Въведение в Java

Борислав Капукаранов, Иван Ст. Иванов / SAP Labs Bulgaria
Октомври, 2014

Public

Съдържание

Езикът Java

Архитектура

Java виртуалната машина

Garbage collection

Java екосистема

Интегрирани среди за разработка

Java езикът

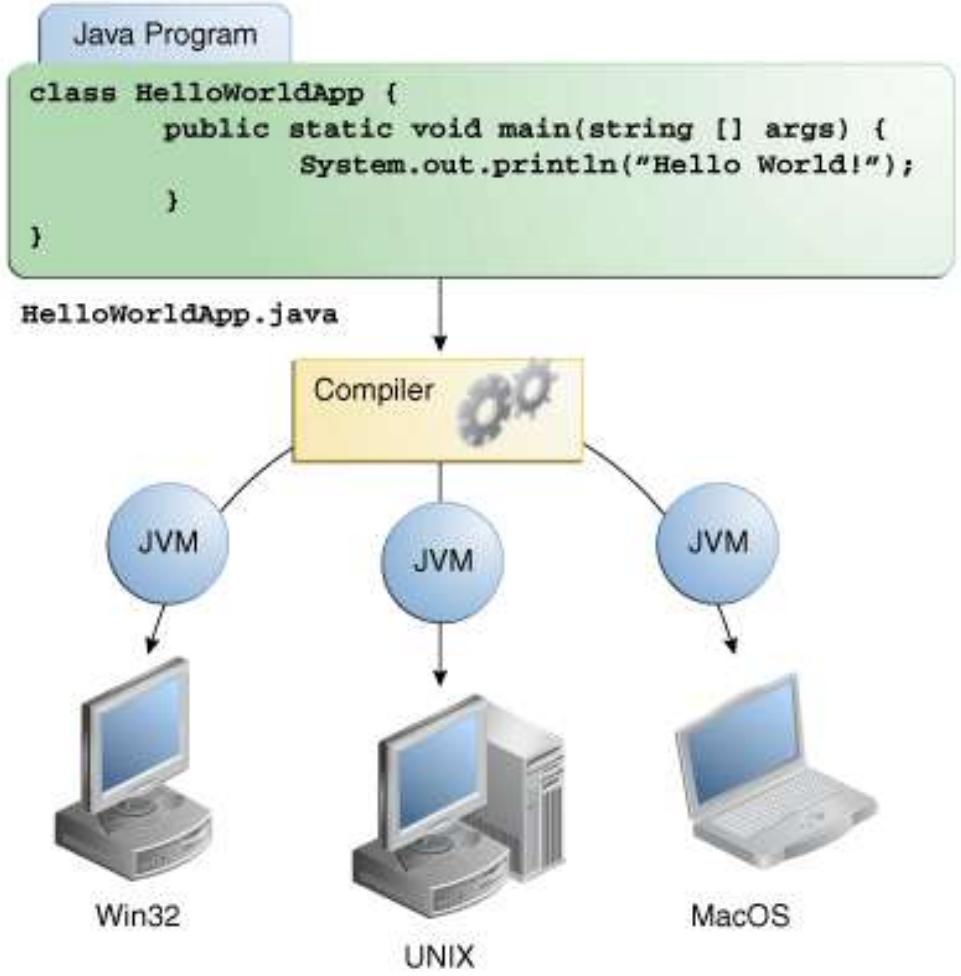
- Създаден през 1995 от James Gosling от Sun Microsystems
- Обектно ориентиран
- Статично компилируем
- Със C/C++ синтаксис
- Напиши веднъж, пусни навсякъде

Hello world!

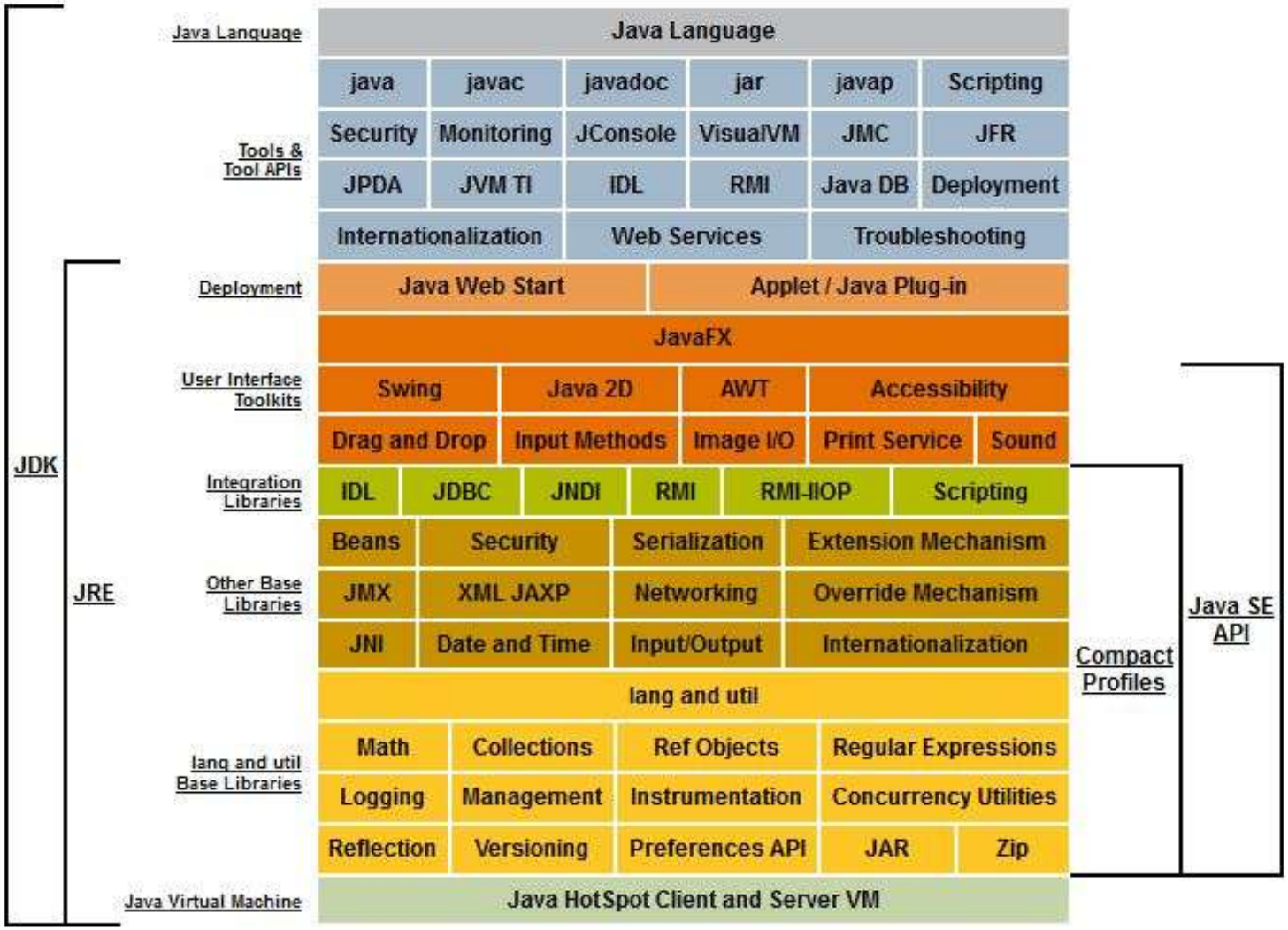
```
public class HelloWorldApp {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Стандартно Java приложение



Архитектура на JDK



Java виртуалната машина

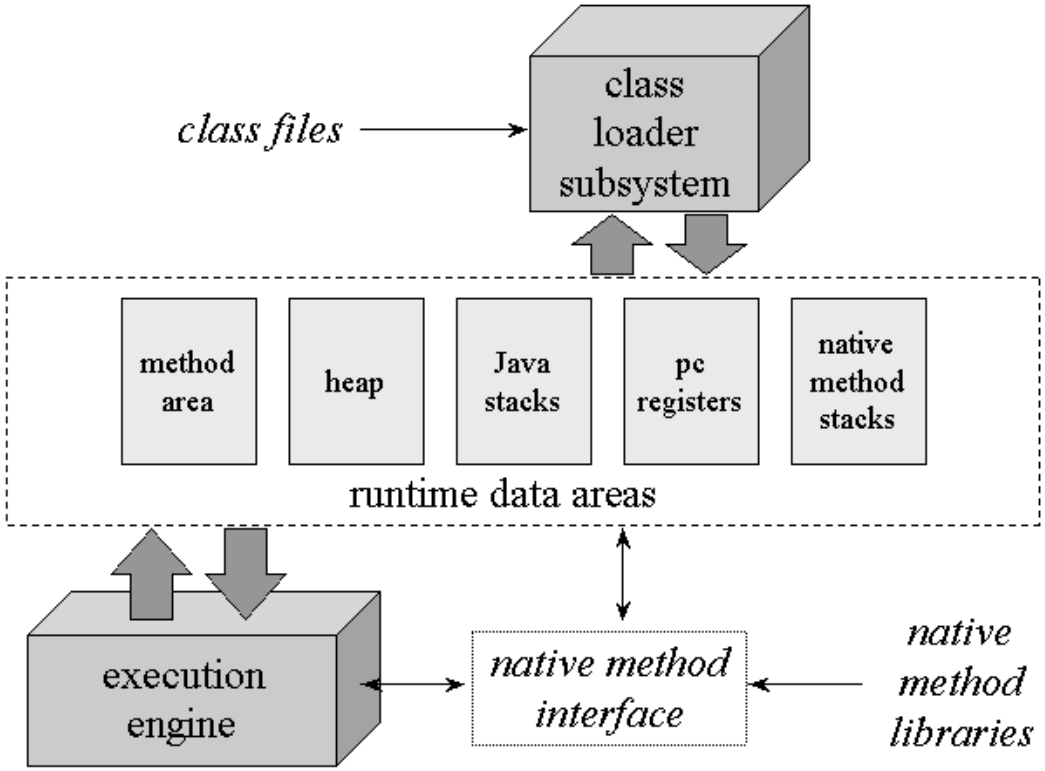
Типичната виртуална машина

- Изпълнява компилирания до специфичен за нея byte код изходен код
- Предоставя структури от данни за съхранение на инструкции и операнди (данните, които инструкциите обработват)
- Поддържа стек за обработване на извикванията към функции
- Съблюдава ‘Instruction Pointer’ (IP), който сочи към следващата инструкция за изпълнение
- Има виртуално CPU – диспечер, за инструкции

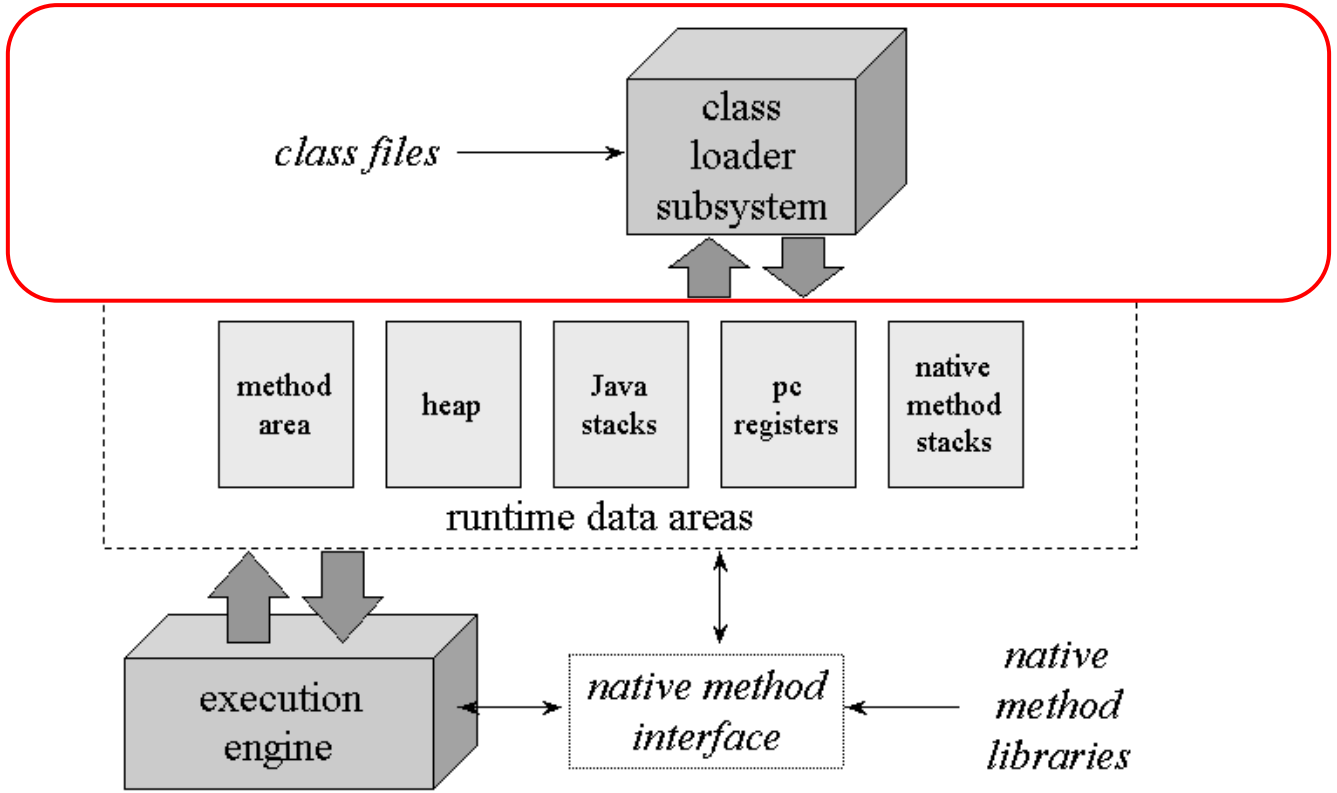
Hotspot виртуалната машина

- Интерпретира и изпълнява byte код инструкции
- Компилира по време на изпълнението байт кода до машинен код
- Заделя памет за оперативните данни
- Автоматично изчиства паметта
- Зарежда класове
- Стартира нишки
- Взаимодейства с операционната система
- Два типа: client и server

Архитектура на Hotspot VM



Classloader под-система



Формат на клас файл

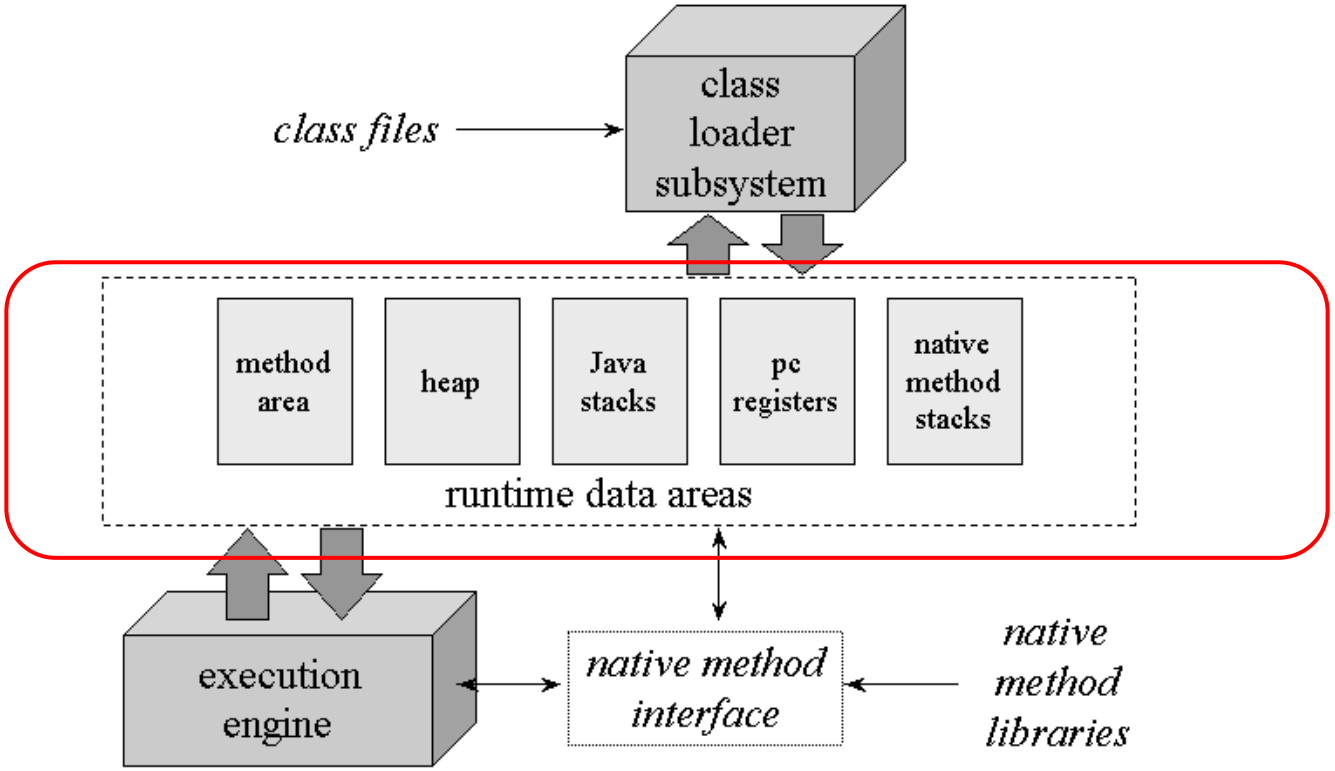
```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

Фази на зареждане на класа

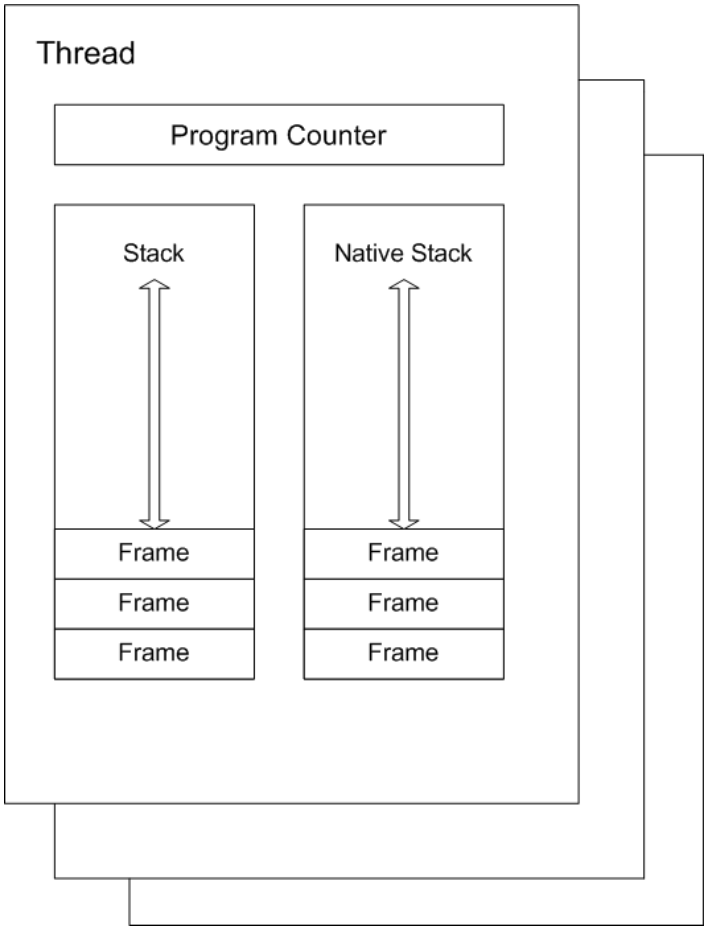
Три фази на зареждане на класа:

- Зареждане
- Свързване
- Инициализация

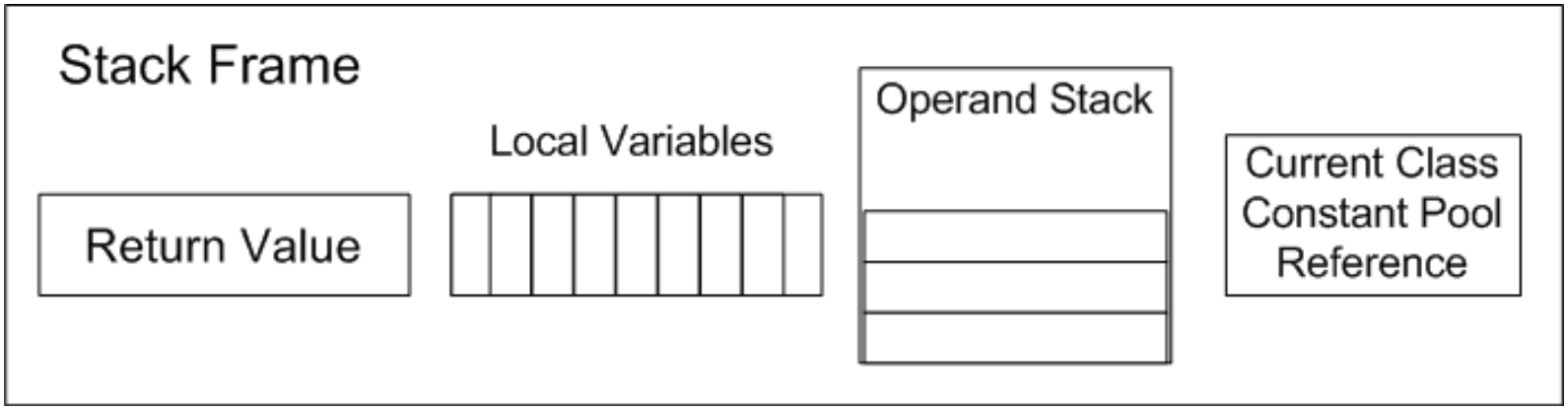
Runtime data под-система



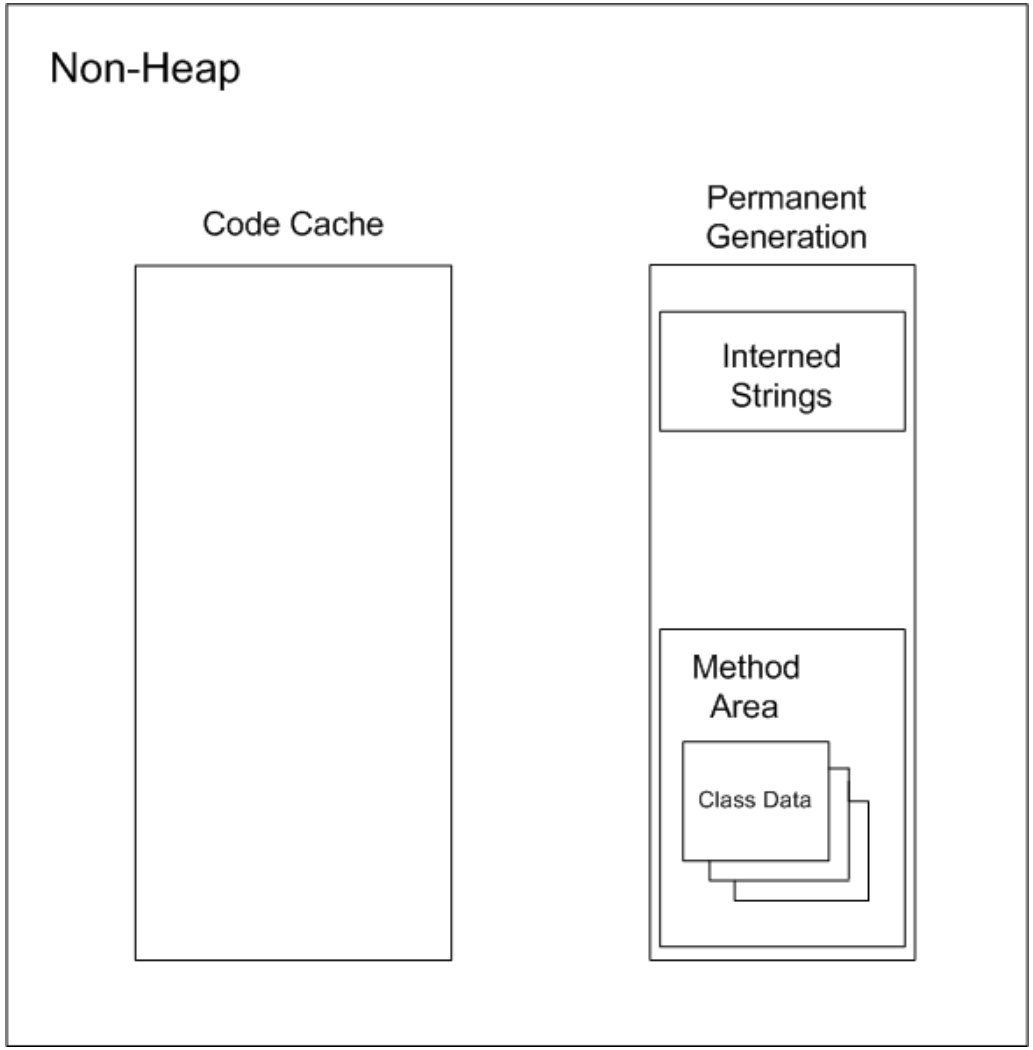
Поглед в heap паметта



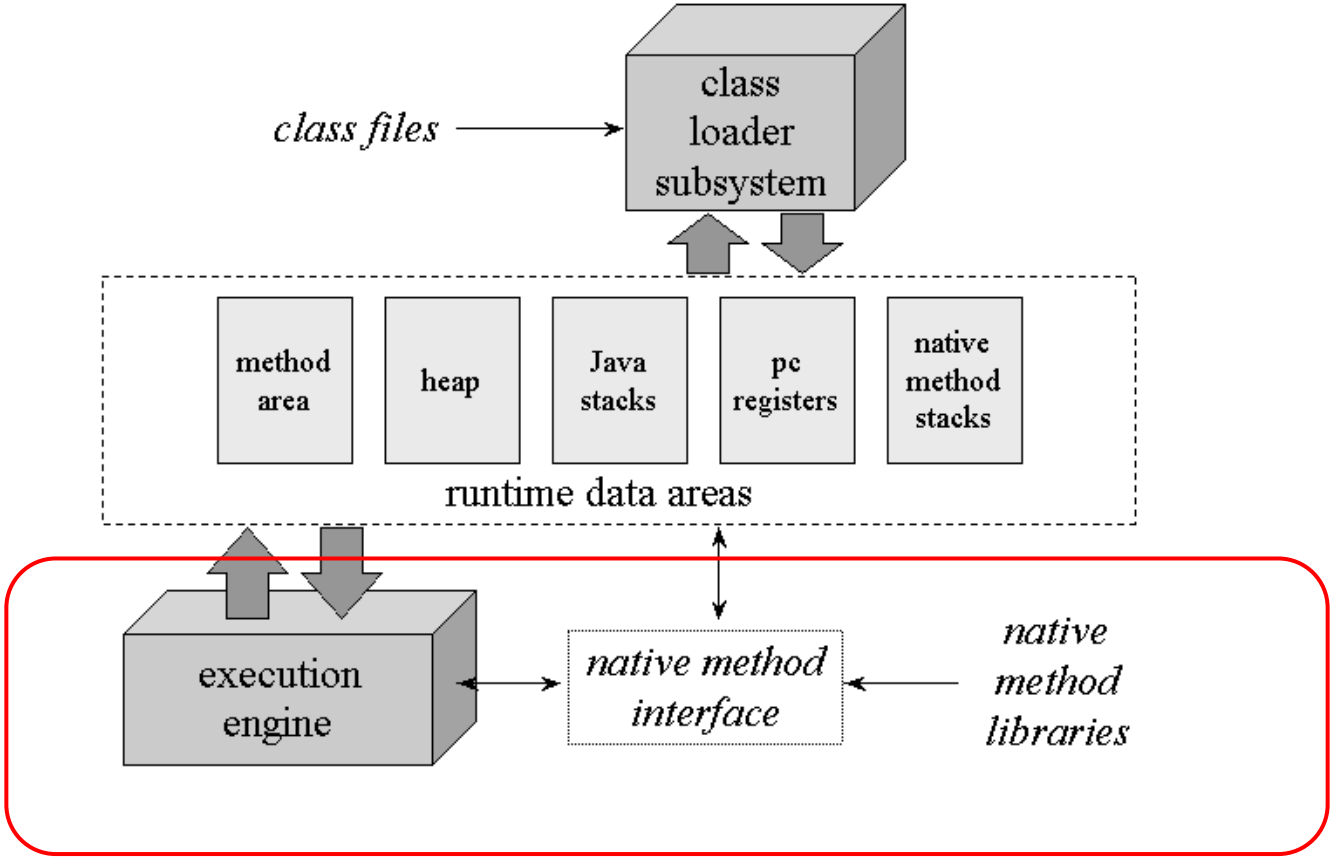
При изпълнение на метод



Извън heap паметта



Execution под-система



Изпълнение на интерпретиран код

```
while(true) {
        bytecode b = bytecodeStream[pc++];
        switch(b) {
                case iconst_1: push(1); break;
                case iload_0: push(local(0)); break;
                case iadd: push(pop() + pop()); break;
        }
}
```

Изпълнение на интерпретиран код

```
while(true) {
        bytecode b = bytecodeStream[pc++];
        switch(b) {
                case iconst_1: push(1); break;
                case iload_0: push(local(0)); break;
                case iadd: push(pop() + pop()); break;
        }
}
```

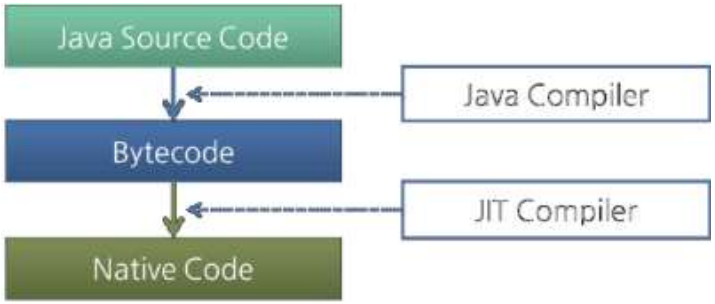
Не е толкова просто ...

Техники на изпълнение

Интерпретиране

Just-in-time (JIT) компилация

Адаптивна компилация



JIT компилация

- Стартира се автоматично при наличие на многократно изпълнение на даден метод
- Генерира машинен код
- Ако компилиран код вика некомпилиран, изпълнението се връща на интерпретатора
- Компилиран код може да бъде де-оптимизиран до интерпретиран
- По-агресивни оптимизации на server виртуалната машина

Да обобщим

1) Parse-ват се параметрите на командната линия
2) Заделя се паметта
3) Прочитат се environment променливите
4) Открива се главния клас (main class)
5) Създава се и се инициализира виртуалната машина
6) Зарежда се на главния клас
7) Изпълнява се неговия main метод
8) Когато изпълнението му свърши, се подава връщаният от него резултат
9) Унищожава се виртуалната машина

Garbage collection

Преди VM-ите

Програмистите сами се грижат за паметта, която ползват

Цял клас грешки, които могат да бъдат допуснати

Чрез GC самия VM се грижи за паметта, която е заделил за нас

Garbage Collection

И Java обектите имат живот...

...някои дълъг, други кратък

Garbage Collection

И Java обектите имат живот...

...някои дълъг, други кратък

Те живеят в паметта, която виртуалната машина е запазила (heap)

-Xmx:2g

Garbage Collection

И Java обектите имат живот...

...някои дълъг, други кратък

Те живеят в паметта, която виртуалната машина е запазила (heap)

-Xmx:2g

Какво става като се напълни heap-а?

Garbage Collection

И Java обектите имат живот...

...някои дълъг, други кратък

Те живеят в паметта, която виртуалната машина е запазила (heap)

-Xmx:2g

Какво става като се напълни heap-а?

Трябва да изхвърлим боклука...

Two-generational parallel garbage collection with a heap compaction phase.

При пускане на GC…

Не всеки момент е подходящ за GC

Виртуалната машина се грижи да избере момент, в който е безопастно

Например, когато не тече заделяне на памет за нови обекти (safe points)

Кои обекти са боклук?

Обекти, до които никой “жив” обект няма връзка.

GC процеса гарантира, че никой “жив” обект няма да бъде събран, но няма гаранция, че “умрелите” обекти ще бъдат събрани веднага

Кои обекти са боклук?

Обекти, до които никой “жив” обект няма връзка.

GC процеса гарантира, че никой “жив” обект няма да бъде събран, но няма гаранция, че “умрелите” обекти ще бъдат събрани веднага

Предизвикателството пред GC е да открие кой е “жив”

Reference Counting

Ранна и интуитивна техника

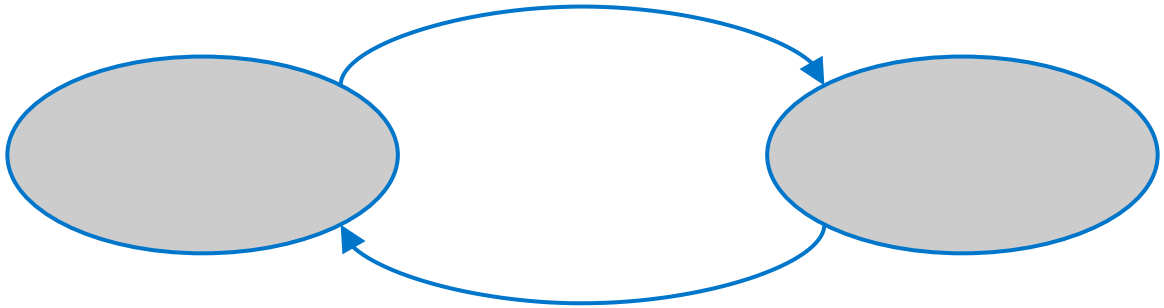
Тези събирачи следят колко връзки сочат към даден обект

Когато броят им стане 0, паметта веднага се освобождава

Reference Counting

Проблеми:

- основната трудност е да държим връзките актуални
- сложно се справят с циклични структури от обекти (overhead)



Tracing Collectors

Базират се на идеята, че можем да открием всички активни обекти като вървим по графа от връзки.

1. Съставя се начално множество сигурни активни обекти(roots), чрез анализ на регистрите*, глобалните полета, статичните полета и други
2. След това се обхожда всичко свързано с root обектите и се отбелязва като активно
3. Всичката останала заета памет, се освобождава

Предимство: справят се лесно с циклични структури

Tracing Collectors

Недостатък: фазата на отбелязване предизвиква изчакване (stop-the-world)

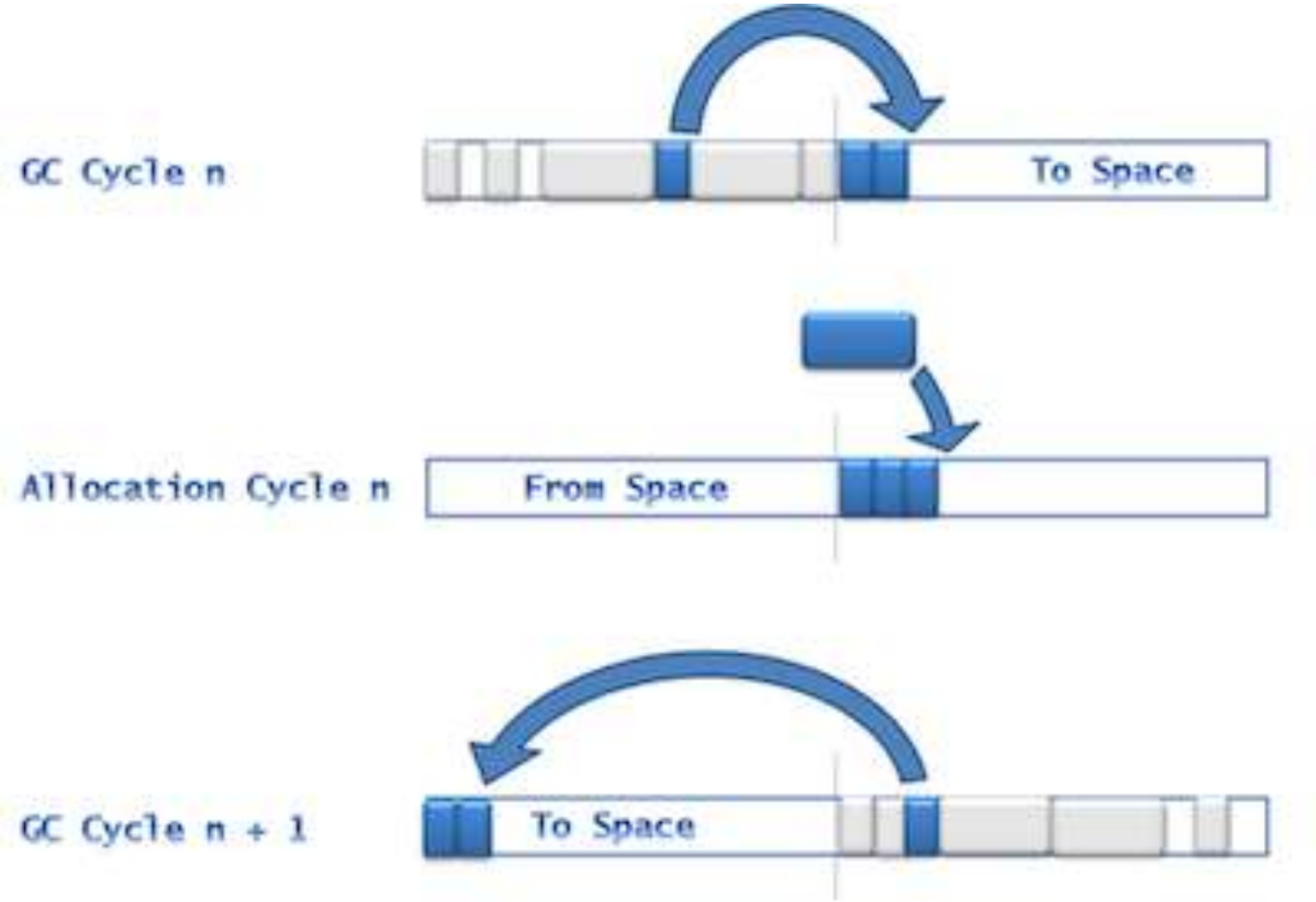
Най-популярния избор за подход при имплементация на collector

Използват се широко при Java езика и са доказани в продуктивни условия

Видове Tracing Collectors

- Copying
- Mark-and-sweep
  - parallel
  - concurrent

Copying Collectors



Copying Collectors

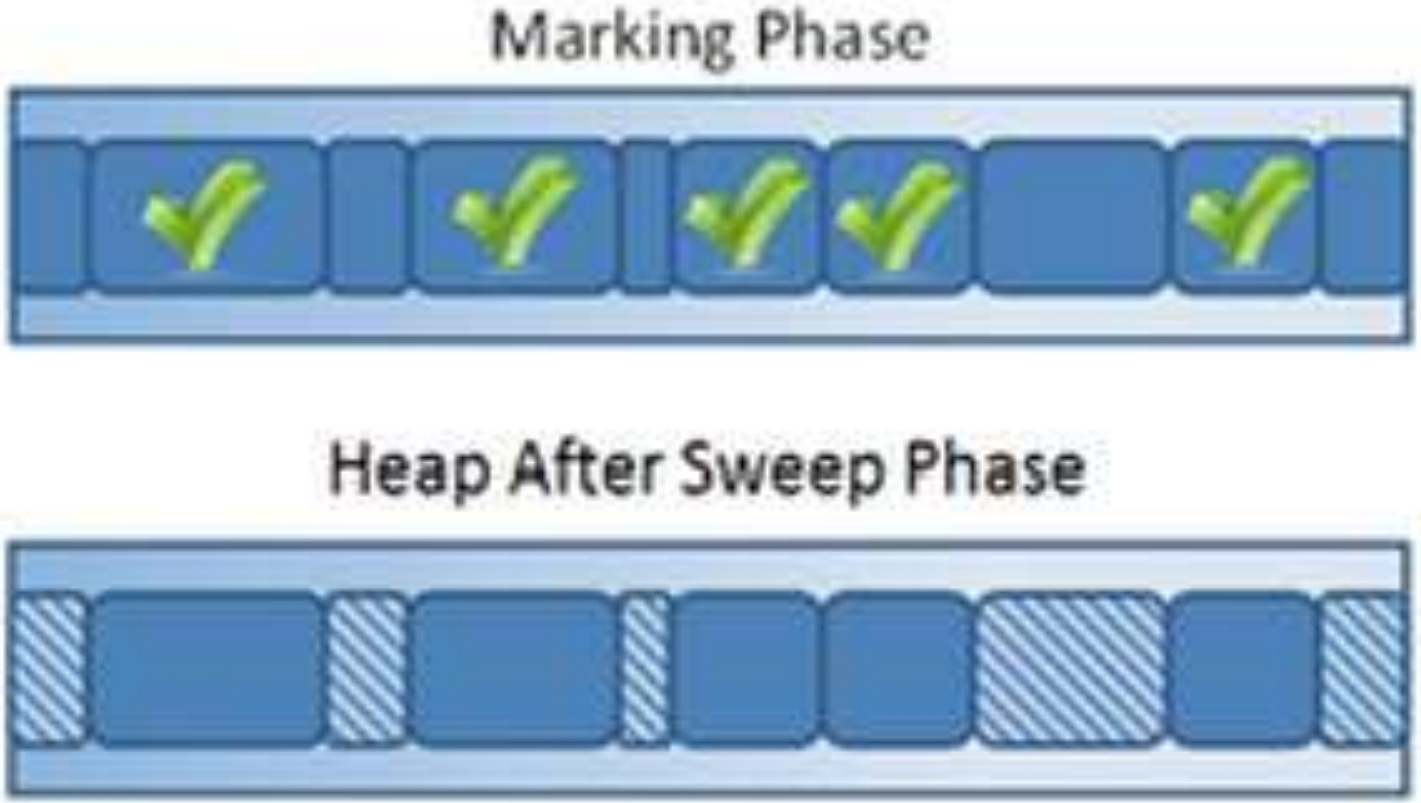
Предимства:

- обектите са гъсто събрани
- няма фрагментация

Недостатъци:

- stop-the-world collectors (big copy area)
- неефикасни откъм памет (enough headroom)

Mark-and-sweep collectors



Mark-and-sweep collectors

Mark

- отбелязва с по един bit всички активни обекти, обикаляйки heap-а по връзките

Sweep

- обикаля целия heap
- паметта, което не е отбелязана може да се ползва (freelists)
- chunks

Mark-and-sweep collectors

Недостатъци

- Mark фазата е зависима от количеството активни обекти в heap-а
- Sweep е зависима от размера на целия heap
- Тъй като GC чака и двете да приключат, това би могло да довече до сериозни паузи

Mark&sweep parallel collectors

Всички налични ресурси се използват паралелно за изпълнение на GC

монолитен stop-the-world подход - всички нишки на приложението спират докато не мине collector-a

Mark&sweep parallel collectors

По-ефикасно събиране, тъй като участват всички ресурси

Може да довече до големи паузи в приложението, ако има голям обем активни обекти

Неподходящо за приложения, чувствителни към време за връщане на отговор

Mark&sweep concurrent collectors

Много по-подходящи за приложения изискващи бързо връщане на отговор

GC тук се върши, докато приложението продължава да работи върху JVM

Повече предизвикателства…

Mark&sweep concurrent collectors

- Кога да стартира GC?
- Ще има ли достатъчно време, преди приложението да остане без памет?
- Не искаме постоянно да правим GC - яде ресурсите на приложението ни
- re-marking loops - може да е трудно да се освободи памет

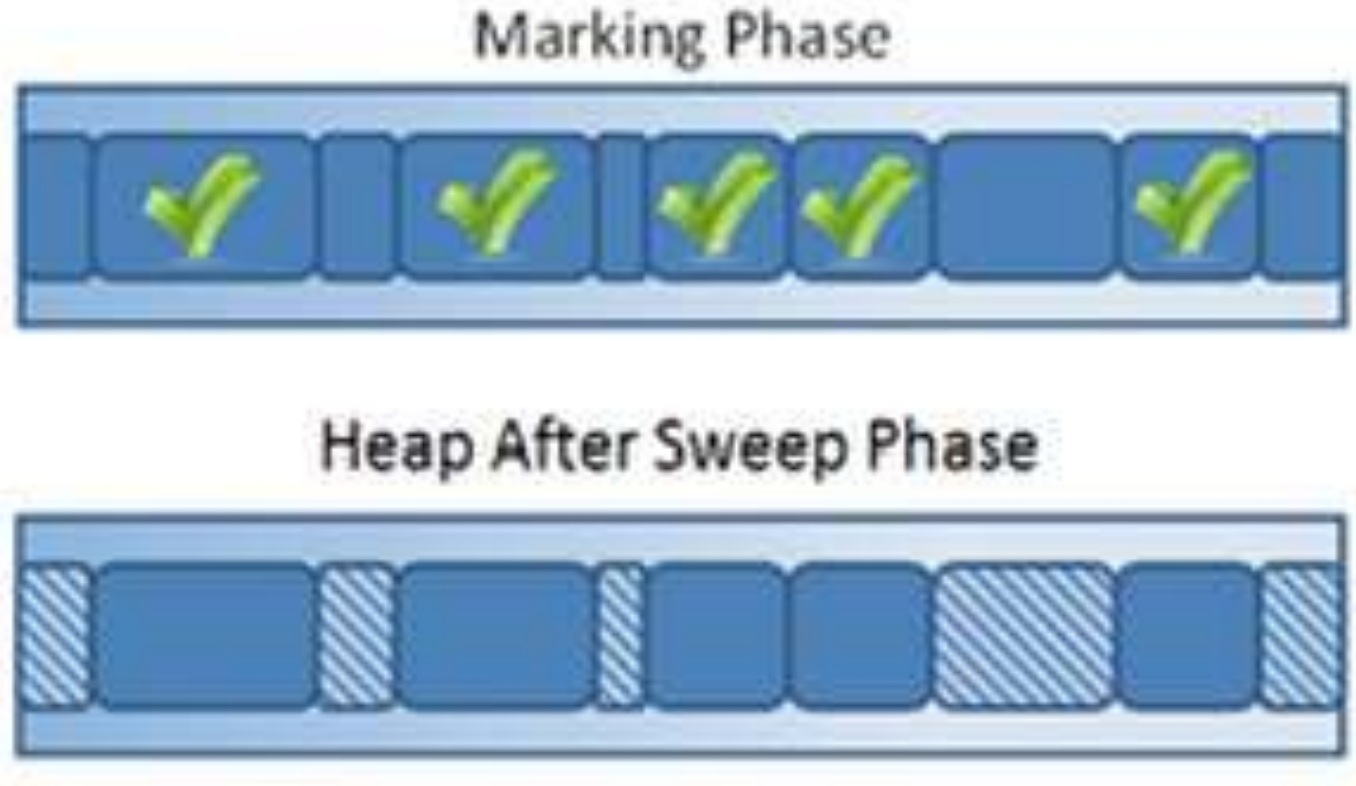
Големия проблем

f r a g m e n t a t i o n

истинското предизвикателсно на Java не е garbage collector-a, а фрагментацията на паметта и как collector-a се справя с нея

Fragmentation

памет има, но не в достатъчно големи парчета



Generational Garbage Collection

Повечето обекти умират…

Generational Garbage Collection

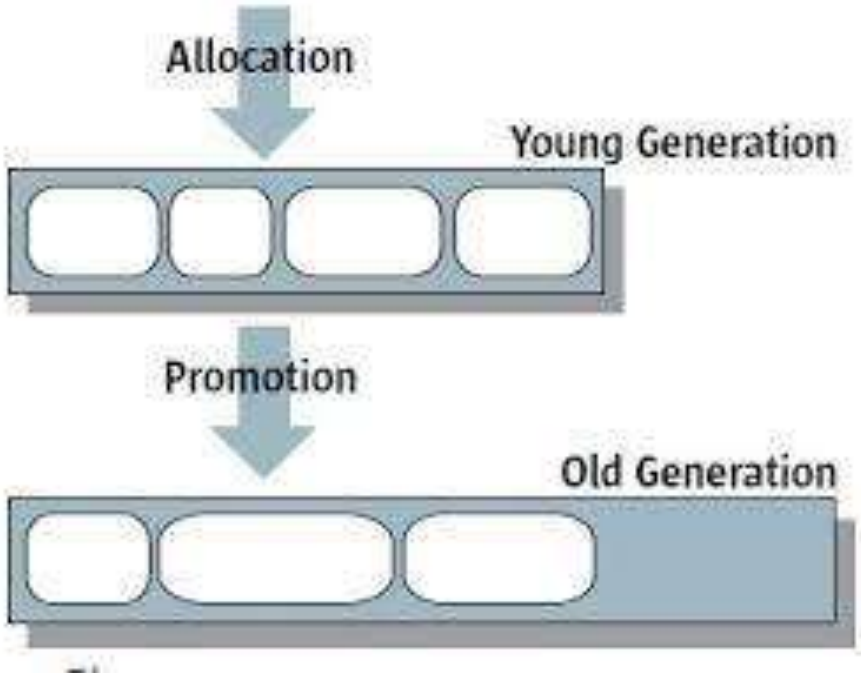
Повечето обекти умират… млади.
Heap-а се разделя на части (поколения) - най-често две (young и old)

Обектите преминават от младото към старото поколение, ако са оцелели след определено време или брой GC-ни

Може всяко поколение се събира от различен тип collector

Generational Garbage Collection

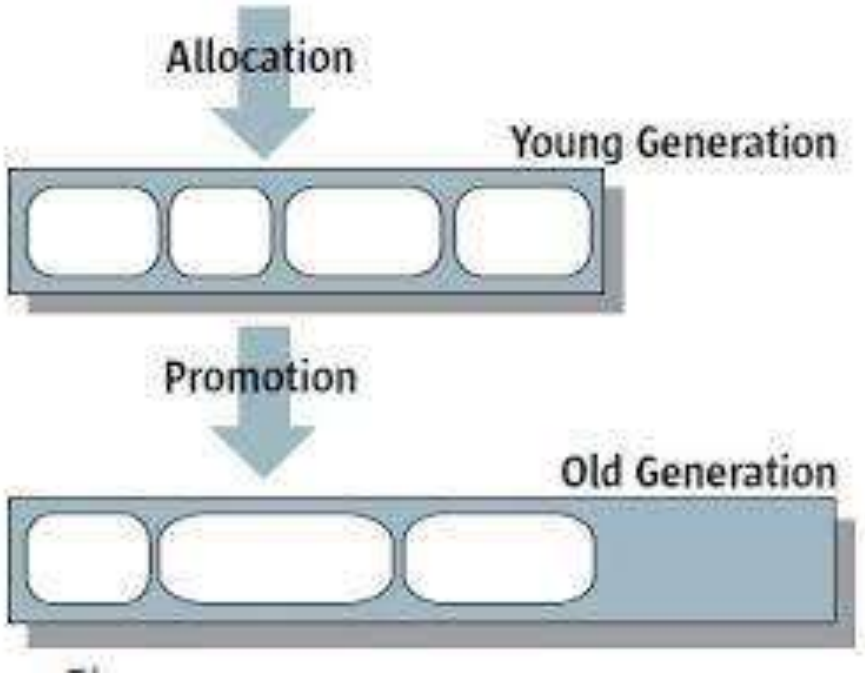
One-directional copy collectors
Parallel collectors for young gen
Rarely big objects go straight to old gen



Generational Garbage Collection

One-directional copy collectors
Parallel collectors for young gen
Rarely big objects go straight to old gen

Fragmentation is still
not solved,
just delayed!

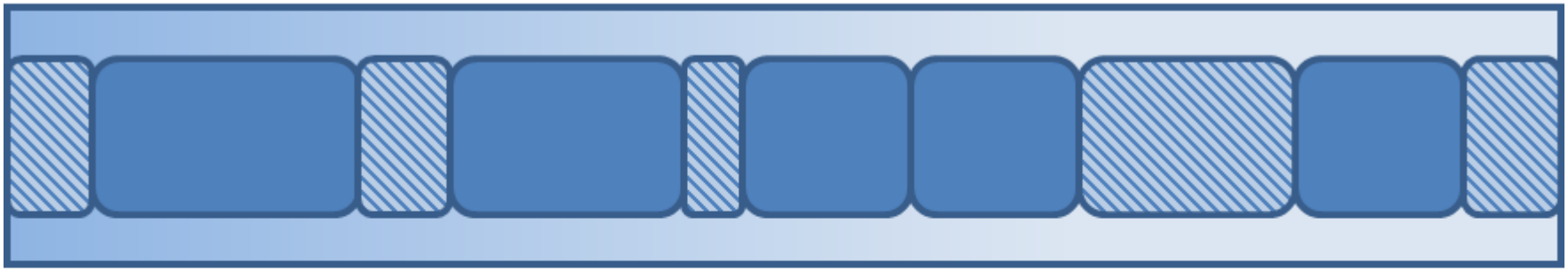


Compaction

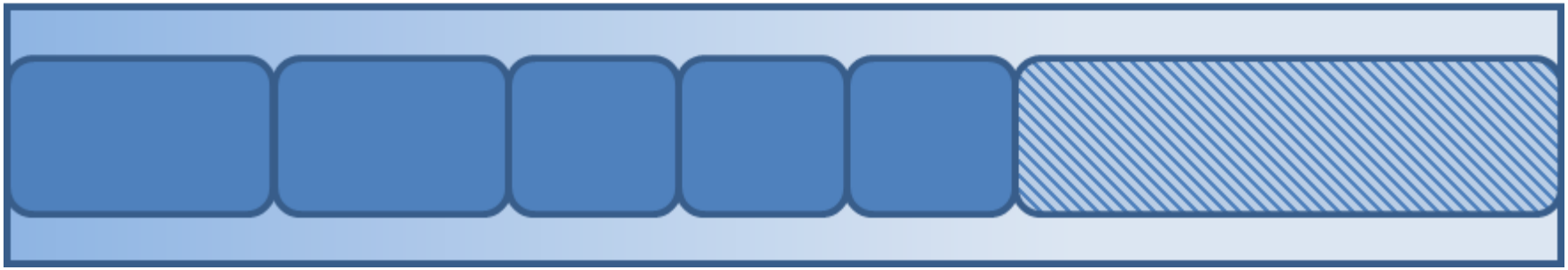
- Стратегия за местене по време на GC
- Освобождава по-големи парчета памет
- Stop-the-world операция, влияе на performance
- Колкото по-популярен е обект, толкова по-голяма пауза

Compaction

Fragmented Heap



Heap after Compacting



Java екосистемата

Java се разработва отворено в OpenJDK проекта

Дефинирането на всички спецификации става през Java Community process

Java екосистемата

Java се разработва отворено в OpenJDK проекта

Дефинирането на всички спецификации става през Java Community process

JDK не е всичко!

Java екосистемата

Java се разработва отворено в OpenJDK проекта

Дефинирането на всички спецификации става през Java Community process

JDK не е всичко!

OpenSource обществата играят голяма роля в екосистемата на Java

Eclipse - 240+ проекта, инструмента, основно java

Apache - 200+ проекта, не само java

Включи се и ти

Стани член на JUG

http://java-bg.org

https://groups.google.com/forum/#!forum/bg-jug

Включи се в Adopt OpenJDK

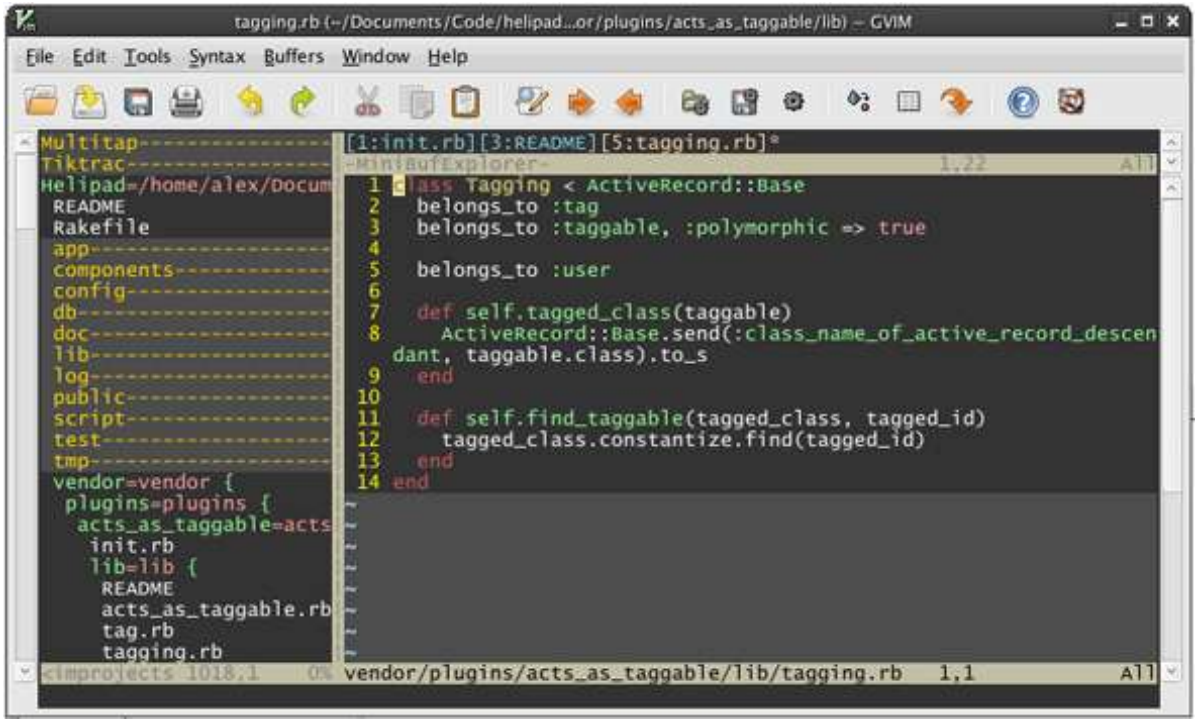
Включи се в Adopt a JSR

Среди за Разработка

Integrated Development Environment

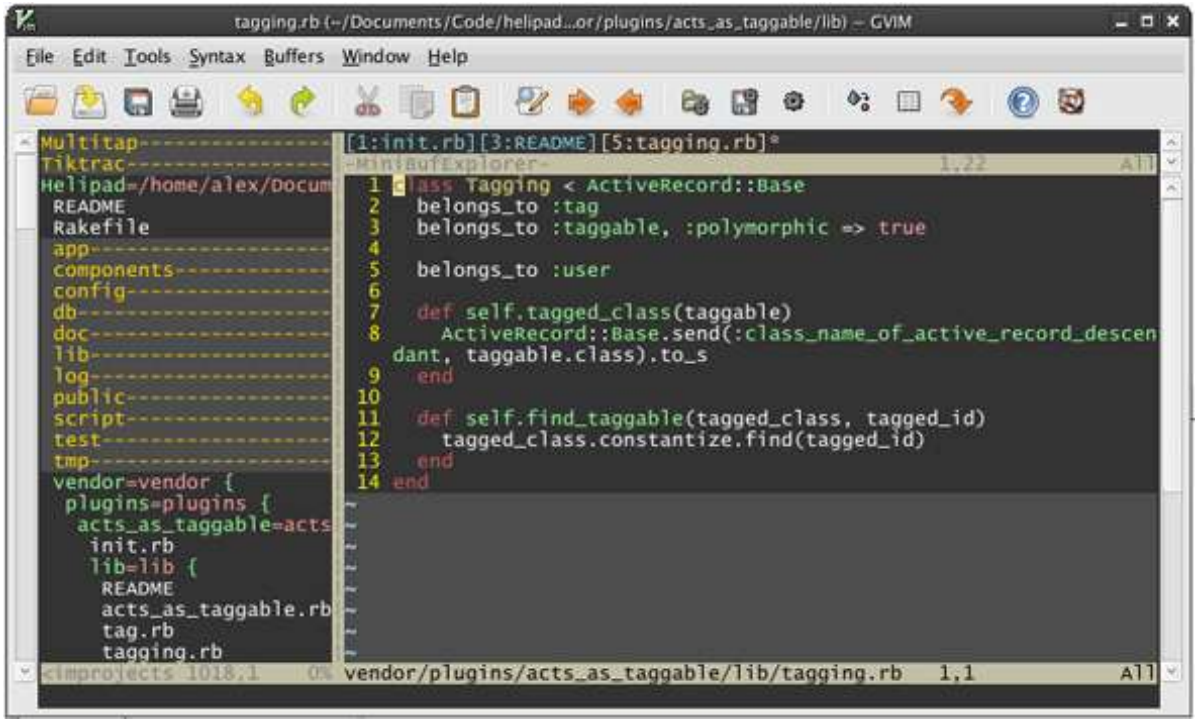
Какво?

- текстов редактор?
  ...много повече



Какво?

- текстов редактор?
  ...много повече



- integrated
- Целта е да правят живота по-лесен,
  не обратното (понякога се получава :)

Защо е важно?

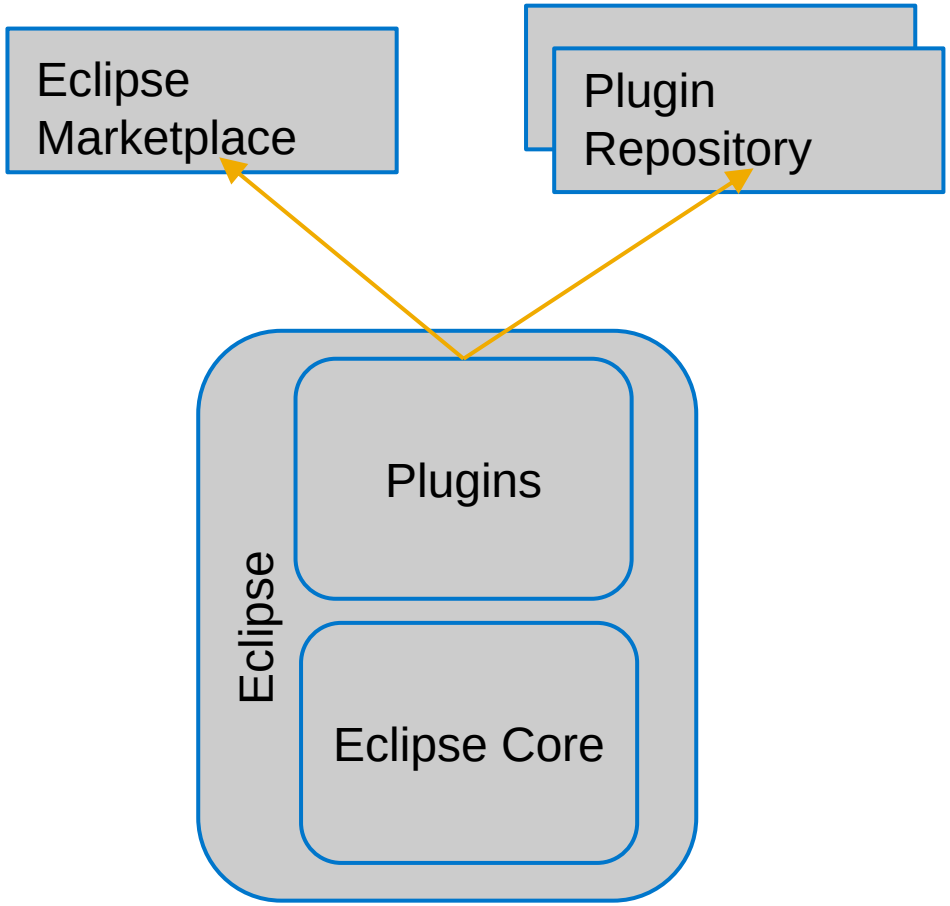
- Продукт, който събира всичко нужно за работа по проект
- Скрива сложността на отделните инструменти
- Снижава входната бариера за учещите се

Популярни IDE-та

Полезни неща

- Project navigation
- Error highlighting
- Code Completion
- Code Coloring
- Code Refactoring
- Code Formatting

Архитектура



Plugin Repository

- Канал за дистрибуция
- Централизирано
- Индивидуално
- Всеки може да си направи
- Как да го ползваме?

Tips & Tricks

Кой е най-лесния начин да изглеждаме като хакери?

Tips & Tricks

Кой е най-лесния начин да изглеждаме като хакери?

Като знаем яките shortcuts в IDEтата :)

Shortcuts magic!

- ctrl+space
- ctrl+alt+H
- ctrl+shift+G
- ctrl+shift+T
- ctrl+shift+R
- ctrl+shift+O
- ctrl+shift+F
- ctrl+/
- ctrl+shift+U
- ctrl+D
- ctrl+1
- ctrl+O
- alt+shift+R
- alt+up/down
- alt+ctrl+up/down
- alt+shift+up/down

Благодаря за вниманието!

За контакти:

ivan.ivanov@sap.com
borislav.kapukaranov@sap.com
SAP Labs Bulgaria
София 1618
бул. Цар Борис III, 136А
тел: 02 91 57 690

ИЗПОЛЗВАНА ЛИТЕРАТУРА

JVM Specification, version 7

http://docs.oracle.com/javase/specs/jvms/se7/jvms7.pdf

The Architecture of the Java Virtual Machine

http://www.artima.com/insidejvm/ed2/jvm2.html

Byte code basics

http://www.javaworld.com/article/2077233/core-java/bytecode-basics.html
http://www.ibm.com/developerworks/library/it-haggar_bytecode/

Garbage collection

http://www.javaworld.com/article/2078645/java-se/jvm-performance-optimization-part-3-garbage-collection.html