ADVANCED JAVA

Java Virtual Machine

Vivek Shah bonii@di.ku.dk

August 20, 2018

DIKU, University of Copenhagen

OUTLINE

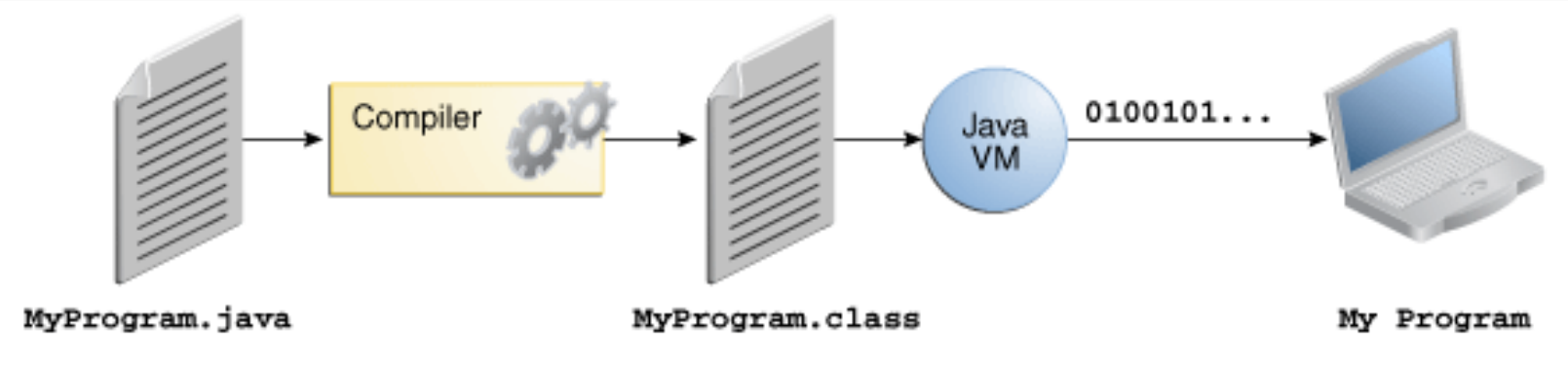
Java Technology Overview

JVM Tuning Options

Profiling and Other Utilities

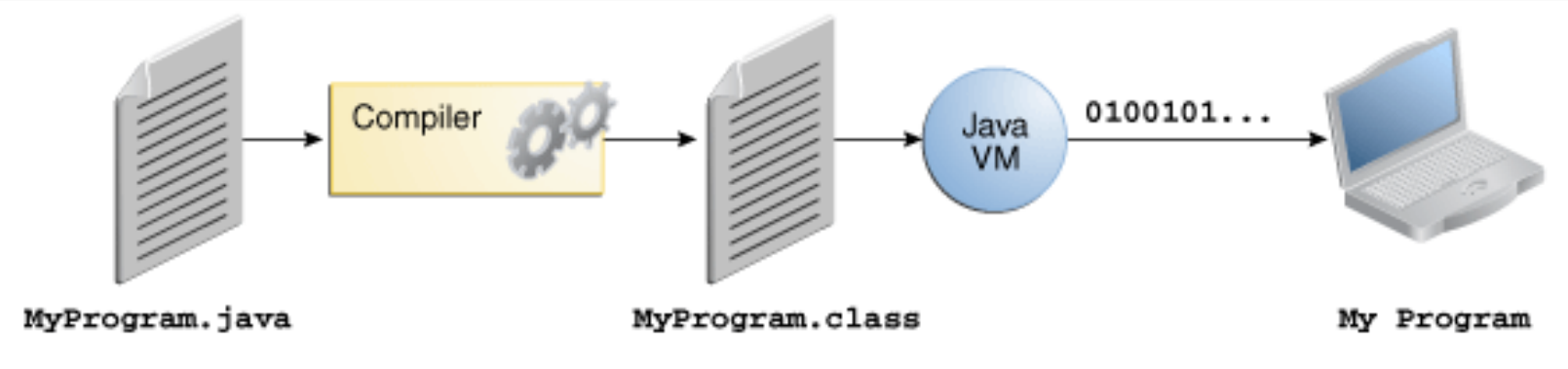
JAVA TECHNOLOGY OVERVIEW

HOW JAVA PROGRAMS ARE EXECUTED?



Overview of Java Development Process (Oracle Docs)

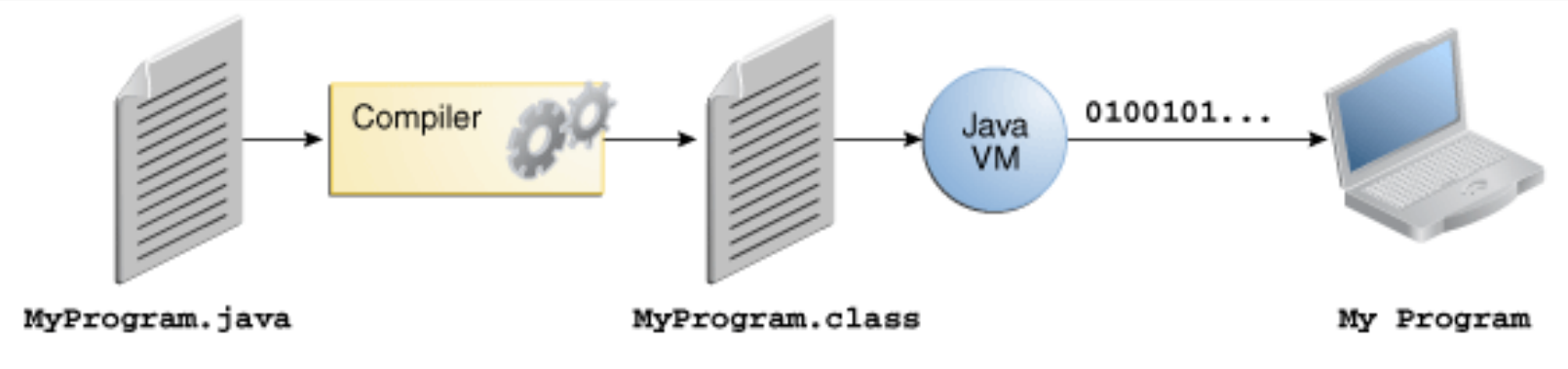
HOW JAVA PROGRAMS ARE EXECUTED?



Overview of Java Development Process (Oracle Docs)

- Source programs are written in plain-text in `.java` files

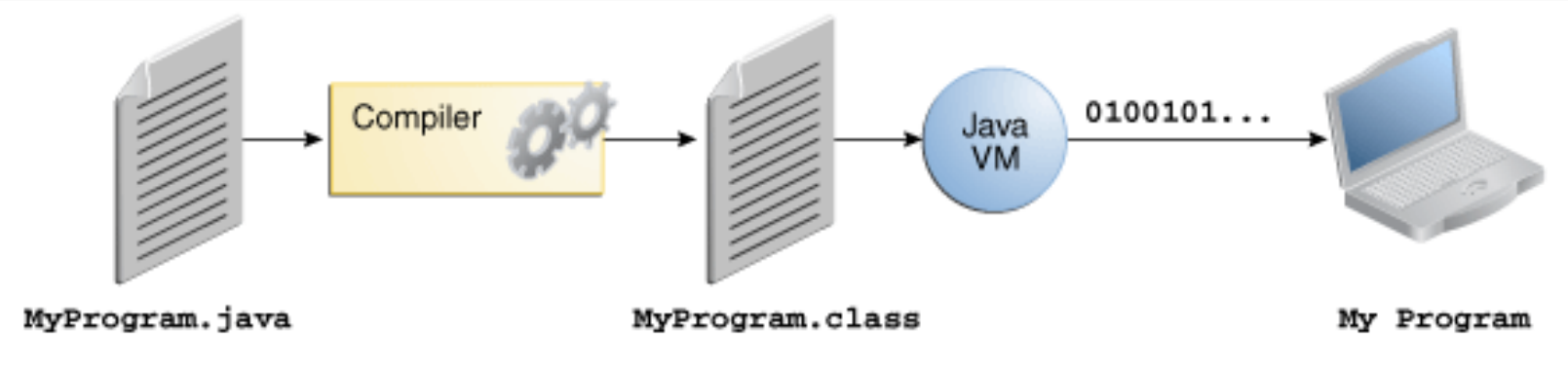
How Java programs are executed?



Overview of Java Development Process (Oracle Docs)

- Source programs are written in plain-text in `.java` files
- Java compiler compiles them into `.class` files (consisting of bytecodes)

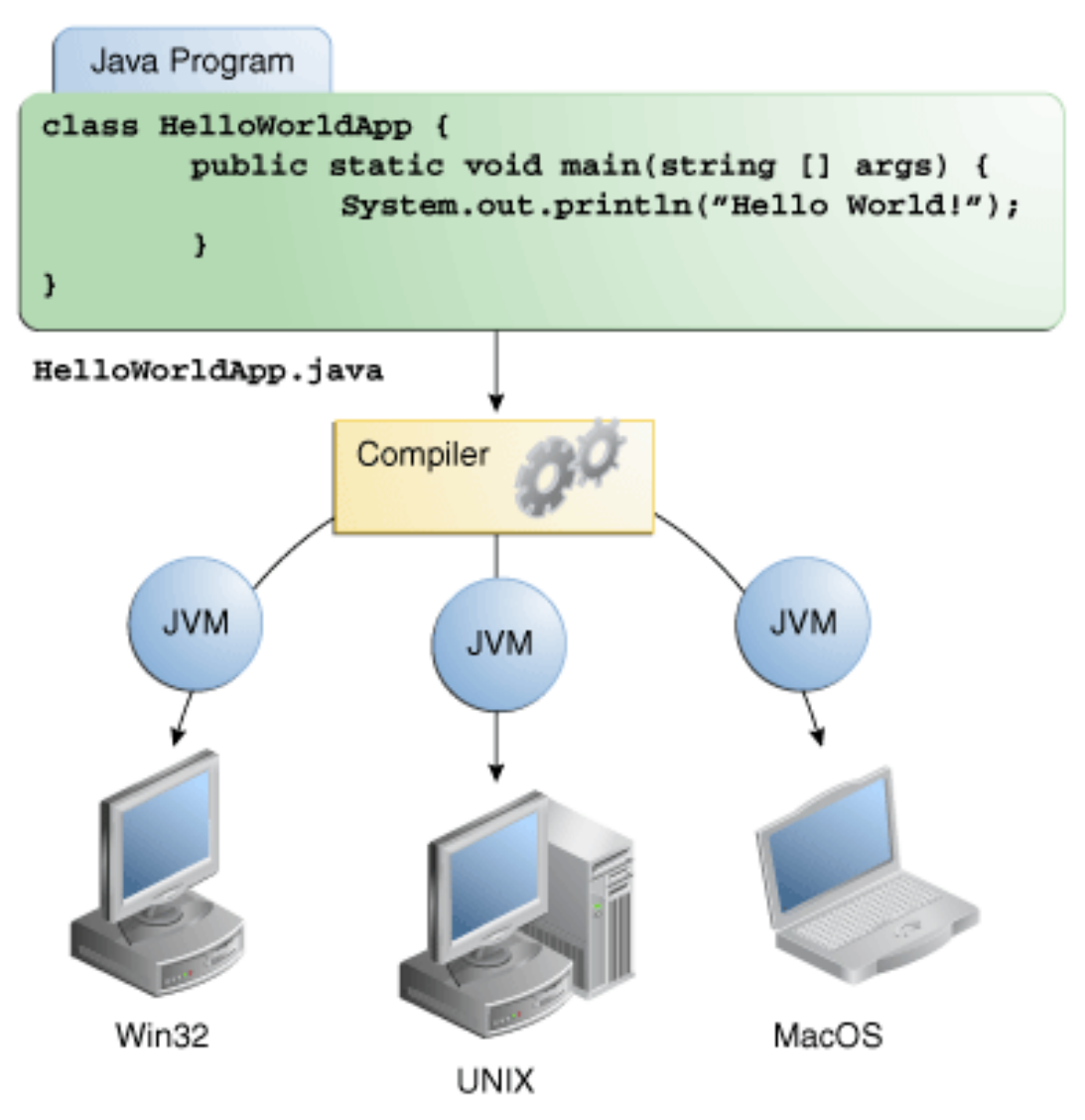
How Java programs are executed?



Overview of Java Development Process (Oracle Docs)

- Source programs are written in plain-text in `.java` files
- Java compiler compiles them into `.class` files (consisting of bytecodes)
- The Java Virtual Machine runs the `.class` files

OVERVIEW OF JVM



Overview of JVM (Oracle Docs)

JVM and Bytecode

- Bytecodes
  - Instruction set of the JVM
  - Both a stack and register machine

JVM and bytecode

- Bytecodes
  - Instruction set of the JVM
  - Both a stack and register machine
- Java Virtual Machine
  - Abstract machine that can execute Java byte-code $\rightarrow$ Java software processor
  - JVM components
    1. Bytecode verifier
    2. Memory manager
    3. Execution engine (Interpreter/JIT compiler)

WHY USE A JVM?

Why do we need a JVM to `interpret` the `.class` file instead of compiling to machine code?

WHY USE A JVM?

1. Platform independence $\rightarrow$ Source and compiled code is independent of hardware and operating system platform
    - Compiled code is just dependent on the JVM platform
2. Allows distribution of binaries (`.class` files)
3. Compilation process is platform independent but compilers are not always
4. Compile once, run anywhere
5. Allows sand-boxing e.g., Applets
6. Allows instrumentation. Dynamic memory management.

Why use a JVM? (contd.)

7. Provides Extensibility
    - Any language that can compile down to bytecode can be interpreted (e.g., Scala, Kotlin)
8. Only JVM needs to be platform dependent. JVM writers can focus on efficient processing of bytecodes instead of the whole upper language stack
9. Has overheads. Various optimizations like JIT compilation, hardware processors supporting bytecodes

JVM TUNING OPTIONS

TUNING THE JVM

- JVM provides a tunable execution enviroment
- Various ways to affect performance of programs by tuning JVM
  1. Memory management (heap tuning and garbage collection)
  2. Code generation (e.g., control inlining of code)
  3. Configuration of enviroment variables using properties
  4. Usage of JVM hooks
- Use limited profiling support to understand program performance
- More than meets the eye (Use extreme care)

JVM TUNING OPTIONS

Meaning of the option prefix

- `-x` $\rightarrow$ non standard across different VM, subject to change without notice
- `-xx` $\rightarrow$ Non standard and not stable

For standard options see the reference pages for the platform

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files
`-d32` or `-d64`	Run in 32 bit or 64 bit enviroment

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files
`-d32` or `-d64`	Run in 32 bit or 64 bit enviroment
`-server`	Turns on optimizing JIT

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files
`-d32` or `-d64`	Run in 32 bit or 64 bit enviroment
`-server`	Turns on optimizing JIT
`-verbose`	Runs in verbose mode

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files
`-d32` or `-d64`	Run in 32 bit or 64 bit enviroment
`-server`	Turns on optimizing JIT
`-verbose`	Runs in verbose mode
`-jar`	Execute a program in a jar file

FEW COMMONLY USED GENERIC JVM TUNING OPTIONS

Option	Meaning
`-classpath`	Directories to search for class files
`-d32` or `-d64`	Run in 32 bit or 64 bit enviroment
`-server`	Turns on optimizing JIT
`-verbose`	Runs in verbose mode
`-jar`	Execute a program in a jar file
`-Xint`	Turns off compilation to native code, only interpretation

Few JVM memory tuning options (heap and stack)

Option	Meaning

Few JVM memory tuning options (heap and stack)

Option	Meaning
`-Xms`	Initial size of heap e.g., `-Xms1024m`

Few JVM memory tuning options (heap and stack)

Option	Meaning
`-Xms`	Initial size of heap e.g., `-Xms1024m`
`-Xmx`	Maximum size of heap e.g., `-Xmx8192m`

Few JVM memory tuning options (heap and stack)

Option	**Meaning**
`-Xms`	Initial size of heap e.g., `-Xms1024m`
`-Xmx`	Maximum size of heap e.g., `-Xmx8192m`
`-Xss`	Sets the stack size, `-Xss4096k` ← this value will be multiplied by the number of threads

GARBAGE COLLECTOR TUNING OPTIONS

- Tuning garbage collection is not only about setting parameters but understanding the process
- Generational garbage collection
- Can choose the type of garbage collector
- Blocking versus Incremental versus Concurrent

FEW GARBAGE COLLECTOR TUNING OPTIONS

Option	Meaning

FEW GARBAGE COLLECTOR TUNING OPTIONS

Option	Meaning
`-XX:+UseSerialGC`	Enables the use of the serial garbage collector (Good when running many JVM's)

FEW GARBAGE COLLECTOR TUNING OPTIONS

Option	Meaning
`-XX:+UseSerialGC`	Enables the use of the serial garbage collector (Good when running many JVM's)
`-XX:+UseParallelGC`	Enables the use of the parallel scavenge garbage collector (also known as the throughput collector)

FEW GARBAGE COLLECTOR TUNING OPTIONS

Option	Meaning
`-XX:+UseSerialGC`	Enables the use of the serial garbage collector (Good when running many JVM's)
`-XX:+UseParallelGC`	Enables the use of the parallel scavenge garbage collector (also known as the throughput collector)
`-XX:+UseParallelOldGC`	Enables the use of the parallel garbage collector for full GCs (heavy duty, long pauses)

Few garbage collector tuning options (contd.)

Option	Meaning

Few garbage collector tuning options (contd.)

Option	Meaning
`-XX:+UseConcMarkSweepGC`	Enables the use of the CMS garbage collector for the old generation (short pauses)

Few garbage collector tuning options (contd.)

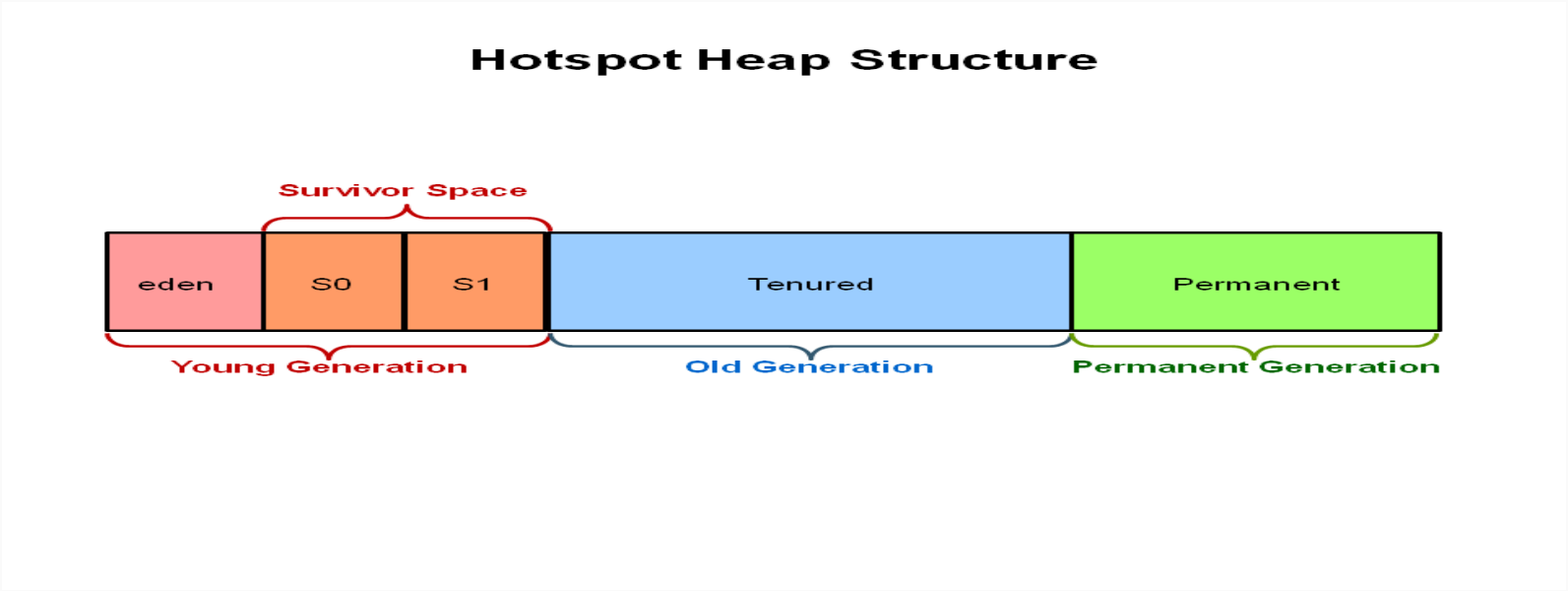
Option	Meaning
`-XX:+UseConcMarkSweepGC`	Enables the use of the CMS garbage collector for the old generation (short pauses)
`-XX:+UseG1GC`	Enables the use of the garbage-first (G1) garbage collector (better CMS)

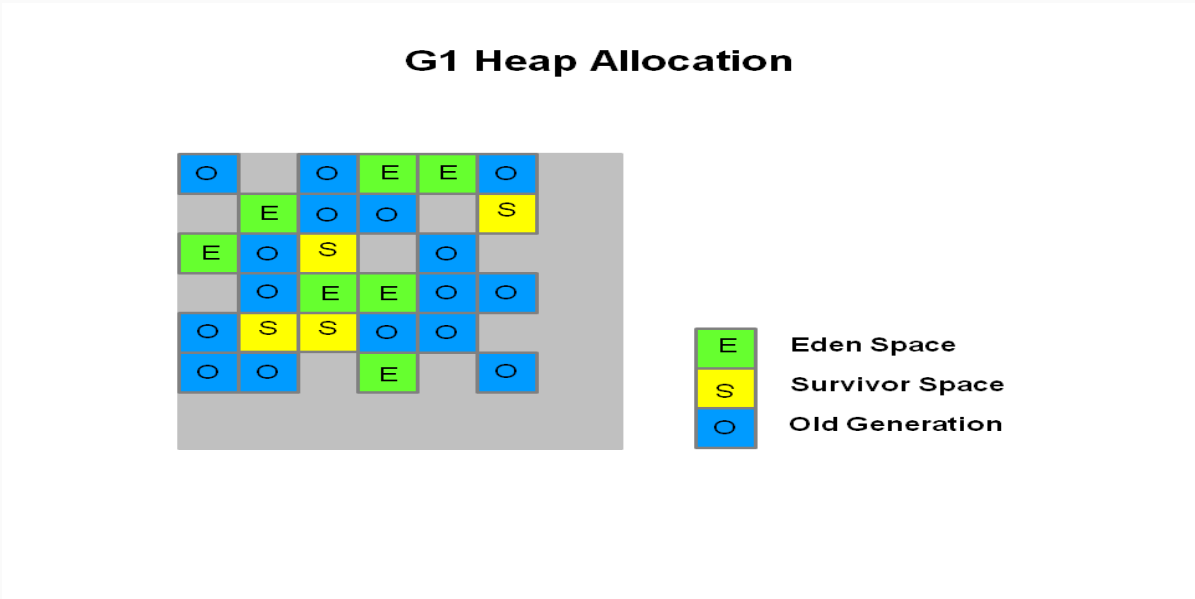
Few garbage collector tuning options (contd.)

Option	Meaning
`-XX:+UseConcMarkSweepGC`	Enables the use of the CMS garbage collector for the old generation (short pauses)
`-XX:+UseG1GC`	Enables the use of the garbage-first (G1) garbage collector (better CMS)

Use these options only if you understand the process and its impact

HEAP STRUCTURES FOR GARBAGE COLLECTION





FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning

FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning
`-XX:NewSize`	Size of new generation

FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning
`-XX:NewSize`	Size of new generation
`-XX:MaxNewSize`	Maximum size of new generation

FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning
`-XX:NewSize`	Size of new generation
`-XX:MaxNewSize`	Maximum size of new generation
`-XX:NewRatio`	Ratio of old/new generations

FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning
`-XX:NewSize`	Size of new generation
`-XX:MaxNewSize`	Maximum size of new generation
`-XX:NewRatio`	Ratio of old/new generations
`-XX:SurviverRatio`	Ratio of eden/survivor space

FEW GARBAGE COLLECTOR HEAP TUNING OPTIONS

Option	Meaning
`-XX:NewSize`	Size of new generation
`-XX:MaxNewSize`	Maximum size of new generation
`-XX:NewRatio`	Ratio of old/new generations
`-XX:SurviverRatio`	Ratio of eden/survivor space

Lots more !! Use these options only if you understand them

Few code generation tuning options (inlining)

Option	Meaning

Few code generation tuning options (inlining)

Option	Meaning
`-XX:MaxInlineSize`	Sets the maximum size in bytes of bytecode that will be inlined

Few code generation tuning options (inlining)

Option	Meaning
`-XX:MaxInlineSize`	Sets the maximum size in bytes of bytecode that will be inlined
`-XX:FreqInlineSize`	Sets the maximum size in bytes of bytecode of frequently executed method that will be inlined

Few code generation tuning options (inlining)

Option	Meaning
`-XX:MaxInlineSize`	Sets the maximum size in bytes of bytecode that will be inlined
`-XX:FreqInlineSize`	Sets the maximum size in bytes of bytecode of frequently executed method that will be inlined
`-XX:LoopUnrollLimit=n`	Unrolls loops with a compiler internal representation node count of less than n

CONFIGURING PROPERTIES

- Properties passed using `-Dproperty=p`
- Retrieve the property value using `System.getProperty(p)`
- Some property keys are already predefined from which certain environment values can be retrieved
- Use `.properties` file instead of long list of properties in command line

PROFILING AND OTHER UTILITIES

PROFILING

- Profiling is done in order to understand performance trade-offs
  - Use a profiler to understand your code better
- `-Xprof` $\rightarrow$ option for some basic profiling

PROFILING

- Profiling is done in order to understand performance trade-offs
  - Use a profiler to understand your code better
- `-Xprof` $\rightarrow$ option for some basic profiling
- JVM Monitor has nice integration with Eclipse
- Most full featured profilers are commercial

1. jdb - Java debugger
2. javap - Java disassembler
3. javadoc - Java document generator