Java Memory Model

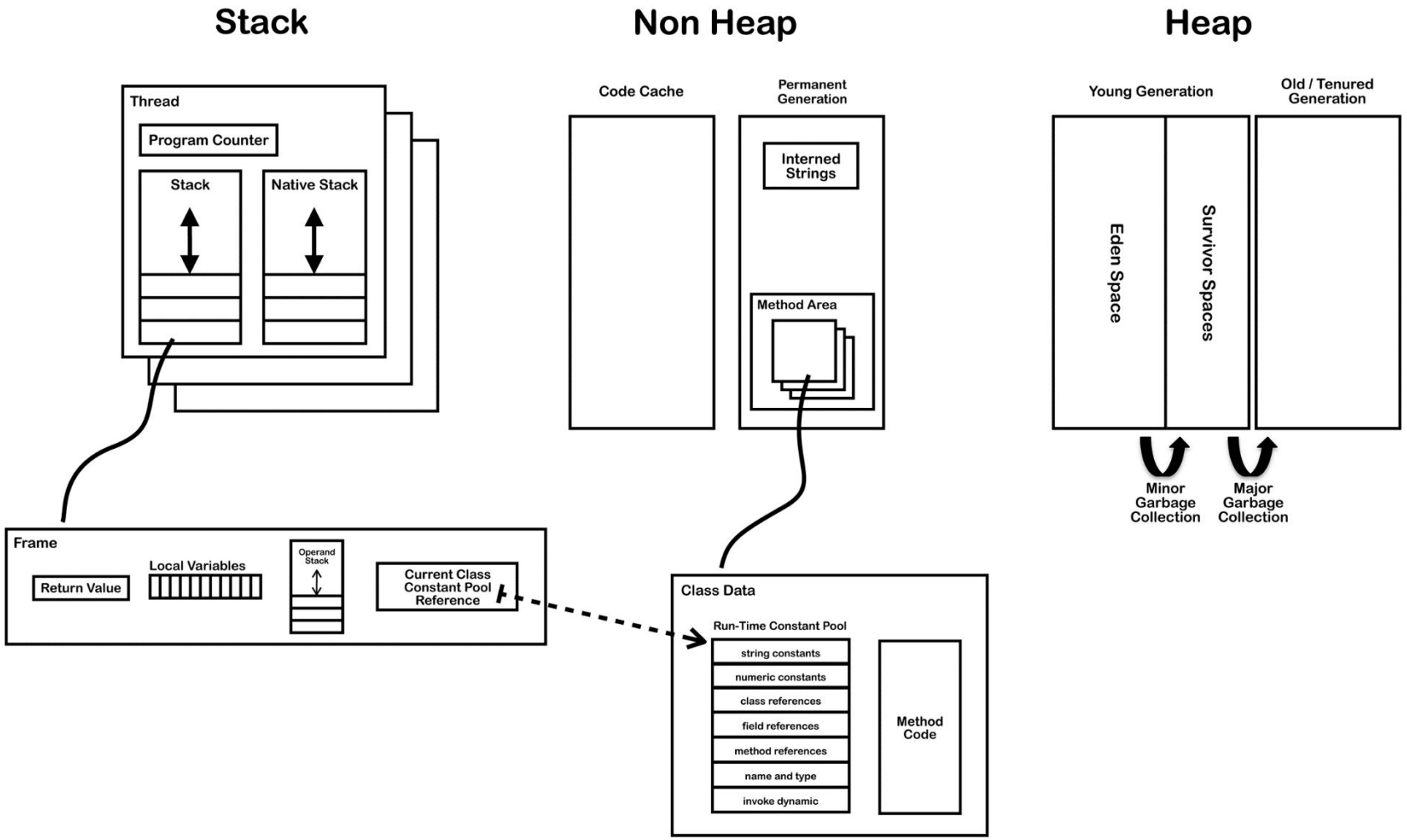
Части Java Memory

- Stack / Стэк
- Heap / “Куча”
- ~~Permanent generation (PermGen)~~ > Metaspace

Опции JVM:

- -Xms начальный размер памяти в куче
- -Xmx максимальный размер памяти в куче
- -Xss размер памяти стека

Части Java Memory



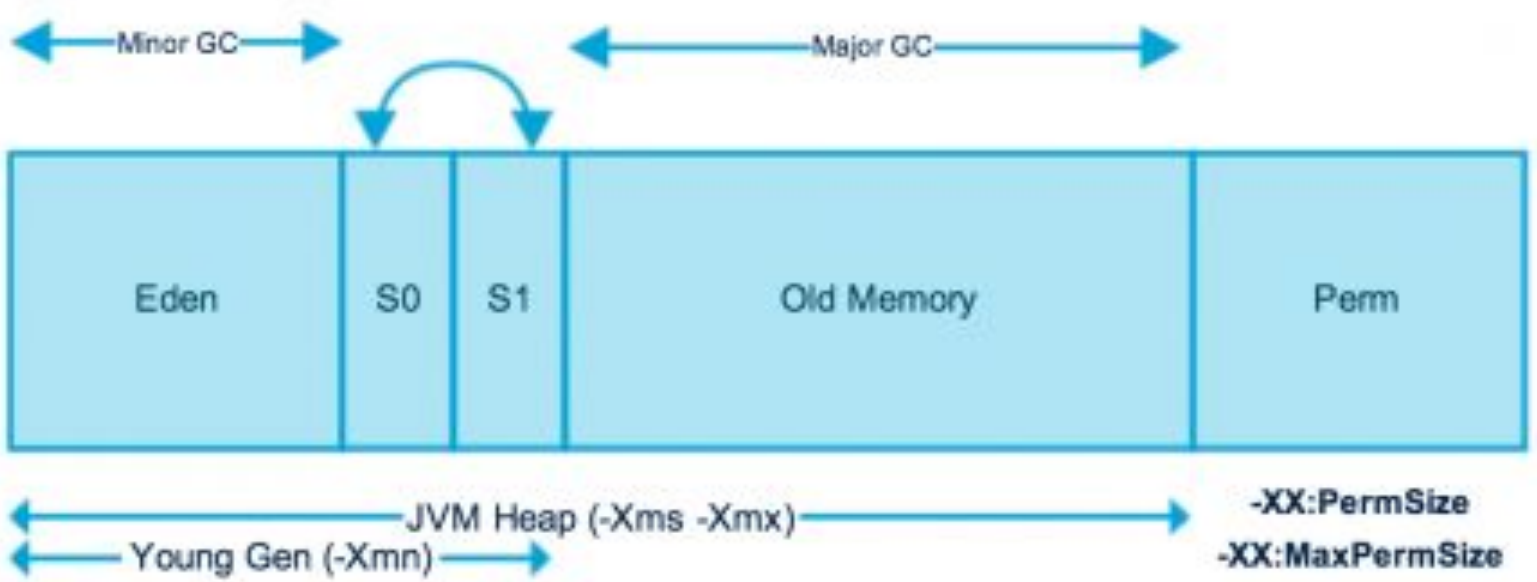
Java heap

- Куча используется для выделения экземпляров и массивов классов во время выполнения.
- Массивы и объекты никогда не могут храниться в стеке, потому что фрейм не предназначен для изменения размера после его создания. Фрейм хранит только ссылки, которые указывают на объекты или массивы в куче.
- В отличие от примитивных переменных и ссылок в массиве локальных переменных (в каждом фрейме) объекты всегда хранятся в куче, поэтому они не удаляются при завершении метода. Вместо этого объекты удаляются только сборщиком мусора.

Java heap

Для поддержки сборки мусора куча разделена на три раздела:

- Молодая генерация: Eden Space и Survivor Space
- Старая генерация - Tenured (Old) Generation
- Permanent Generation



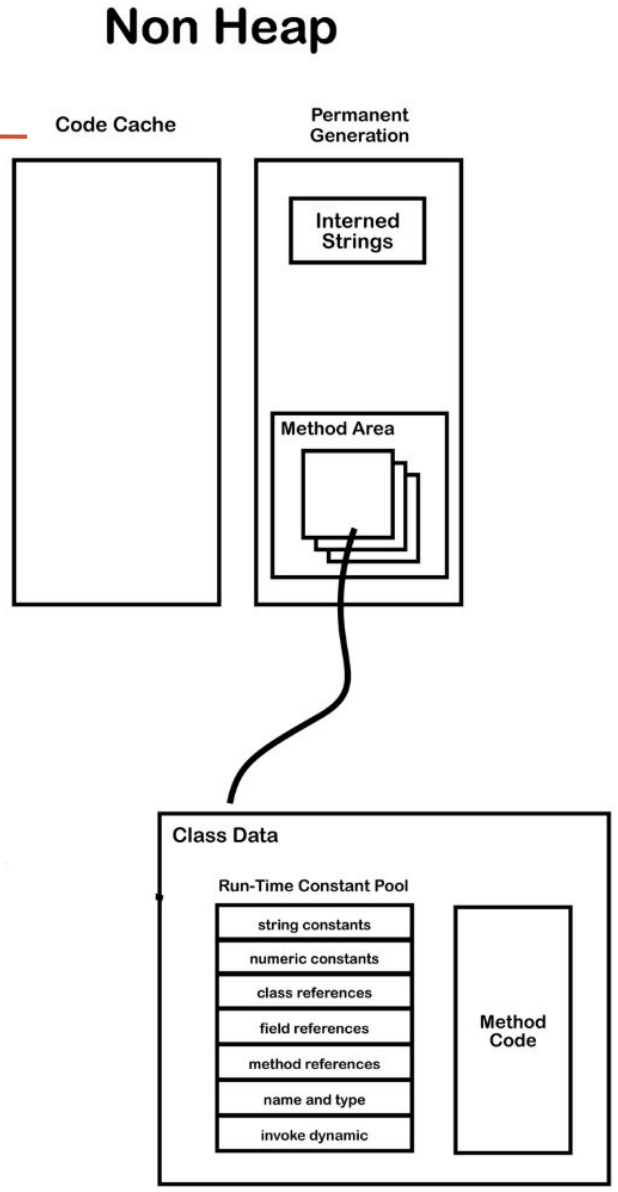
Permanent Generation

PermGen (Permanent Generation) - это специальное пространство кучи, отделенное от кучи основной памяти.

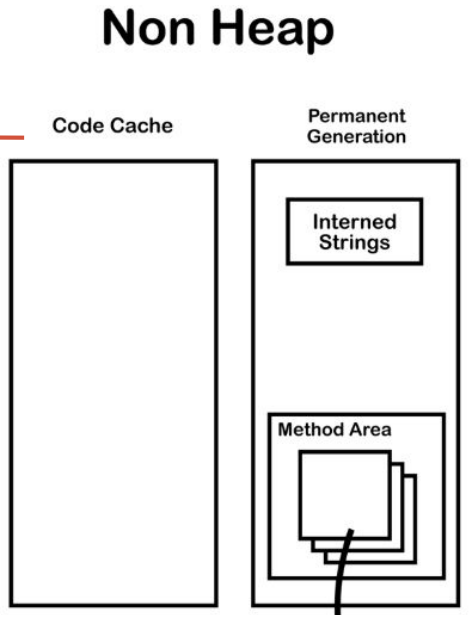
JVM отслеживает загруженные метаданные класса в PermGen. Кроме того, JVM хранит весь статический контент в этом разделе памяти. Это включает в себя все статические методы, примитивные переменные и ссылки на статические объекты.

Кроме того, он содержит данные о байт-коде, именах и JIT-информации. До Java 7, String Pool также был частью этой памяти. Недостатки фиксированного размера пула перечислены в нашей статье.

-XX: PermSize = [размер] - начальный или минимальный размер пространства PermGen.
-XX: MaxPermSize = [размер] - максимальный размер



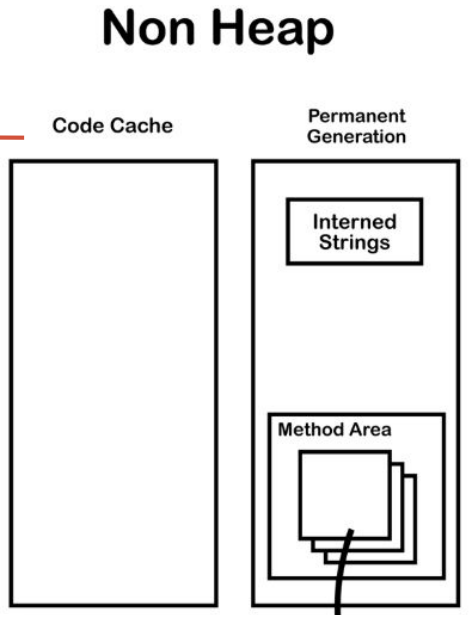
Java String Pool



Java String Pool - специальная область памяти, в которой строки хранятся в JVM.
Благодаря неизменности строк в Java JVM может оптимизировать объем выделяемой для них памяти, сохраняя только одну копию каждой литеральной строки в пуле. Этот процесс называется **интернированием**.

```
String first = "Baeldung";
String second = "Baeldung";
System.out.println(first == second); // True
```

Java String Pool



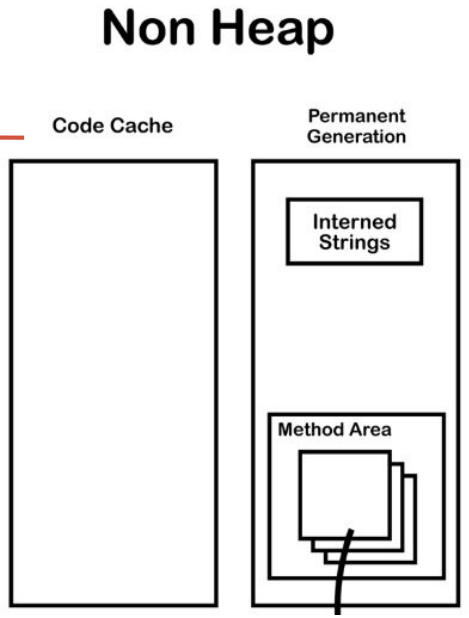
Когда создаем строку через оператор new, компилятор Java создаст новый объект и сохранит его в пространстве кучи, зарезервированном для JVM.

Каждая строка, созданная таким образом, будет указывать на другую область памяти со своим собственным адресом.

```
String third = new String("Baeldung");
String fourth = new String("Baeldung");
System.out.println(third == fourth); // False

String fifth = "Baeldung";
String sixth = new String("Baeldung");
System.out.println(fifth == sixth); // False
```

Java String Pool



Можно вручную интернировать String в пуле Java String, вызывая метод intern () для объекта, который мы хотим интернировать.

Интернирование String вручную сохранит его ссылку в пуле, а JVM вернет эту ссылку при необходимости.

```
String constantString = "interned Baeldung";
String newString = new String("interned Baeldung");

assertThat(constantString).isNotSameAs(newString);

String internedString = newString.intern();

assertThat(constantString)
  .isSameAs(internedString);
```

Metaspace

Metaspace - это новое пространство памяти - начиная с версии Java 8; он заменил старое пространство памяти PermGen. Metaspace по умолчанию может автоматически расти.

- **MetaspaceSize** и **MaxMetaspaceSize** - верхние границы Metaspace.
- **MinMetaspaceFreeRatio** - минимальный процент емкости метаданных, свободных после сборки мусора
- **MaxMetaspaceFreeRatio** - это максимальный процент емкости метаданных класса, свободной после сборки мусора, чтобы избежать уменьшения объема пространства

Garbage collection

Традиционно, при определении эффективности работы сборщика мусора учитываются следующие факторы:

- Максимальная задержка — максимальное время, на которое сборщик приостанавливает выполнение программы для выполнения одной сборки. Такие остановки называются *stop-the-world*(или *STW*).
- Пропускная способность — отношение общего времени работы программы к общему времени простоя, вызванного сборкой мусора, на длительном промежутке времени.
- Потребляемые ресурсы — объем ресурсов процессора и/или дополнительной памяти, потребляемых сборщиком.

Garbage collection

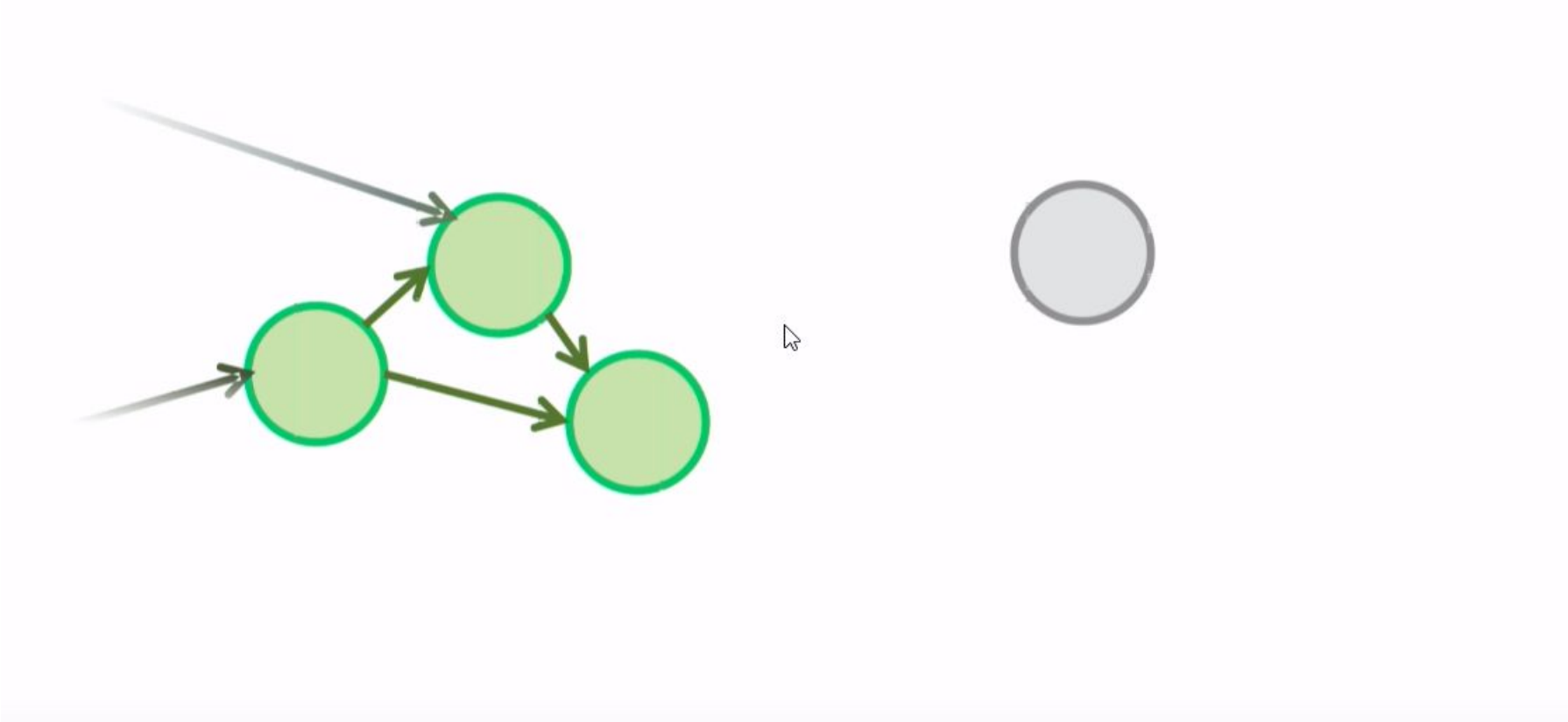
Java HotSpot VM предоставляет разработчикам на выбор четыре различных сборщика мусора:

Serial (последовательный) — самый простой вариант для приложений с небольшим объемом данных и не требовательных к задержкам. Редко когда используется, но на слабых компьютерах может быть выбран виртуальной машиной в качестве сборщика по умолчанию.
Parallel (параллельный) — наследует подходы к сборке от последовательного сборщика, но добавляет параллелизм в некоторые операции, а также возможности по автоматической подстройке под требуемые параметры производительности.
Concurrent Mark Sweep (CMS) — нацелен на снижение максимальных задержек путем выполнения части работ по сборке мусора параллельно с основными потоками приложения. Подходит для работы с относительно большими объемами данных в памяти.
Garbage-First (G1) — создан для постепенной замены CMS, особенно в серверных приложениях, работающих на многопроцессорных серверах и оперирующих большими объемами данных.

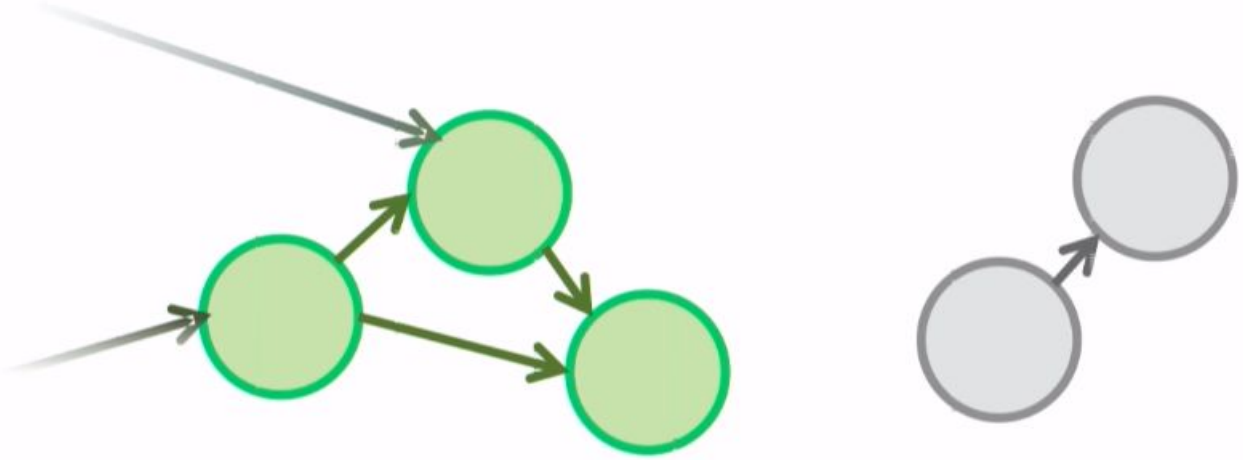
Garbage collection

- Если разработчик вызывает System.gc () или Runtime.getRunTime ().gc () предлагает JVM инициировать GC.
- Если JVM решит, что недостаточно места для аренды.
- Во время второстепенного GC, если JVM не в состоянии восстановить достаточно памяти из пространств Eden или Survivor, тогда может быть запущен главный GC.
- Если установлена опция «MaxMetaspaceSize» для JVM и не хватит места для загрузки новых классов, то JVM вызовет основной GC.

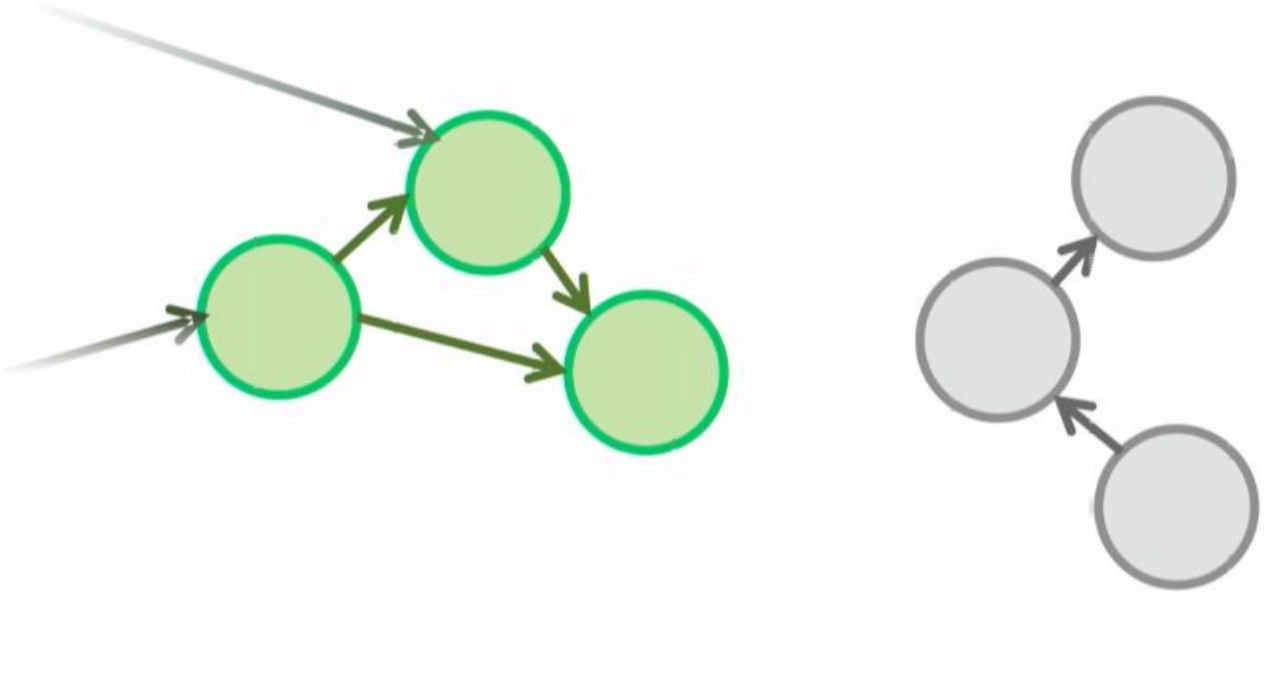
Что такое мусор?



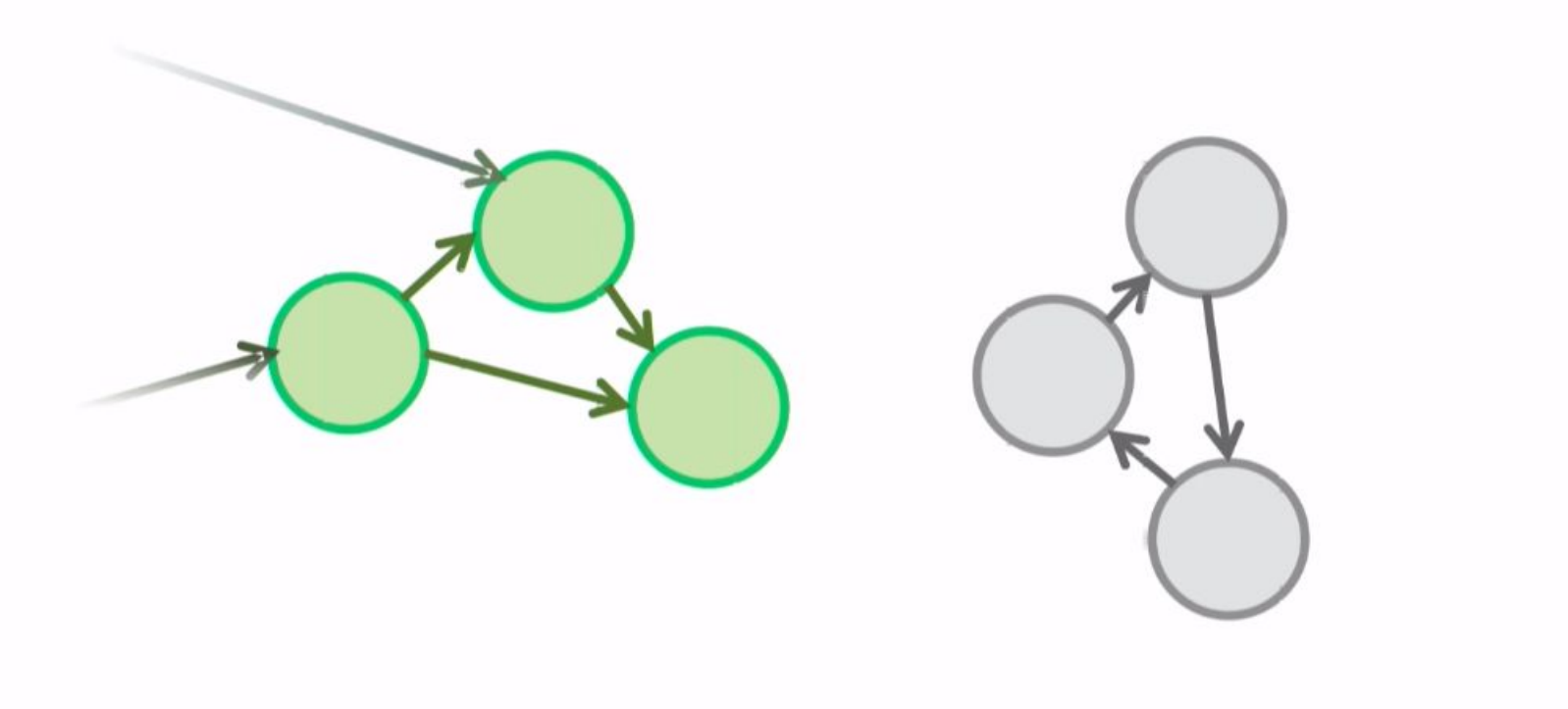
Что такое мусор?



Что такое мусор?



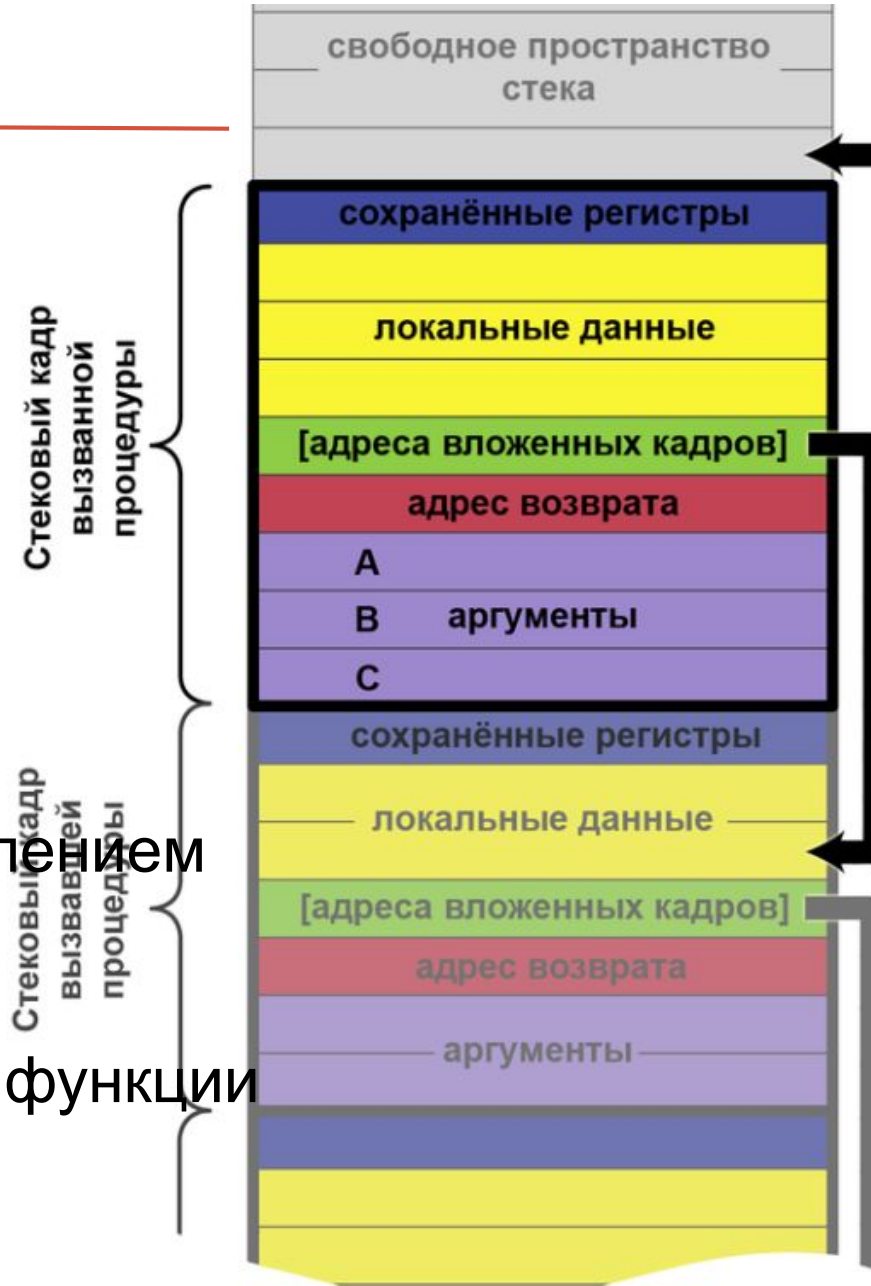
Что такое мусор?



Стек вызовов

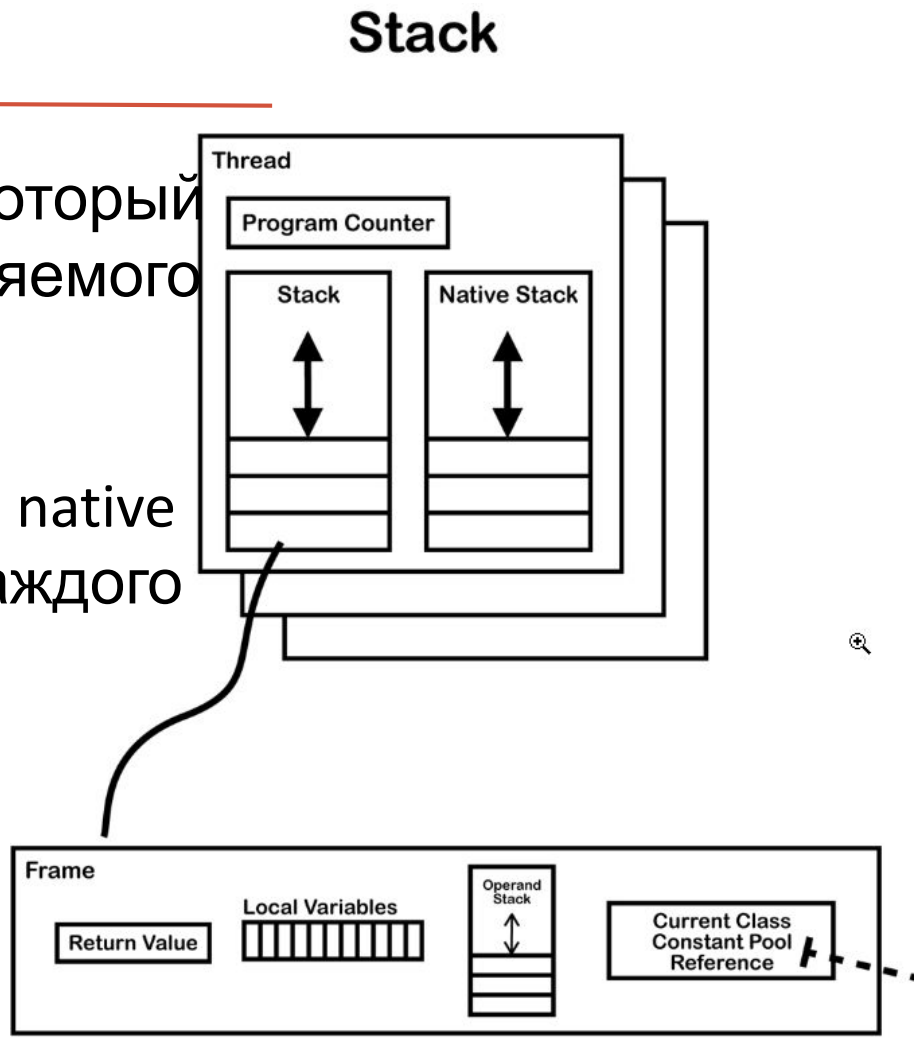
Стек вызовов— в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм (процедур, функций) в программу (или подпрограмму, при вложенных или рекурсивных вызовах) и/или для возврата в программу из обработчика прерывания (в том числе при переключении задач в многозадачной среде).

- Адреса возврата
- значения регистров с их последующим восстановлением
- данные стекового кадра языков высокого уровня:
  - аргументы, переданные в функцию
  - локальные переменные — временные данные функции
- другие произвольные данные



Стэк Java

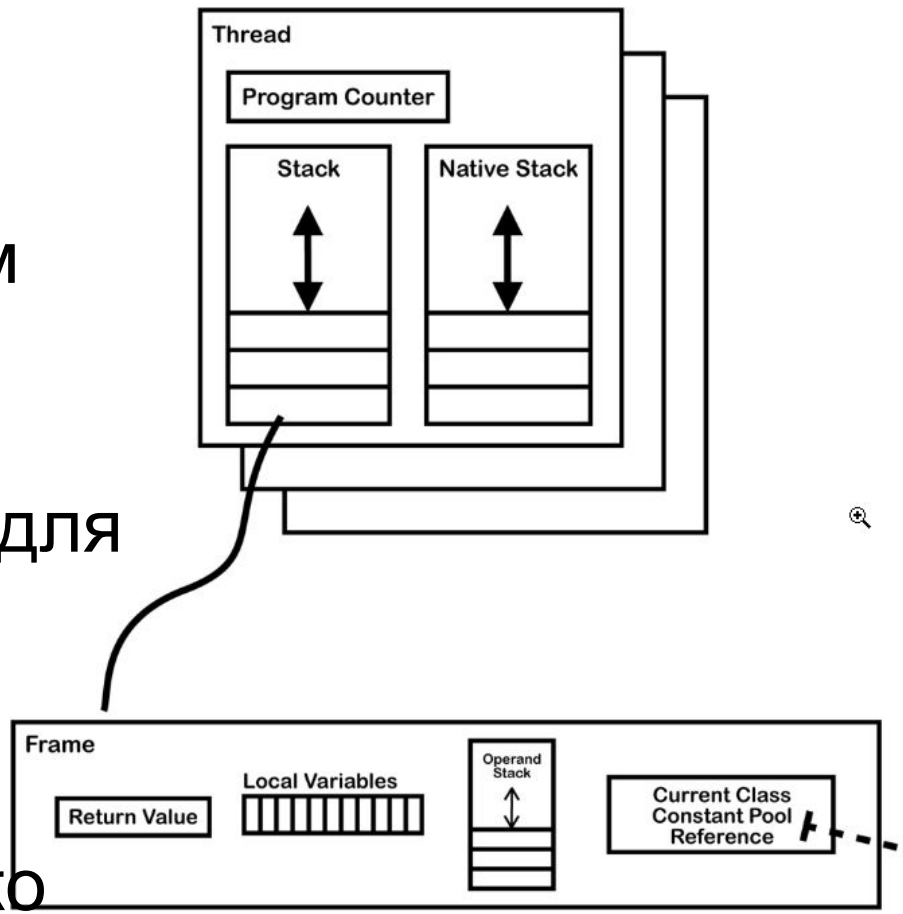
- Каждый поток имеет свой собственный стек, который содержит фрейм для каждого метода, выполняемого в этом потоке.
- Структура LIFO
- Если JVM поддерживает JNI и, соответственно, native методы, то создатся стек native методов для каждого потока.



Ограничения на стек Java

Стек может быть динамического или фиксированного размера.

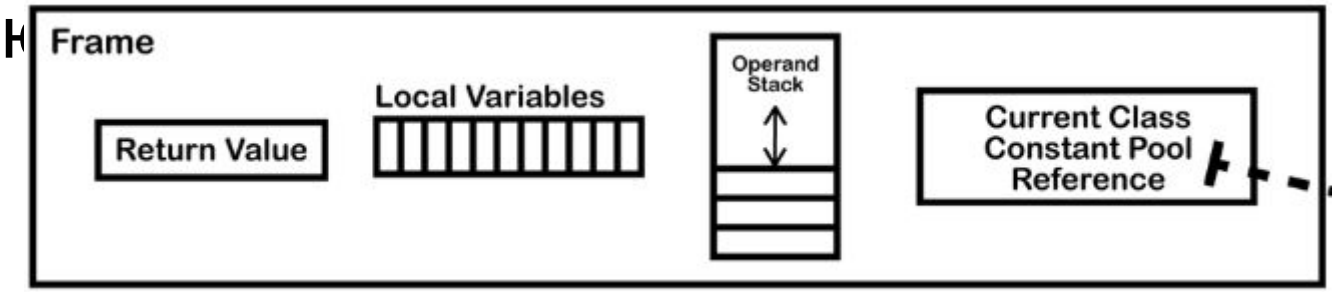
- Если поток требует большего стека, чем позволено, выдается ошибка StackOverflowError.
- Если потоку требуется новый фрейм, и для его выделения недостаточно памяти, генерируется ошибка OutOfMemoryError.



При старте в HotSpot можно указать сколько памяти для стека выделяется на 1 поток.

Frame (Кадр)

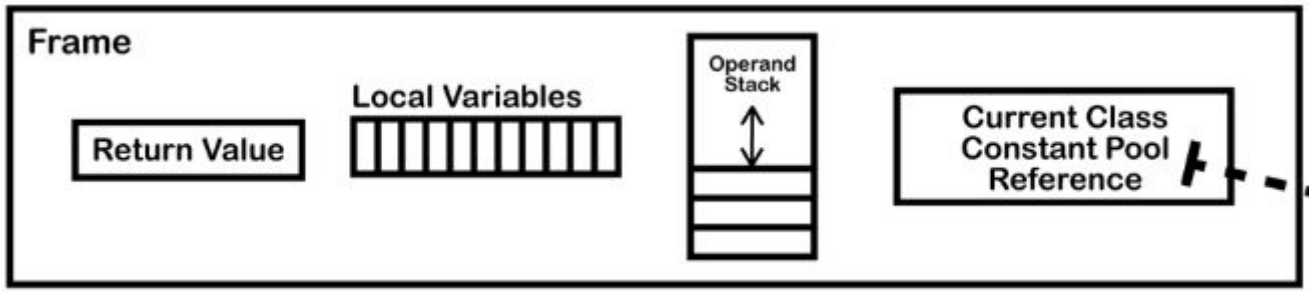
- Каждый кадр содержит ссылку на пул констант текущего классав рантайме.
- Новый фрейм создается и добавляется (push) в верхнюю часть стека для каждого вызова метода.
- Фрейм удаляется (pop), когда метод возвращается нормально или если во время вызова метода генерируется неперехваченное искл



Массив локальных переменных

- Массив локальных переменных содержит все переменные, используемые во время выполнения метода, включая ссылку на него, все параметры метода и другие локально определенные переменные.
- Локальная переменная может быть:

  boolean byte char **long** short int float **double** reference returnAddress



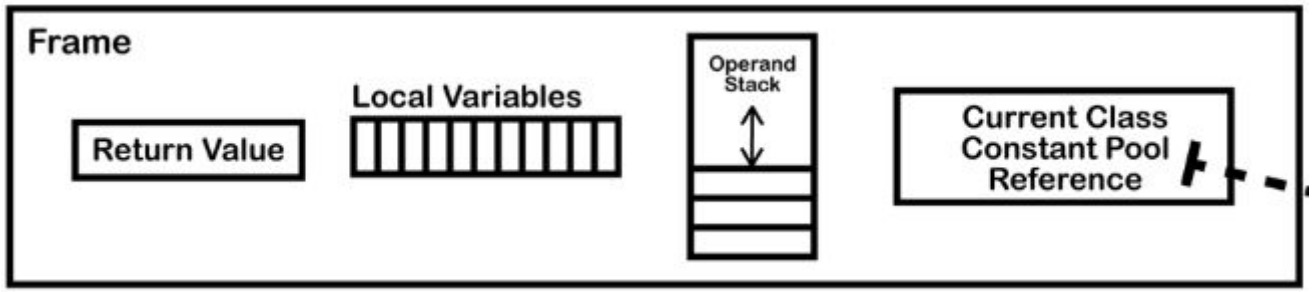
Стек операндов

Стек операндов используется во время выполнения инструкций байтового кода аналогично тому, как регистры общего назначения используются в собственном CPU.

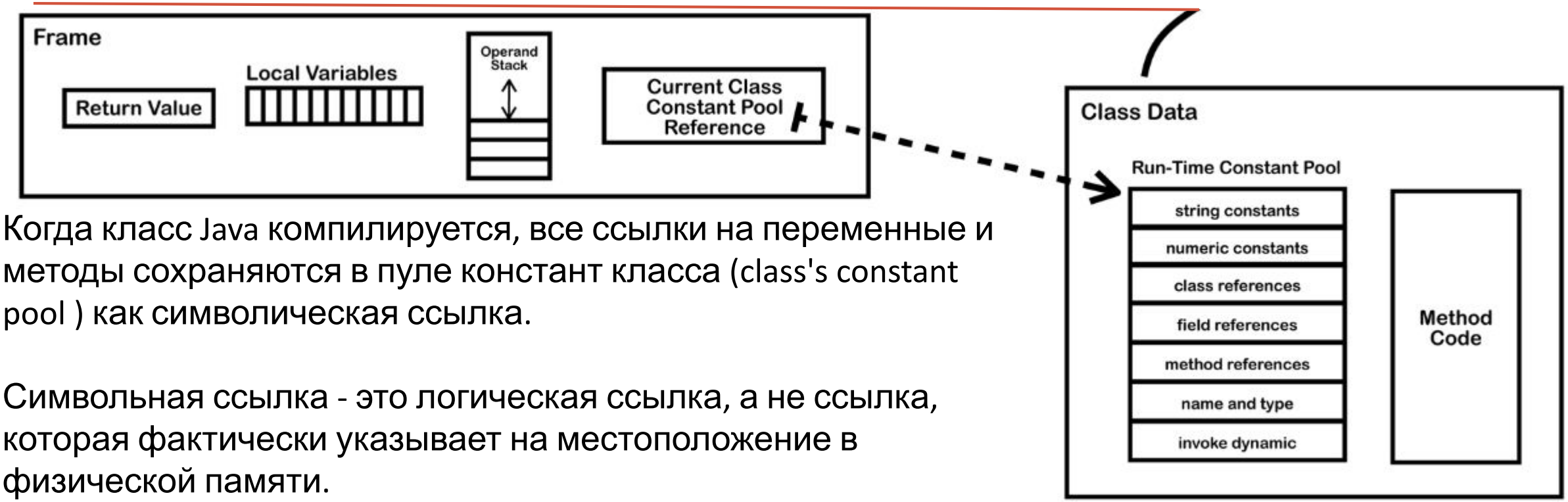
```
int i;
```

Будет скомпилированно в следующий байт-код:

```
0: iconst_0     // Push 0 to top of the operand stack
1: istore_1     // Pop value from top of operand stack and store as local
variable 1
```



Динамическая линковка



Когда класс Java компилируется, все ссылки на переменные и методы сохраняются в пуле констант класса (class's constant pool) как символическая ссылка.

Символьная ссылка - это логическая ссылка, а не ссылка, которая фактически указывает на местоположение в физической памяти.

Привязка (Binding) - это процесс, в котором поле, метод или класс, идентифицируемые символической ссылкой, заменяются прямой ссылкой, это происходит только один раз, потому что символическая ссылка полностью заменяется.