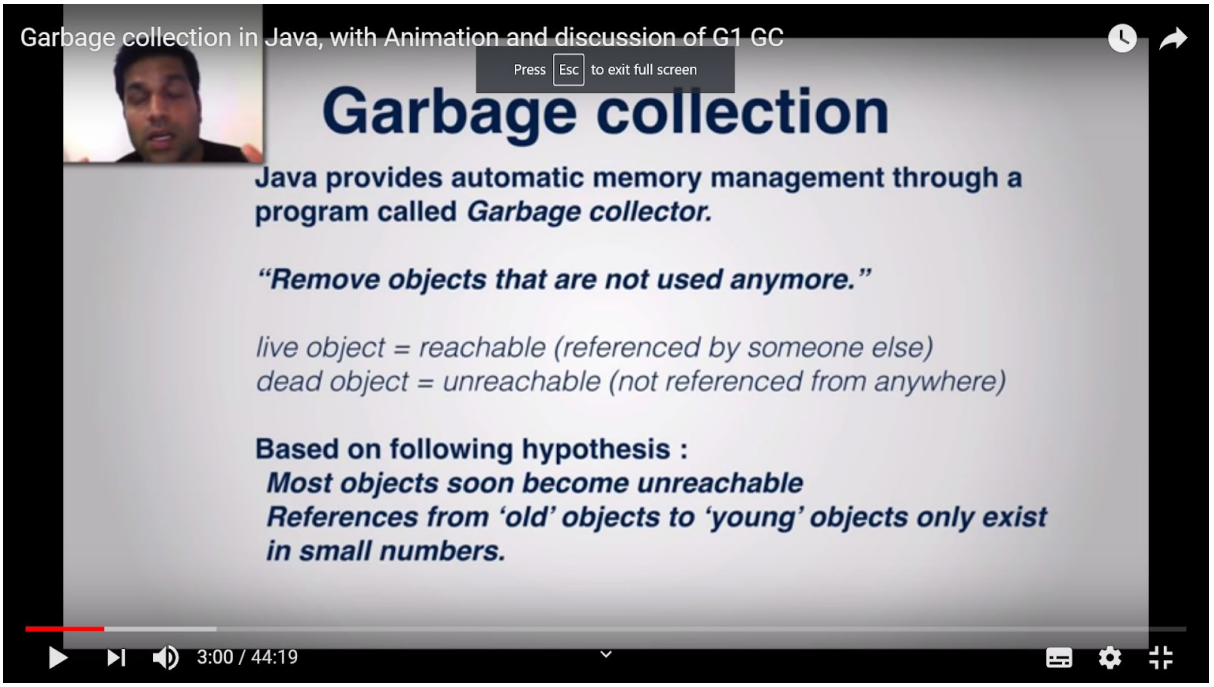
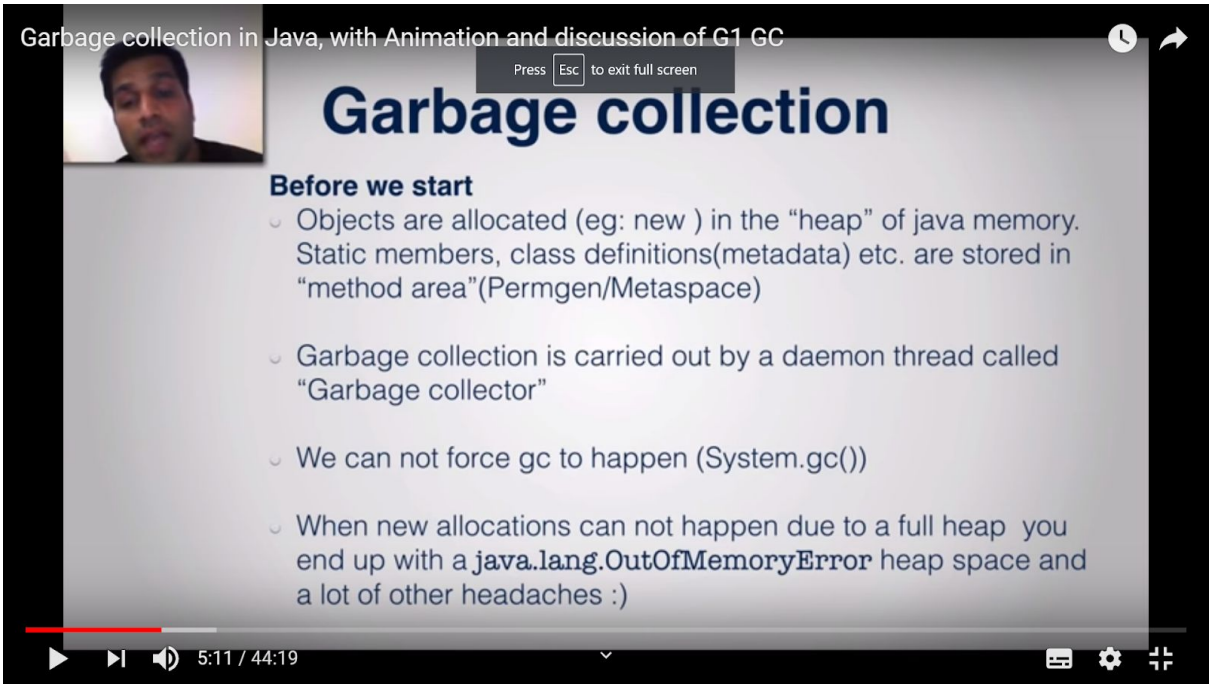
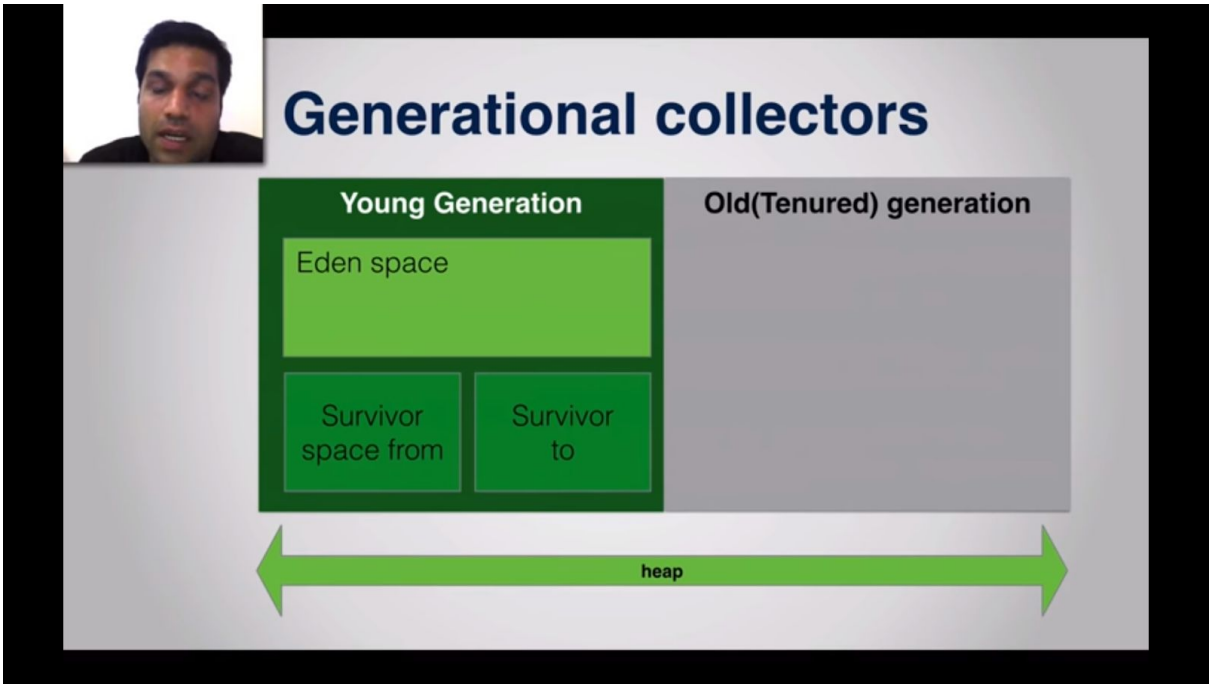
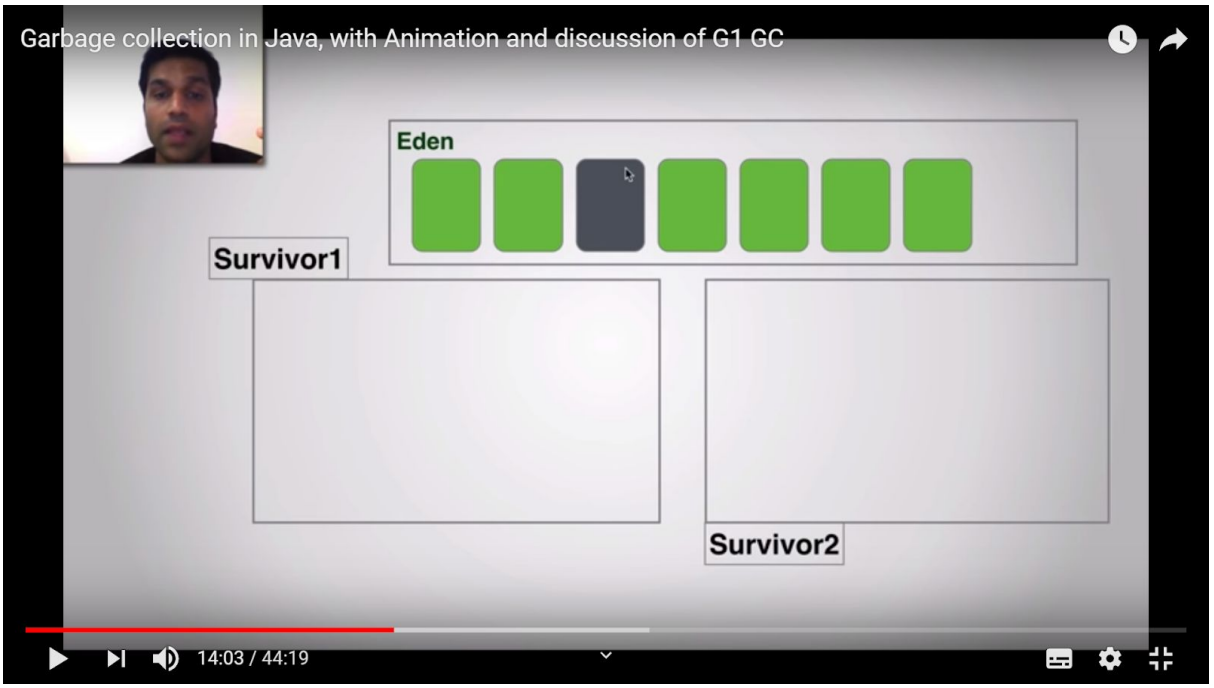
Garbage Collection :

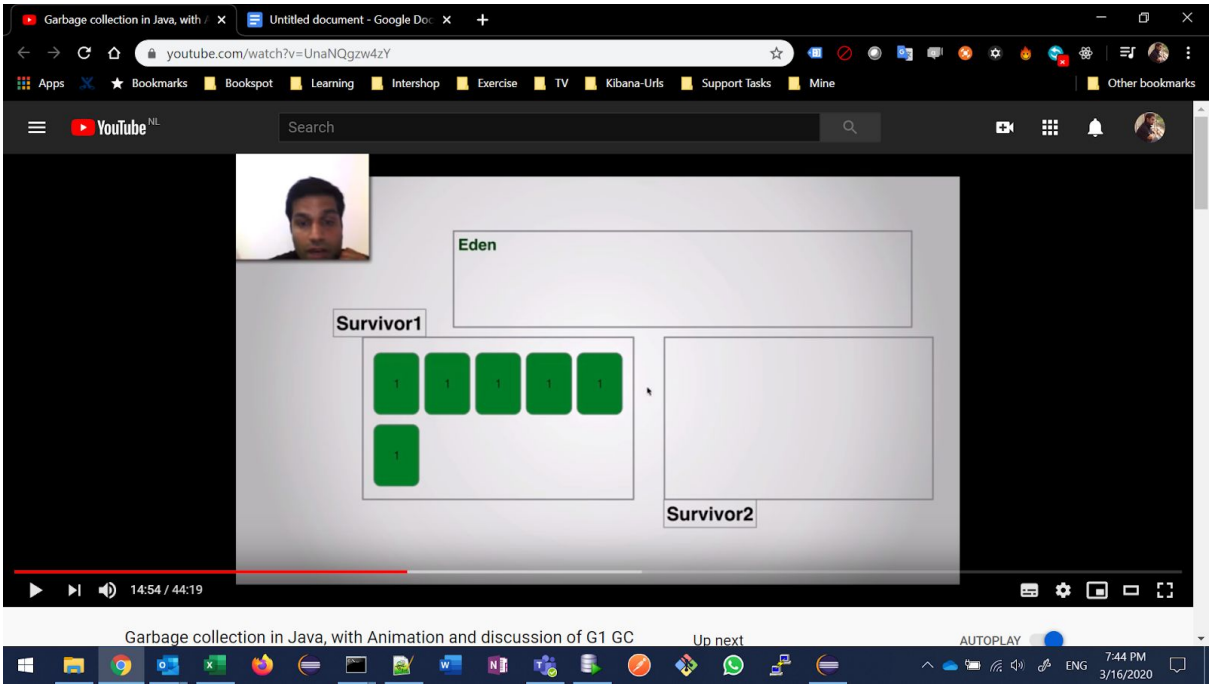


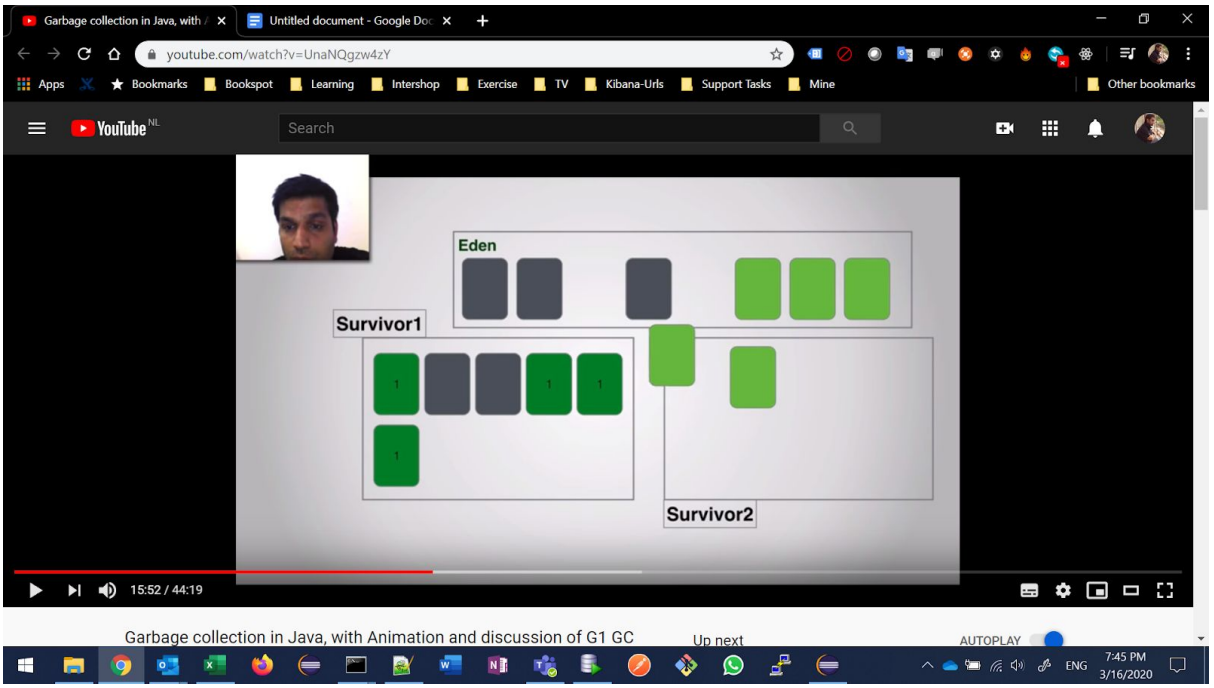


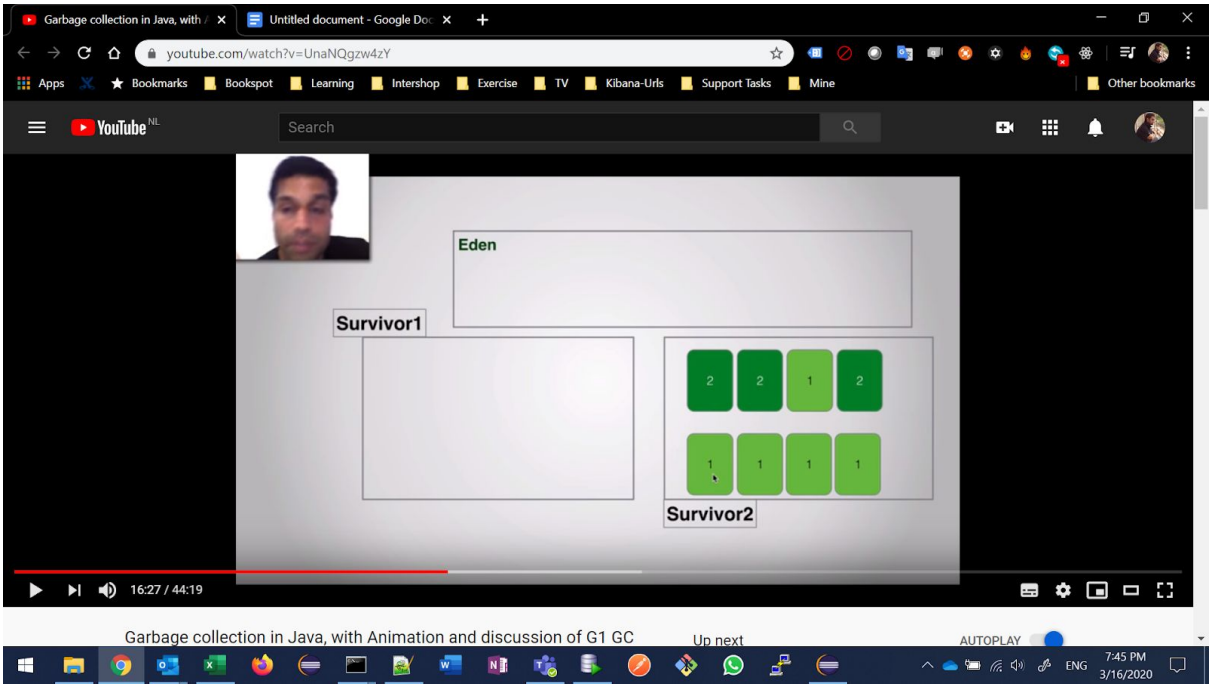

Garbage collection
Involves
Mark: Starts from root node of your application(main), walks the object graph, marks objects that are reachable as live.
Delete/sweep: Delete unreachable objects
Compacting: Compact the memory by moving around the objects and making the allocation contiguous than fragmented.

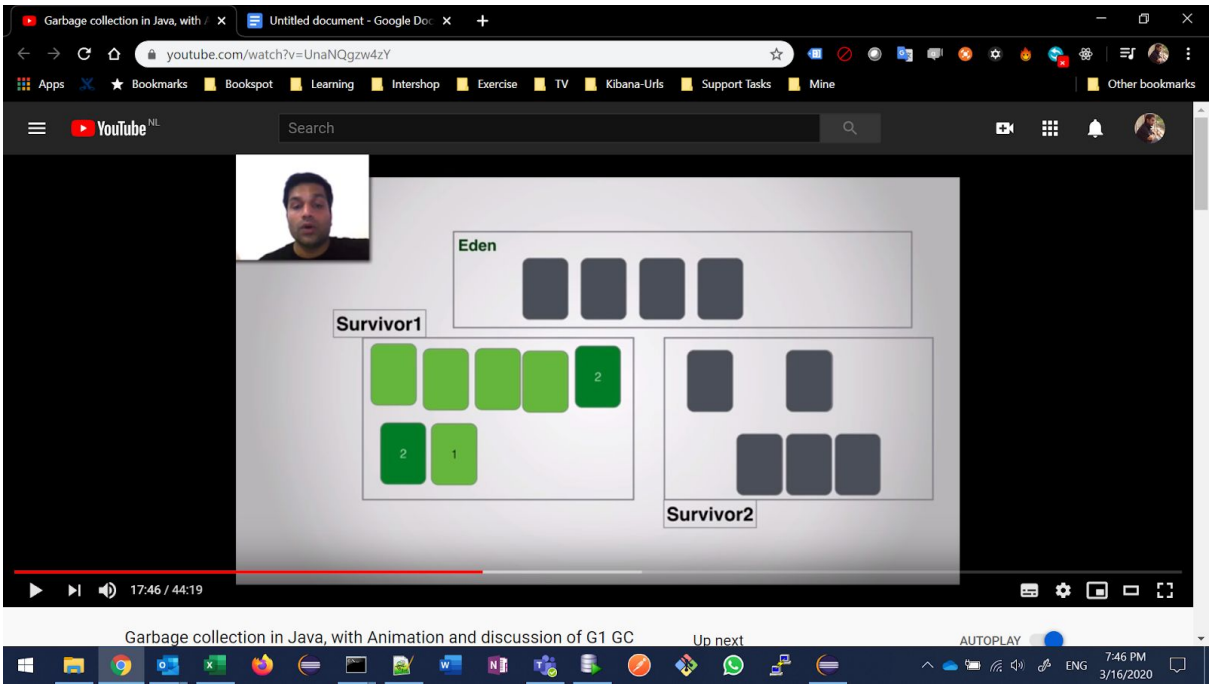

Generational collectors
Young Generation
Eden space
Survivor space from
Survivor to
Old(Tenured) generation
heap

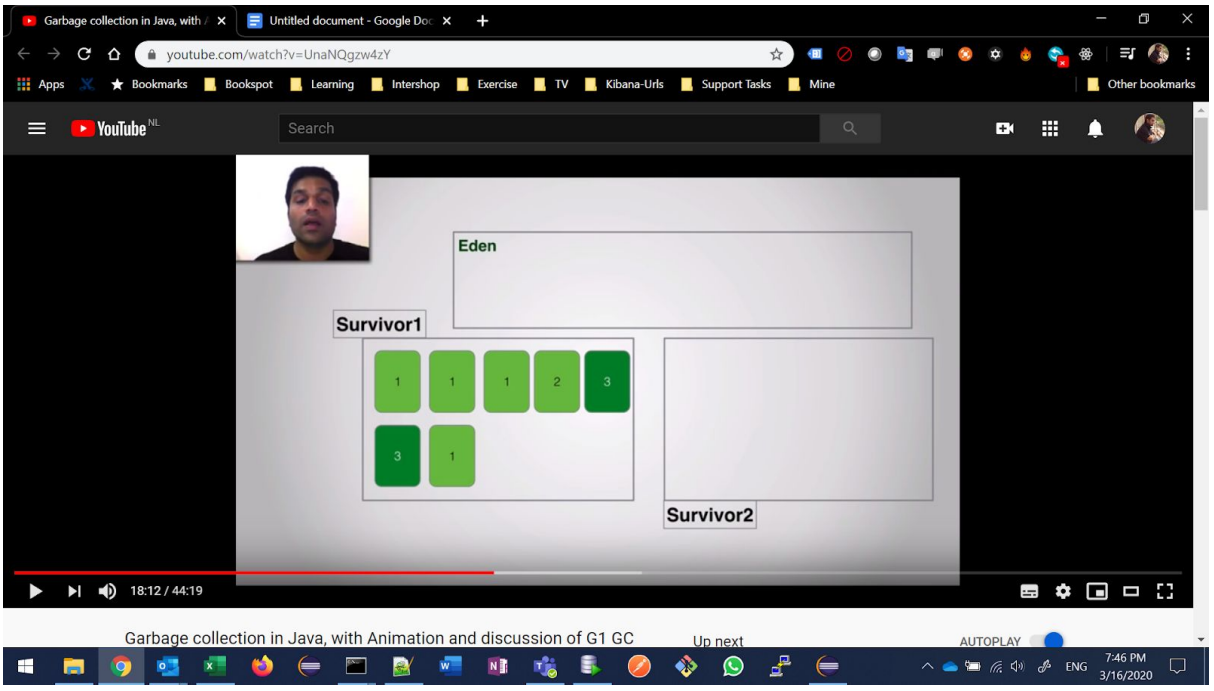

Garbage collection in Java, with Animation and discussion of G1 GC
Eden
Survivor1
Survivor2
14:03 / 44:19

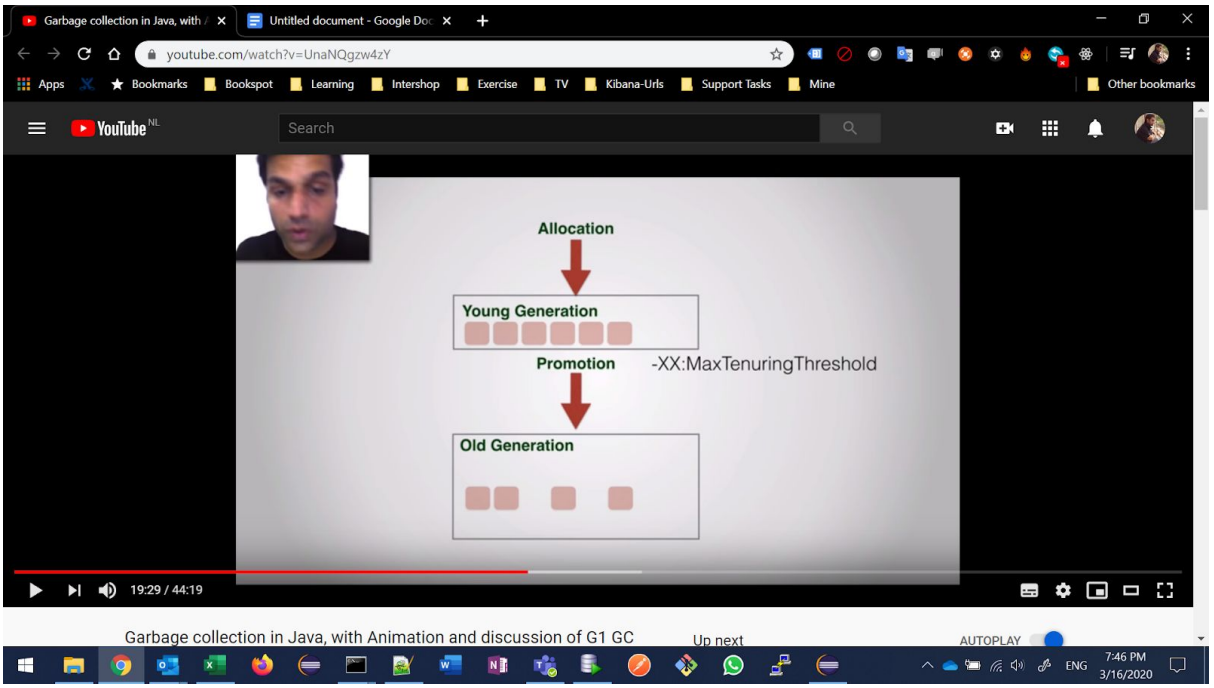

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Eden
Survivor1
Survivor2
14:54 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:44 PM
3/16/2020

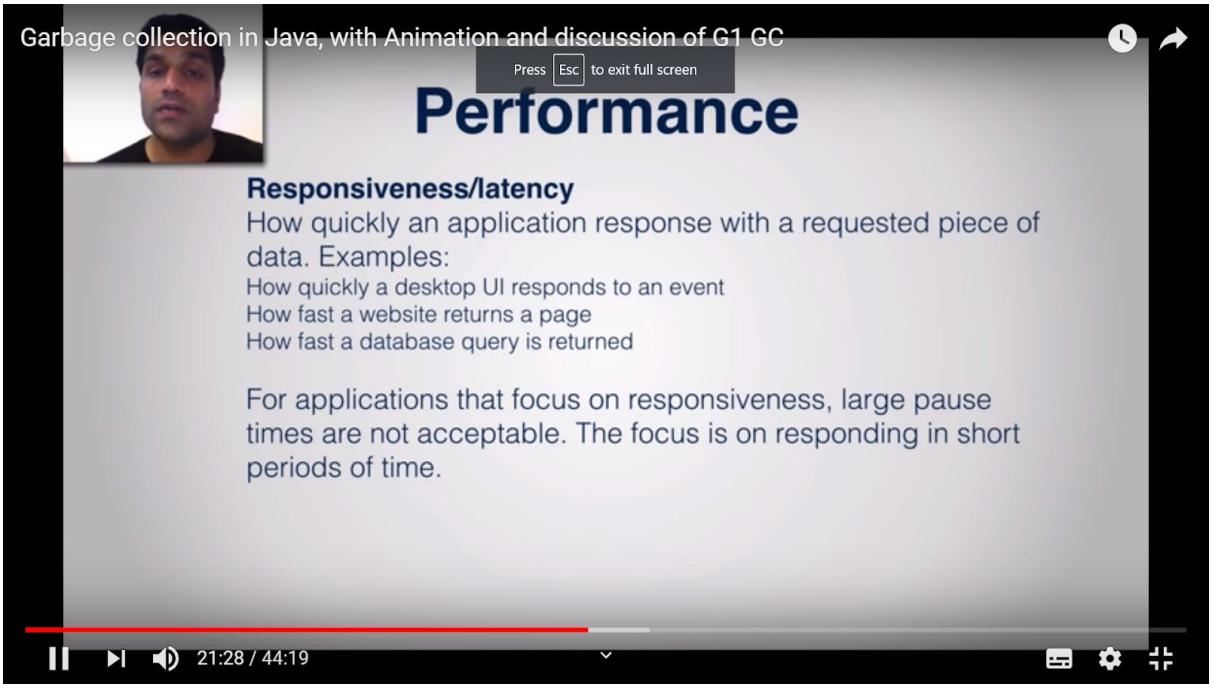

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Eden
Survivor1
Survivor2
15:52 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:45 PM
3/16/2020

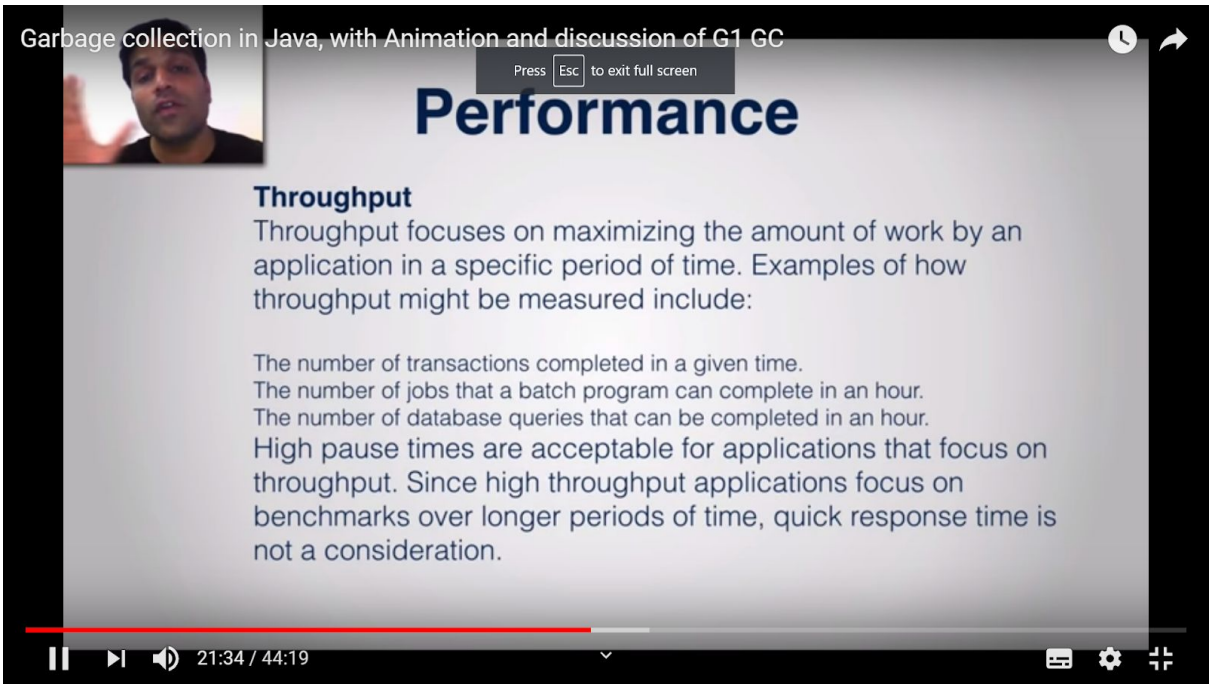

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Eden
Survivor1
Survivor2
16:27 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:45 PM
3/16/2020

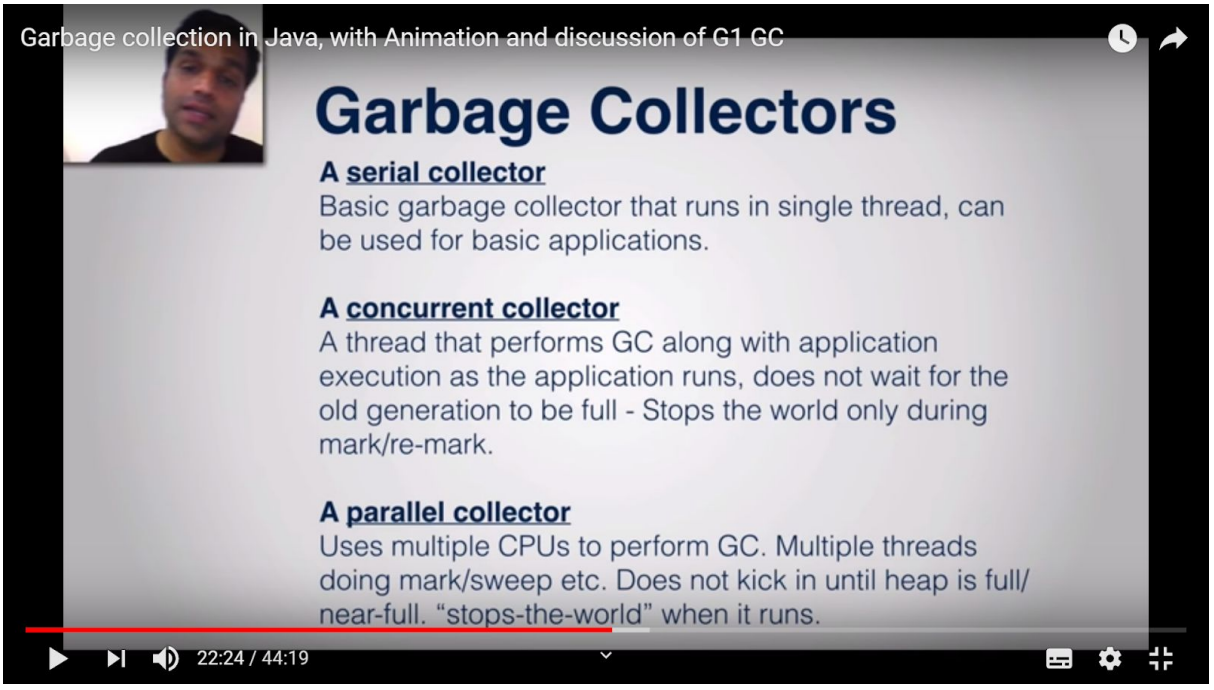

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Eden
Survivor1
2
2
1
Survivor2
17:46 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:46 PM
3/16/2020

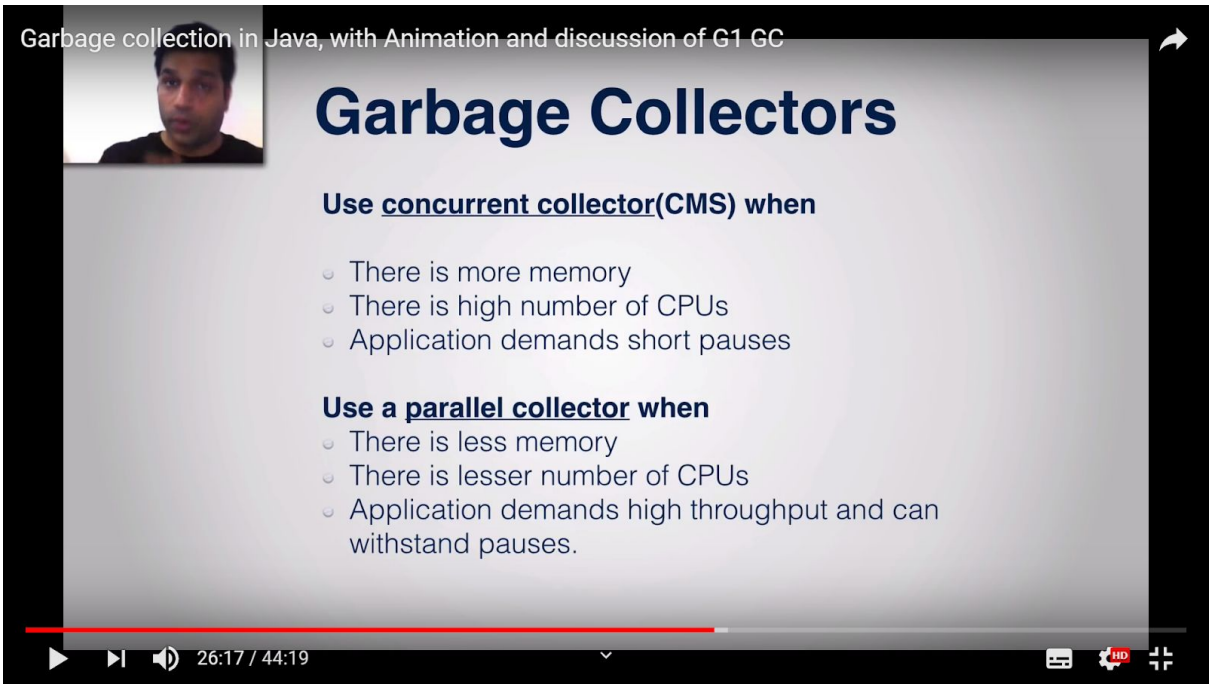

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Eden
Survivor1
1
1
1
2
3
3
1
Survivor2
18:12 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:46 PM
3/16/2020

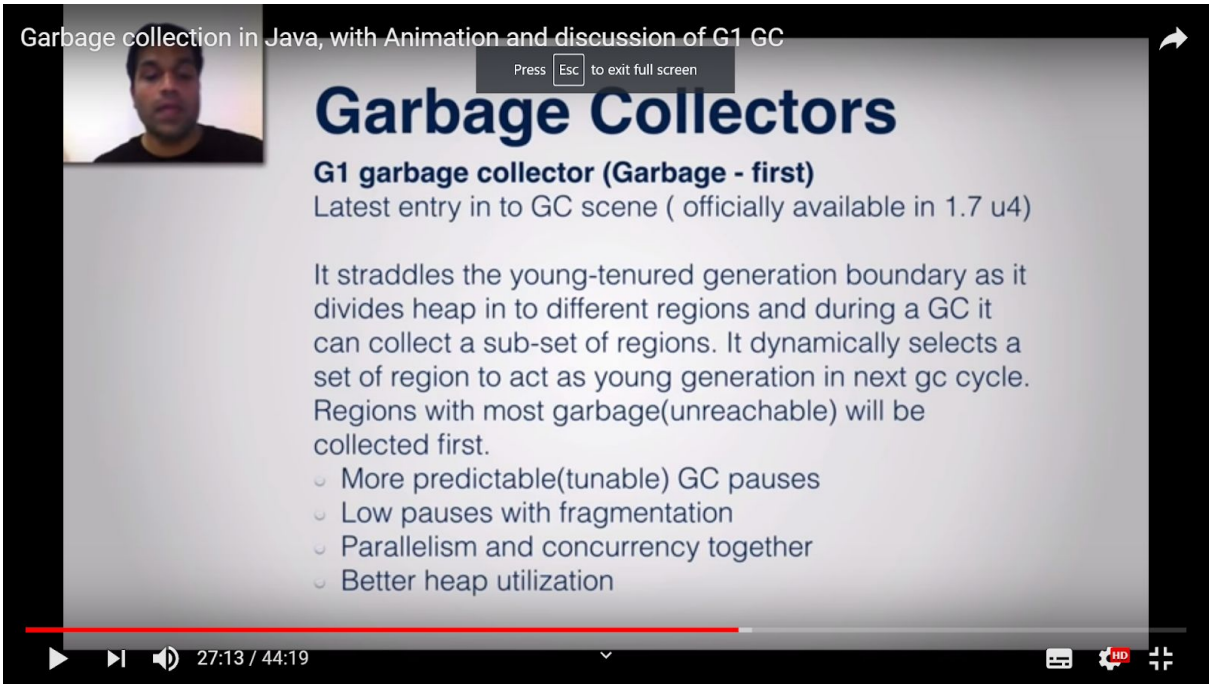

Garbage collection in Java, with
Untitled document - Google Doc
youtube.com/watch?v=UnaNQgzw4zY
Apps
Bookmarks
Bookspot
Learning
Intershop
Exercise
TV
Kibana-Urls
Support Tasks
Mine
Other bookmarks
YouTube
Search
Allocation
Young Generation
Promotion
-XX:MaxTenuringThreshold
Old Generation
19:29 / 44:19
Garbage collection in Java, with Animation and discussion of G1 GC
Up next
AUTOPLAY
ENG
7:46 PM
3/16/2020


Garbage collection in Java, with Animation and discussion of G1 GC
Press Esc to exit full screen
Performance
Responsiveness/latency
How quickly an application response with a requested piece of data. Examples:
How quickly a desktop UI responds to an event
How fast a website returns a page
How fast a database query is returned
For applications that focus on responsiveness, large pause times are not acceptable. The focus is on responding in short periods of time.
21:28 / 44:19

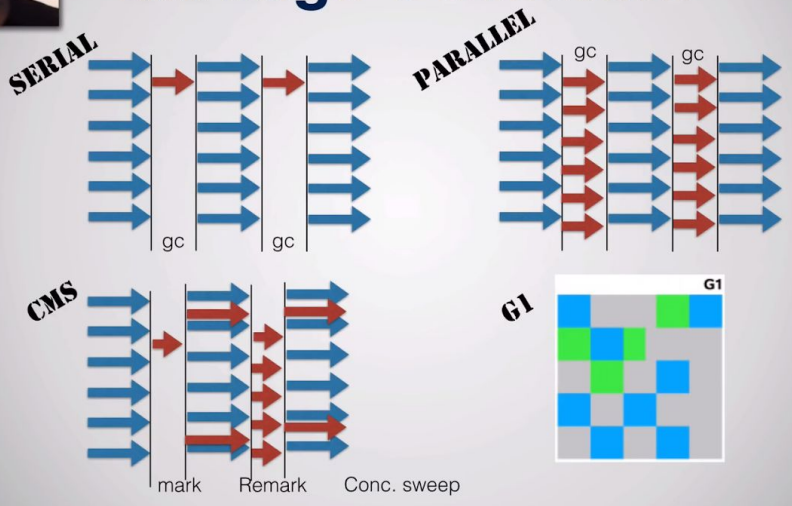

Garbage collection in Java, with Animation and discussion of G1 GC
Press Esc to exit full screen
Performance
Throughput
Throughput focuses on maximizing the amount of work by an application in a specific period of time. Examples of how throughput might be measured include:
The number of transactions completed in a given time.
The number of jobs that a batch program can complete in an hour.
The number of database queries that can be completed in an hour.
High pause times are acceptable for applications that focus on throughput. Since high throughput applications focus on benchmarks over longer periods of time, quick response time is not a consideration.
21:34 / 44:19


Garbage collection in Java, with Animation and discussion of G1 GC
Garbage Collectors
A serial collector
Basic garbage collector that runs in single thread, can be used for basic applications.
A concurrent collector
A thread that performs GC along with application execution as the application runs, does not wait for the old generation to be full - Stops the world only during mark/re-mark.
A parallel collector
Uses multiple CPUs to perform GC. Multiple threads doing mark/sweep etc. Does not kick in until heap is full/ near-full. “stops-the-world” when it runs.
22:24 / 44:19


Garbage collection in Java, with Animation and discussion of G1 GC
Garbage Collectors
Use concurrent collector(CMS) when
There is more memory
There is high number of CPUs
Application demands short pauses
Use a parallel collector when
There is less memory
There is lesser number of CPUs
Application demands high throughput and can withstand pauses.
26:17 / 44:19


Garbage collection in Java, with Animation and discussion of G1 GC
Press Esc to exit full screen
Garbage Collectors
G1 garbage collector (Garbage - first)
Latest entry in to GC scene (officially available in 1.7 u4)
It straddles the young-tenured generation boundary as it divides heap in to different regions and during a GC it can collect a sub-set of regions. It dynamically selects a set of region to act as young generation in next gc cycle. Regions with most garbage(unreachable) will be collected first.
More predictable(tunable) GC pauses
Low pauses with fragmentation
Parallelism and concurrency together
Better heap utilization
27:13 / 44:19

Garbage Collectors



Selecting Garbage collector

Option	Description
-XX:+UseSerialGC	Single-threaded gc on young and old generation. To be used only on small heaps
-XX:+UseParallelGC	Young generation uses parallel gc, Old generation uses single-threaded gc
-XX:+UseParallelOldGC	Both young and old generations have multi-threaded GC
-XX:+UseParNewGC	Multi-threaded young generation garbage collector
-XX:+UseConcMarkSweepGC	Enables concurrent collector. Autoenables ParNewGC by default.
-XX:+UseG1GC	Use G1

Defaults: 1.6 = Parallel, 1.7 G1

Tune the Heap

-Xmsvalue

-Xmxvalue (def: 256m)

-XX:NewRatio=ratio (2 means 1/3 Y +2/3 T)

-XX:NewSize=size

-XX:MaxNewSize=size

-XX:PermSize

-XX:MaxPermSize (def: 64m)

GC logging

-verbose:gc
-XX:+PrintGCDetails
-Xloggc:gc.log

Very useful if gc is the suspect. Use graphical tool to analyze the logs.

View GC

jvisualvm with visual gc plugin