Компиляция и исполнение Java приложений под капотом

Голов Павел

21 апреля 2019 г.

Статья на конкурс для сайта JavaRush.

1 Вступление

Всем привет! Сегодня я хотел бы поделиться знаниями о том, что происходит под капотом JVM (Java Virtual Machine) после того, как мы запускаем написанное Java приложение. В наше время существуют моднейшие среды разработки, которые помогают не думать о внутреннем устройстве JVM, компиляции и выполнении Java - кода, из за чего новые разработчики могут упустить эти важные аспекты. В то же время, на собеседованиях часто задают вопросы касательно этой темы, из-за чего я и решил написать статью.

2 Основная часть

2.1 Компиляция в байт-код

Начнем с теории. Когда мы пишем какое-либо приложение, мы создаем файл с расширением .java и помещаем в него код на языке программирования Java. Такой файл, содержащий код, понятный человеку, называется файлом с исходным кодом. После того, как файл с исходным кодом готов, нужно его выполнить! Но на данной стадии в нем содержится информация, понятная только человеку.

Java - мультиплатформенный язык программирования. Это значит, что программы, написанные на языке Java, можно выполнять на любой платформе, где установлена специальная исполняющая система Java. Такая система называется Java Virtual Machine (JVM). Для того, чтобы перевести программу из исходного кода в код, понятный JVM, нужно её скомпилировать. Код, понятный JVM называется байт-кодом и содержит набор инструкций, которые в дальнейшем будет исполнять виртуальная машина.

Для компиляции исходного кода в байт-код существует компилятор javac, входящий в поставку JDK (Java Development Kit). На вход компилятор принимает файл с расширением .java, содежащий исходный код программы, а на выходе выдает файл с расширением .class, содержащий байт-код, необходимый для исполнения программы виртуальной машиной.

После того, как программа была скомпилирована в байт код, она может быть выполнена с помощью виртуальной машины.

2.2 Пример компиляции

Предположим, что у нас есть простая программа, содержащаяся в файле Calculator.java, которая принимает 2 численных аргумента командной строки и печатает результат их сложения:

```
class Calculator {
        public static void main(String[] args){
                int a = Integer.valueOf(args[0]);
                int b = Integer.valueOf(args[1]);
```

```
                System.out.println(a + b);
        }
}
```

Для того, чтобы скомпилировать эту программу в байт код, воспользуемся компилятором javac в командной строке:

```
javac Calculator.java
```

После компиляции на выходе мы получаем файл с байт-кодом Calculator.class, который мы можем исполнить при помощи установленной на нашем компьютере java-машины при помощи команды java в командной строке:

```
java Calculator 1 2
```

Заметим, что после названия файла были указаны 2 аргумента командной строки - числа 1 и 2. После выполнения программы, в командной строке выведется число 3.

2.3 Исполнение программы виртуальной машиной

Итак, мы запустили написанную программу. Но что же происходит в момент запуска скомпилированной программы виртуальной машиной?

Для начала разберемся, что означают понятия компиляции и интерпретации кода.

Компиляция — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.

Интерпретация — пооператорный (покомандный, построчный) анализ, обработка и тут же выполнение исходной программы или запроса (в отличие от компиляции, при которой программа транслируется без её выполнения).

Язык java обладает как компилятором (javac), так и интерпретатором, в роли которого выступает виртуальная машина, которая построчно преобразует байт-код в машинный код и тут же его исполняет.

Таким образом, когда мы запускаем скомпилированную программу, виртуальная машина начинает её интерпретацию, то есть построчное преобразование байт-кода в машинный код, а так же его исполнение.

К сожалению, чистая интерпретация байт-кода является довольно долгим процессом и делает язык java медленным в сравнении с его конкурентами. Дабы избежать этого, был введен механизм, позволяющий ускорить интерпретацию байт-кода виртуальной машиной. Этот механизм называется Just-in-time компиляцией (JITC).

2.4 Just-in-time (JIT) компиляция

Простыми словами, механизм Just-In-Time компиляции заключается в следующем: если в программе присутствуют части кода, которые выполняются много раз, то их можно скомпилировать один раз в машинный код, чтобы в будущем ускорить их выполнение. После компиляции такой части программы в машинный код, при

каждом следующем вызове этой части программы виртуальная машина будет сразу выполнять скомпилированный машинный код, что естественно ускорит выполнение программы.

Ускорение работы программы достигается за счет увеличения потребления памяти (где-то же нам нужно хранить скомпилированный машинный код!) и за счет увеличения временных затрат на компиляцию во время исполнения программы.

3 Заключение