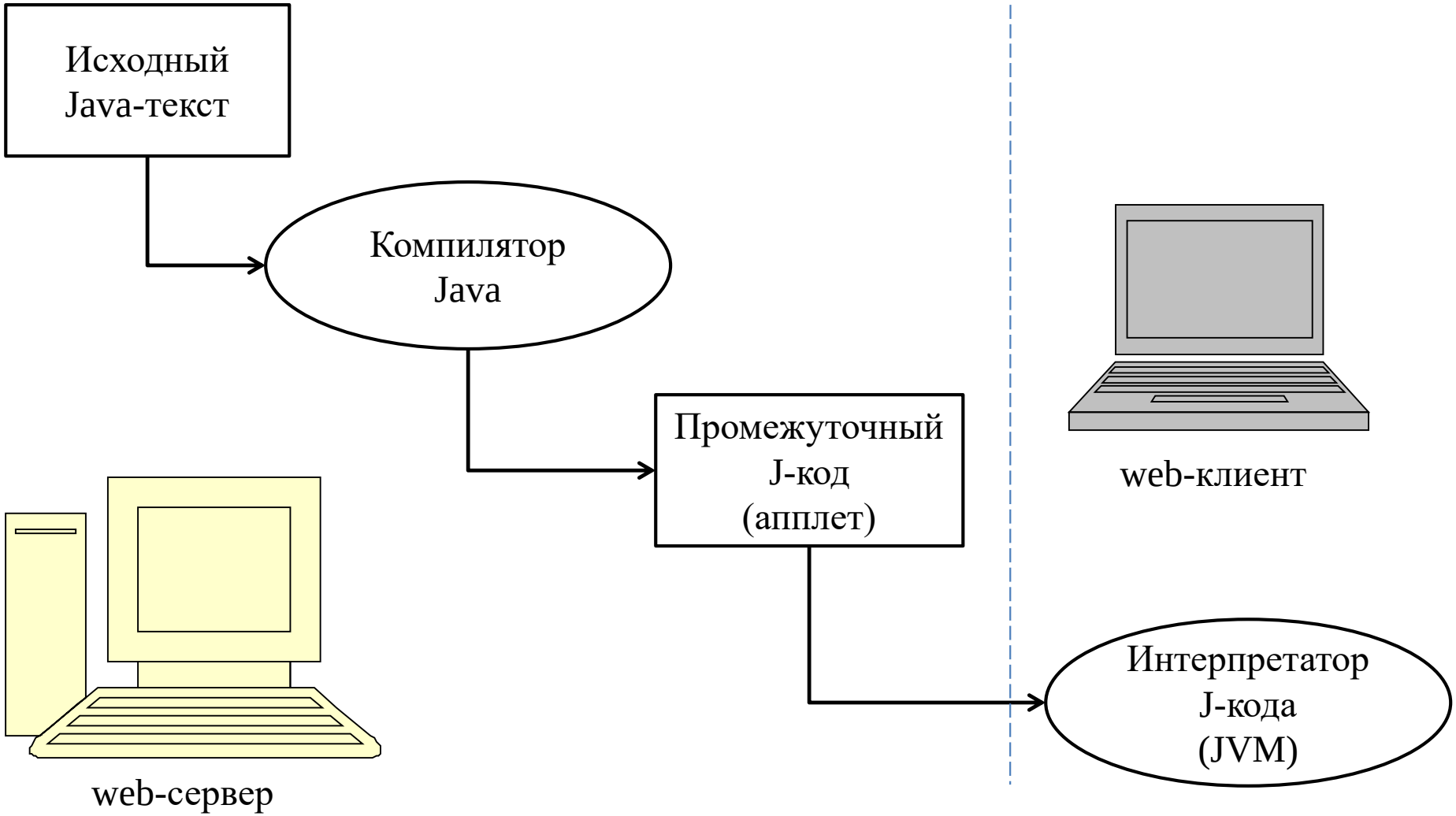
Компиляция, интерпретация

- *Компиляция* – преобразование исходного текста программы в машинный код.
  - Позволяет получить более эффективный код, чем исходный текст за счет оптимизации.
  - Применяется, как правило, для императивных языков со сложной структурой: Pascal, C, Ada и др.
- *Интерпретация* – непосредственное исполнение исходного текста программы .
  - Легкость реализации операций отладки на уровне исходной программы.
  - Более медленна и требует больше памяти, чем компиляция.
  - Применяется, как правило, для языков с простой структурой: декларативные LISP, APL, языки сценариев ОС.

Компиляция + интерпретация

- *Смешанная система реализации языка* предполагает два этапа:
  1. компиляция исходного текста программы в промежуточный язык, и затем
  2. интерпретация программы на промежуточном языке.
- По сравнению с чистой интерпретацией в смешанной реализации этап компиляции позволяет
  - уменьшить ошибки в интерпретируемом коде
  - получить компактный и мобильный интерпретируемый код (этапы смешанной реализации могут выполняться в разных аппаратно-программных средах).
- Примеры: Perl, Java.

Java-модель


Исходный
Java-текст
Компилятор
Java
Промежуточный
J-код
(апплет)
Интерпретатор
J-кода
(JVM)
web-клиент
web-сервер

Проблема эффективности

- Интерпретация абстрактного программного кода выполняется быстро для относительно простых программ, выполняющихся на мощных рабочих станциях.
- Одно из решений – замена JVM на компилятор с J-кода в машинный код компьютера-клиента.
  - Компиляция может выполняться "на лету" – одновременно или почти одновременно с приемом J-кода. В худшем случае скорость работы возрастет при *втором* выполнении апплета.
  - Это частичное решение – не очень практично встраивать в сложный оптимизирующий компилятор внутрь браузера для каждой аппаратно-программной среды.

Проблема безопасности

- Как контролировать действия апплета во время его интерпретации (format c:, вирусы и др.)?
- Возможные стратегии
  - Строгий контроль соответствия типов.
    - Устранение случайного или преднамеренного повреждения данных, вызванного выходом за границы массива, повисшими указателями и др.
  - Проверка J-кода.
    - JVM проверяет, действительно ли поток байтов, полученных с удаленного компьютера, состоит из допустимых инструкций J-кода.
  - Ограничения на апплет.
    - Апплету не разрешается выполнять некоторые операции на получающем компьютере (запись и удаление файлов и др.).

Независимость модели от языка

- Java-модель может быть реализована на базе любого исходного языка программирования (не обязательно Java).
  - Существуют компиляторы, которые переводят исходный текст Ada в J-код.

Язык Java

- Похож на C++ (но отказывается от совместимости с языком C):
  - Элементарные типы данных, выражения, управляющие операторы.
  - Функции и параметры.
  - Классы, наследование, динамический полиморфизм.
  - Обработка исключений.
- Некоторые основные отличия:
  - Семантика ссылки
  - Инкапсуляция классов
  - Встроенные средства параллелизма
  - Библиотеки.

Семантика ссылки

- При объявлении переменной непримитивного типа память не выделяется; вместо этого выделяется неявный указатель.
  - Для реального выделения памяти необходим вызов конструктора
  - Разыменование ссылки не требуется.
- Пример:
  - int [] arr_java; // объявление указателя
  - arr_java=new int[MaxN]; // распределение указателя
  - int[] arr_java=new int[MaxN]; // объявление и распределение
  - for (i=0; i<MaxN; i++) arr_java[i]=i; // использование
  - arr_java=new int[N]; // повторное распределение (исходные данные становятся "мусором"; сборщик мусора внутри JVM)

Семантика ссылки

- Списковые структуры:
  - class TNode {
    TInfo info;
    TNode next;
    }
  - TNode listHead;
  - listHead=new TNode(10, listHead);
- Проверка равенства и присваивание
  - String s1=new String("Java");
    String s2=new String("Java");
  - if (s1==s2) System.out.printIn("Строки равны") else
    System.out.printIn("Строки НЕ равны");
  - if (s1.equals(s2)) System.out.printIn("Строки равны") else
    System.out.printIn("Строки НЕ равны");

Библиотеки Java

- "Размер" языка Java сокращен за счет расширения функциональности библиотек.
- Стандартные библиотеки Java:
  - java.lang обработка строк, матем. функции и др.
  - java.util системные интерфейсы
  - java.io стандартный ввод-вывод
- Java API
  - java.applet создание и выполнение апплетов
  - java.awt графический интерфейс пользователя
  - java.net размещение и пересылка данных по сети