


# JVM Performance and Tuning

## Module Four Threads and Garbage Collection





A top-down view of a wooden desk. In the top right corner, a portion of a silver laptop is visible, showing keys like 'tab', 'Q', 'W', 'E', 'caps lock', 'A', 'S', 'Z', 'fn', 'control', and 'option'. Below the laptop, a pair of black-rimmed glasses lies horizontally. To the right of the glasses is a white ceramic cup filled with dark coffee, with a yellow handle. In the top center, a small green succulent plant in a dark pot sits on the desk. The background is a light-colored wooden surface with a prominent grain pattern.

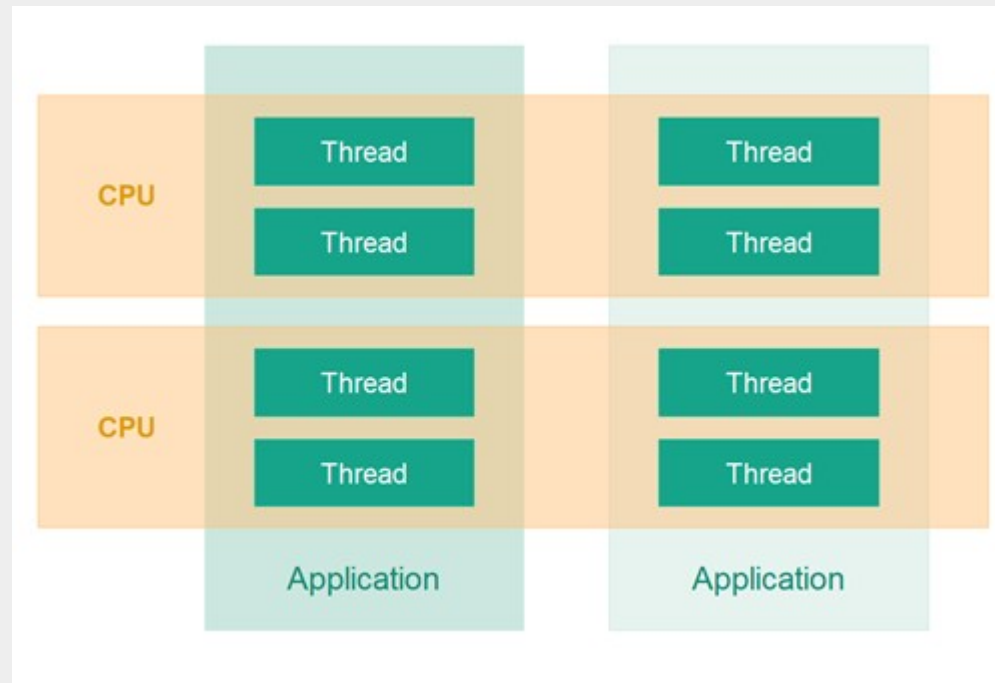
# Module Topics

**1. Threads**

**2. Garbage Collection**

# Threads

- A CPU may have multiple threads of execution
  - Each thread has exclusive access to the CPU for a time slice
  - In a multi-core environment, a thread may use different cores

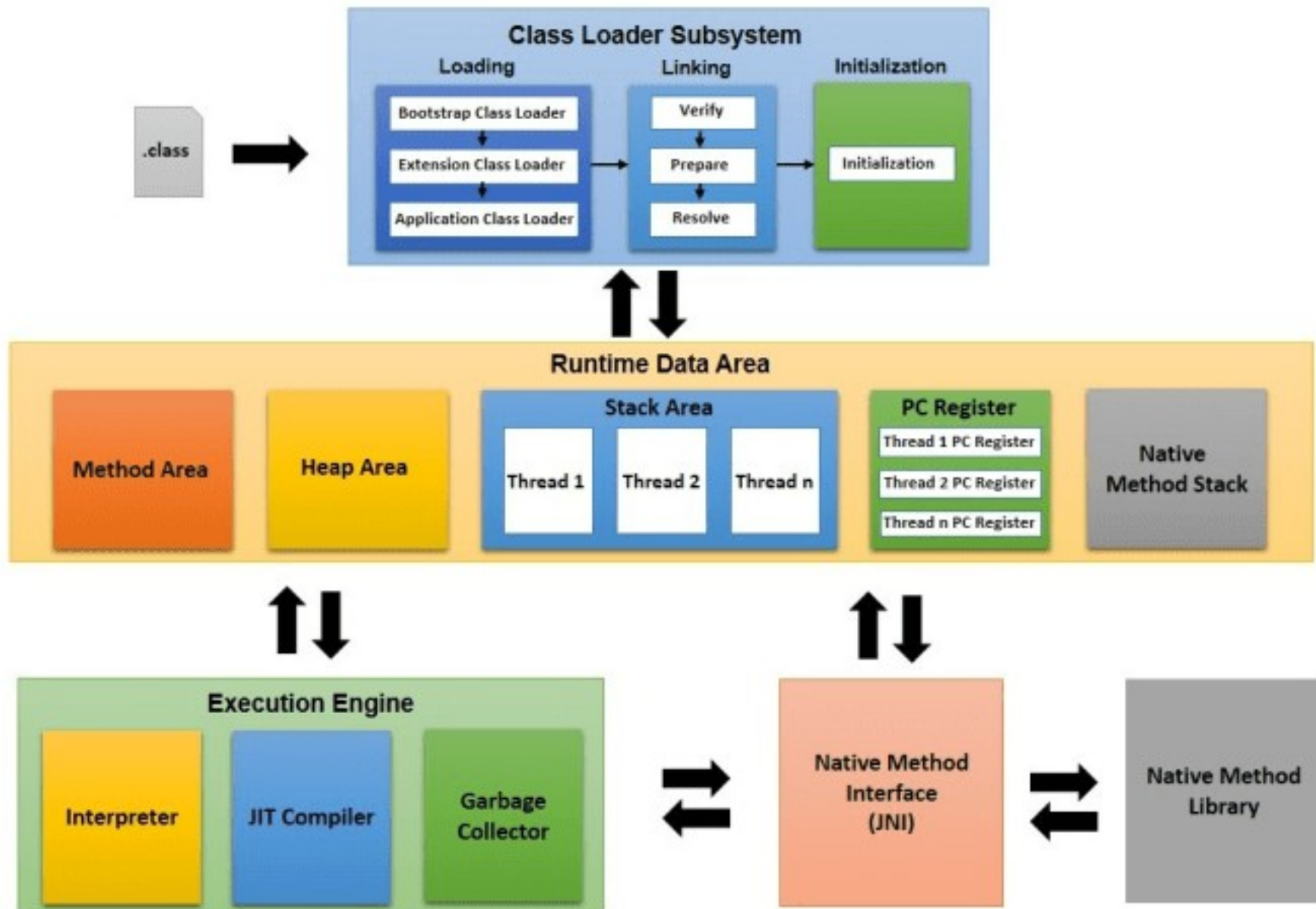


# Why Use Threads

- Generally applications use different resources
  - Threads allow for more effective utilization of resources
  - Allows for more effective use of multiple processors
- Multithreading introduces a layer of complexity
  - How thread are managed can impact performance
  - Thread have their own stack areas but share the heap



# JVM Architecture



# Non-Daemon vs Daemon Threads

- Non-daemon threads
  - Also known as “user” threads or “non-native” threads
  - The main() statement starts a use thread
  - The JVM exits when the last non-daemon thread ends
- Daemon threads
  - Ending a daemon thread does not cause the JVM to exit
  - Manage a lot of the internal JVM tasks
    - *Eg. Garbage collection*

A top-down view of a wooden desk. In the top right, a portion of a silver laptop is visible, showing keys like 'tab', 'Q', 'W', 'E', 'A', 'S', 'Z', 'fn', 'control', and 'option'. Below the laptop, a pair of black-rimmed glasses lies horizontally. To the right of the glasses is a white ceramic cup filled with dark coffee, with a yellow handle. In the top center, a small green succulent plant in a dark pot sits on the desk. The background is a light-colored wooden surface with a prominent grain pattern.

# Module Topics

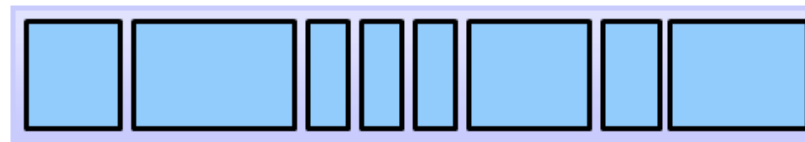
**1. Threads**

**2. Garbage Collection**

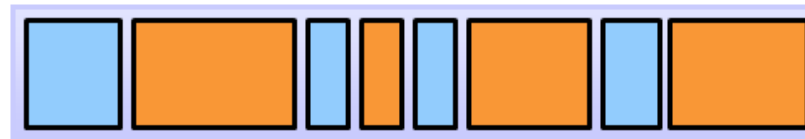


# Step 1: Marking

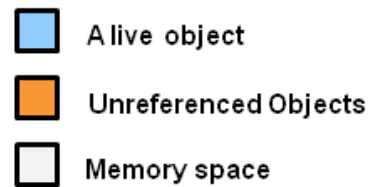
## Marking



Before Marking



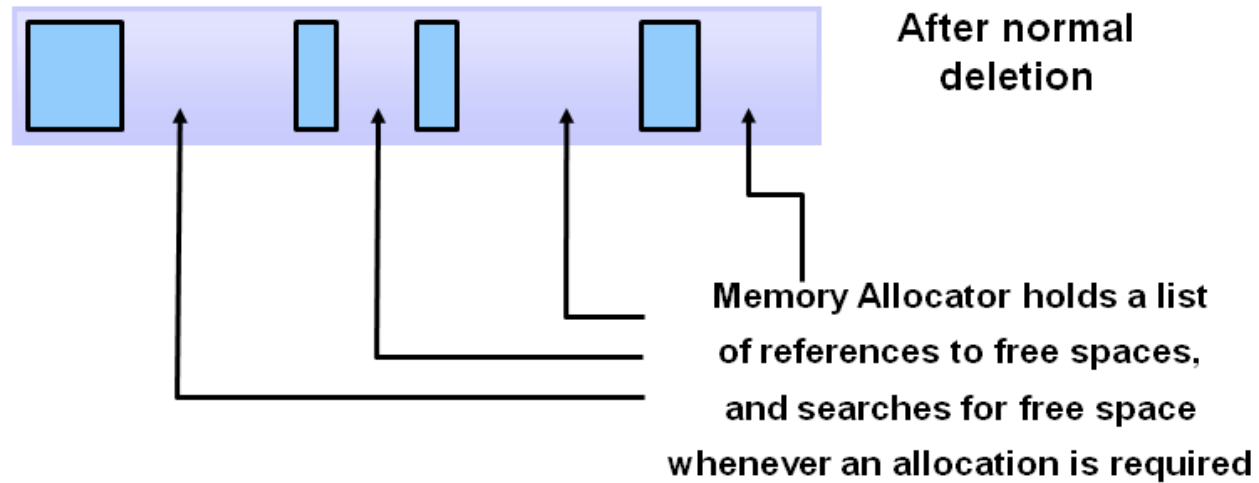
After Marking





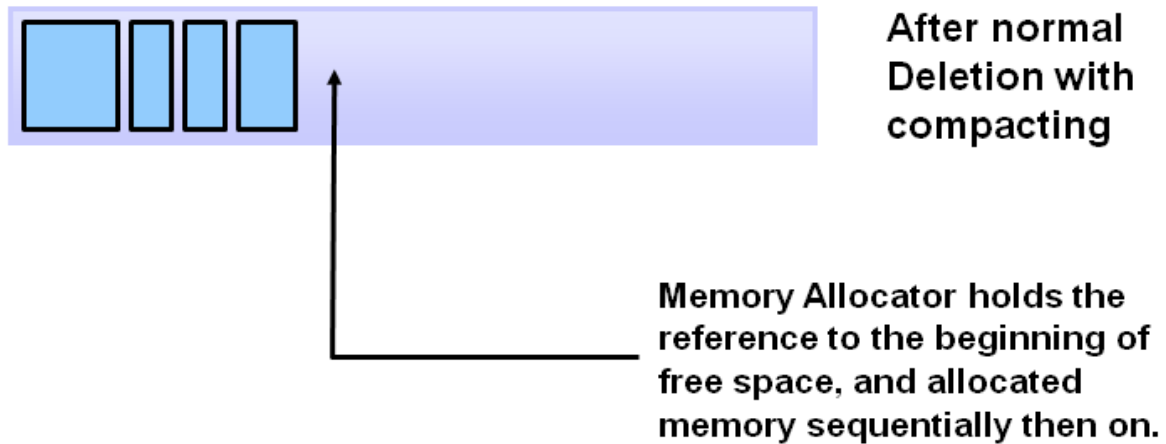
## Step 2: Normal Deletion

### Normal Deletion

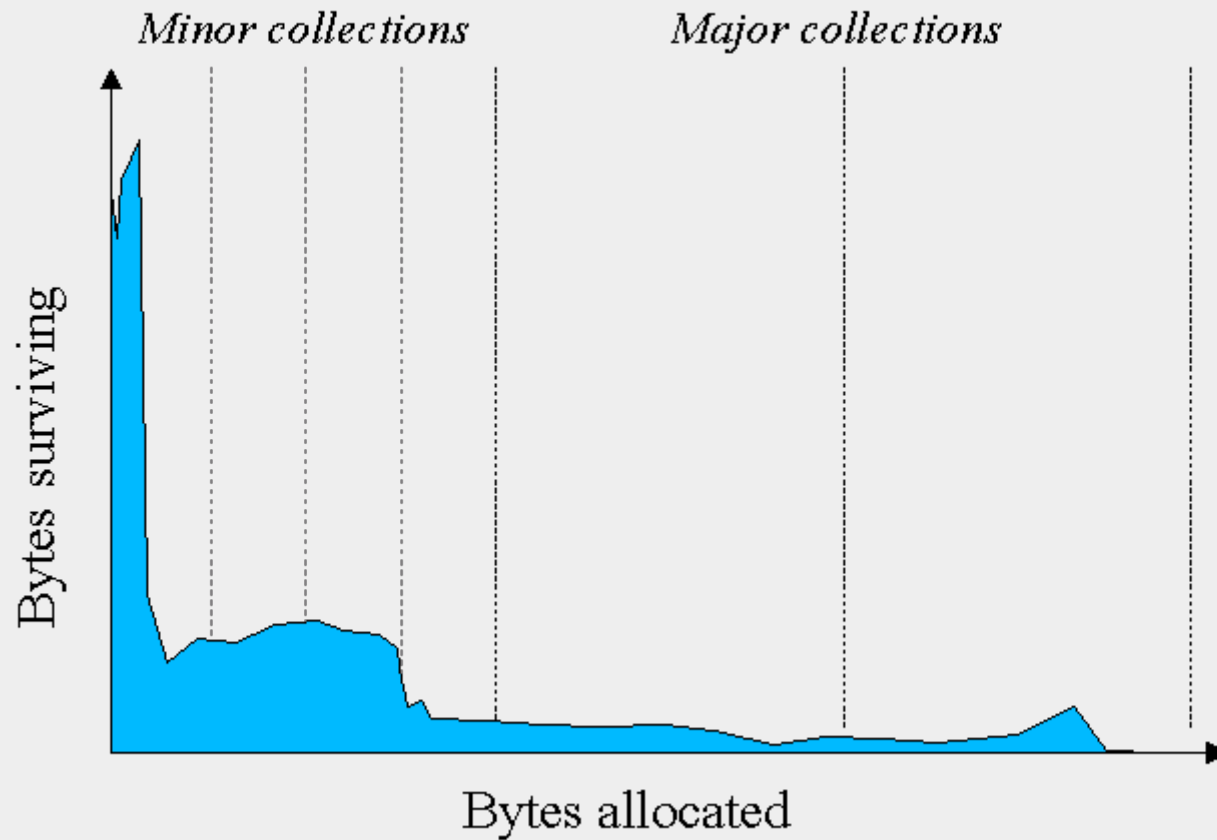


# Step 2a: Deletion with Compacting

## Deletion with Compacting



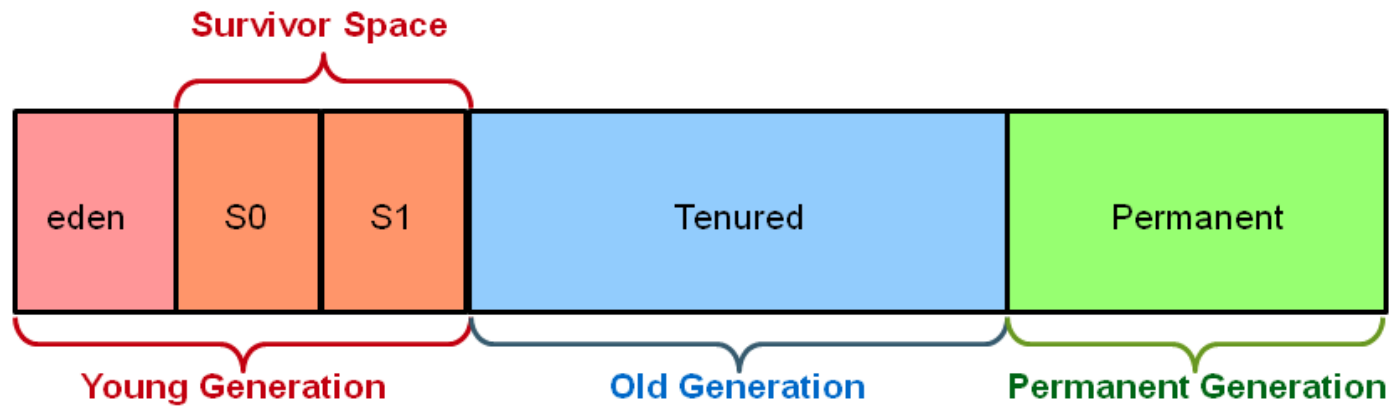
# Why Generational Garbage Collection?



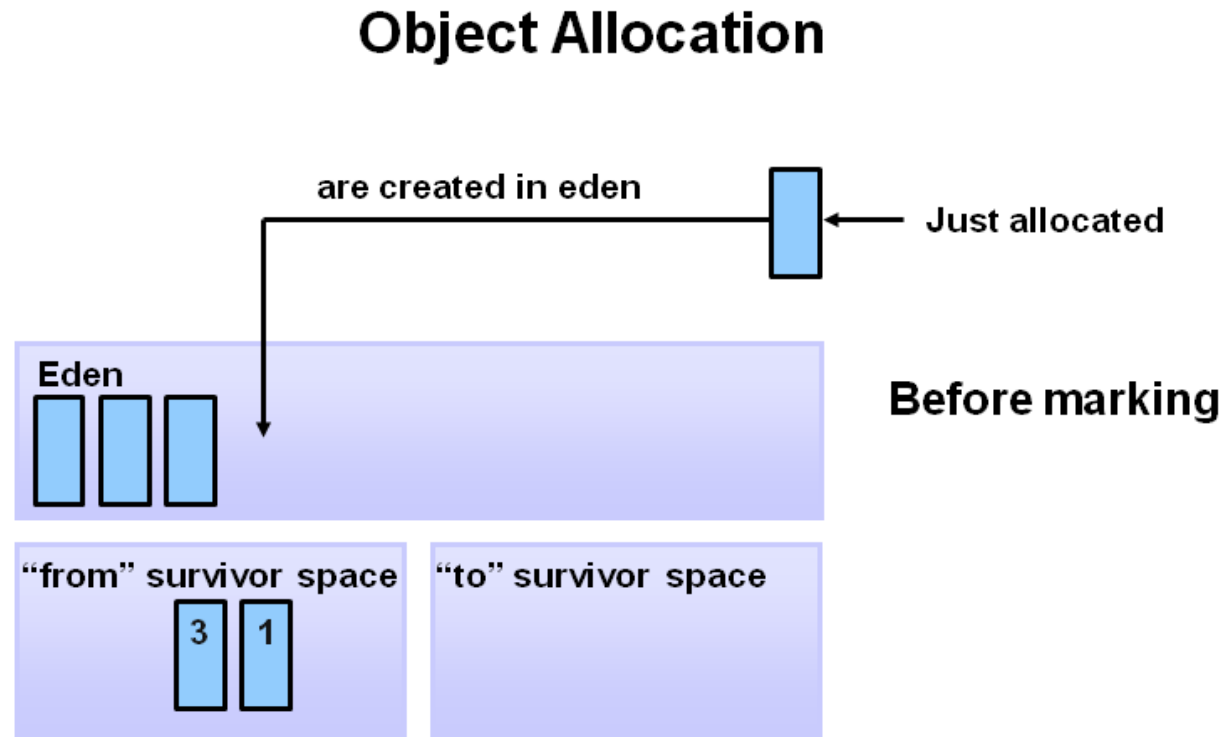


# JVM Generations

## Hotspot Heap Structure

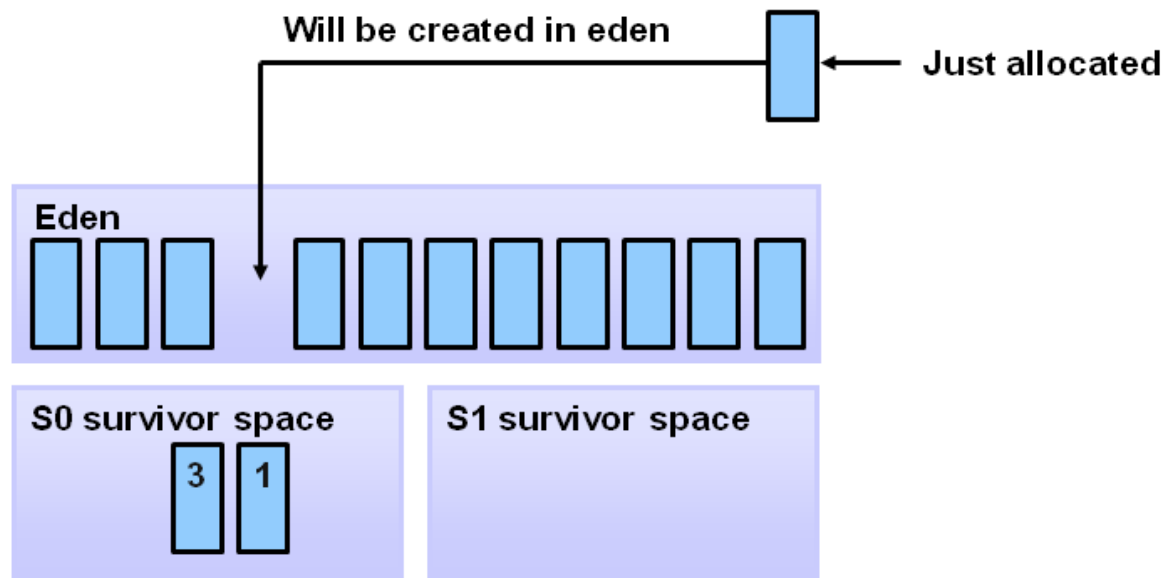


# Object Allocation



# Triggering Minor GC

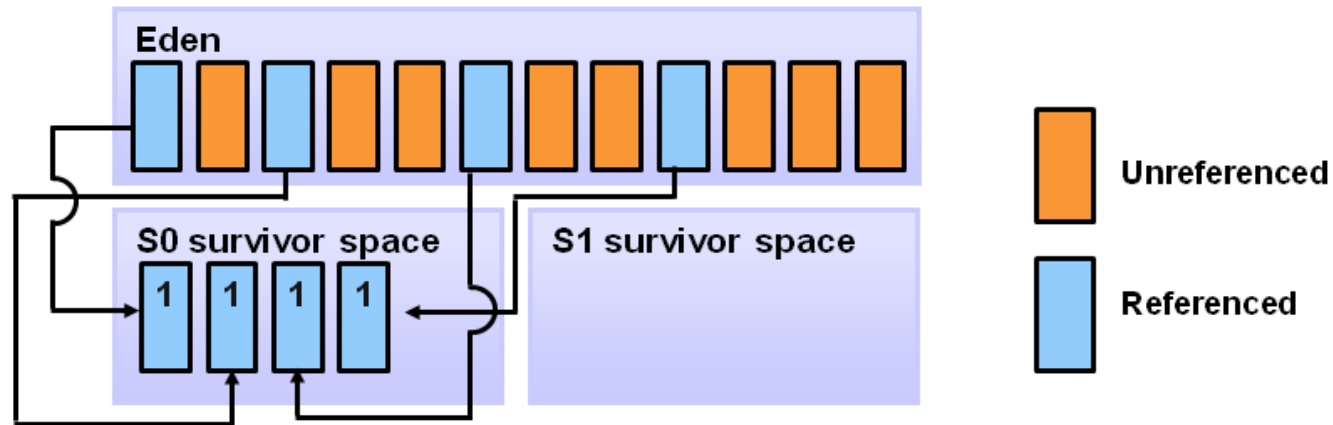
## Filling the Eden Space





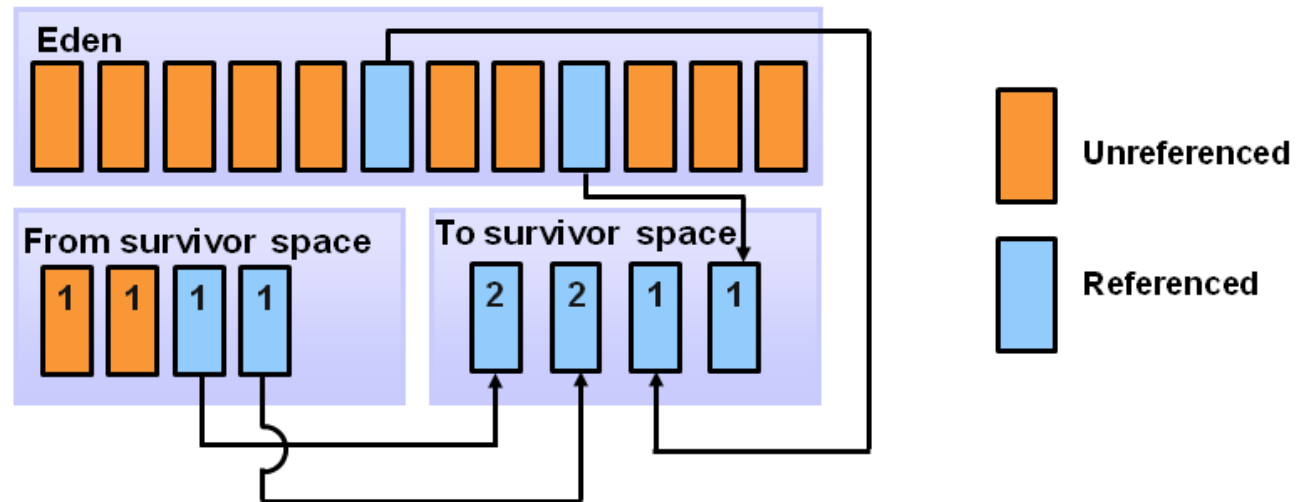
# Executing Minor GC

## Copying Referenced Objects



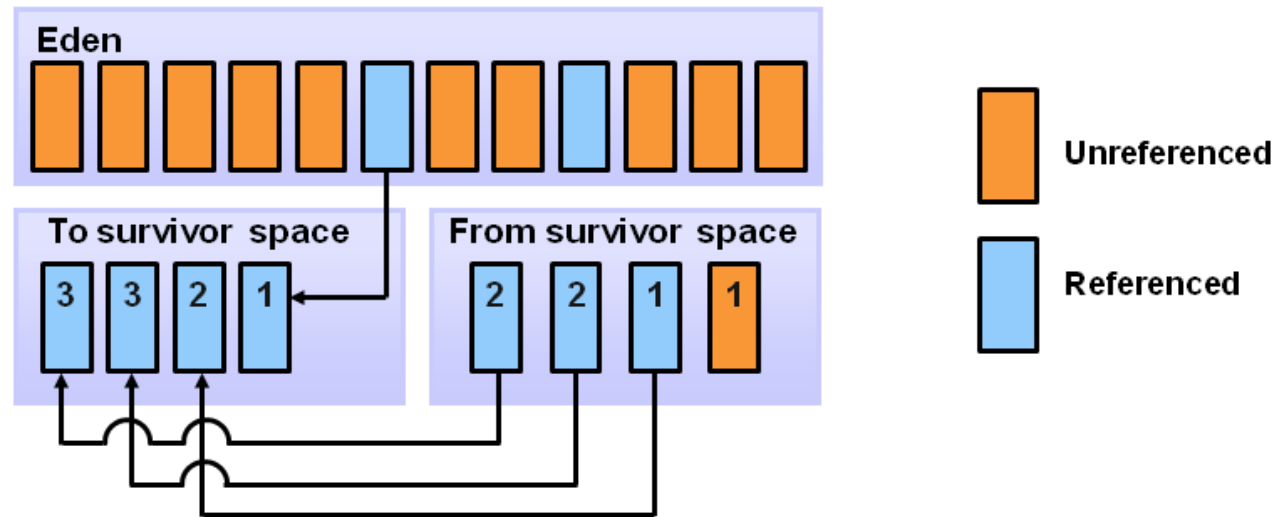
# Next Minor GC

## Object Aging



# Next Minor GC

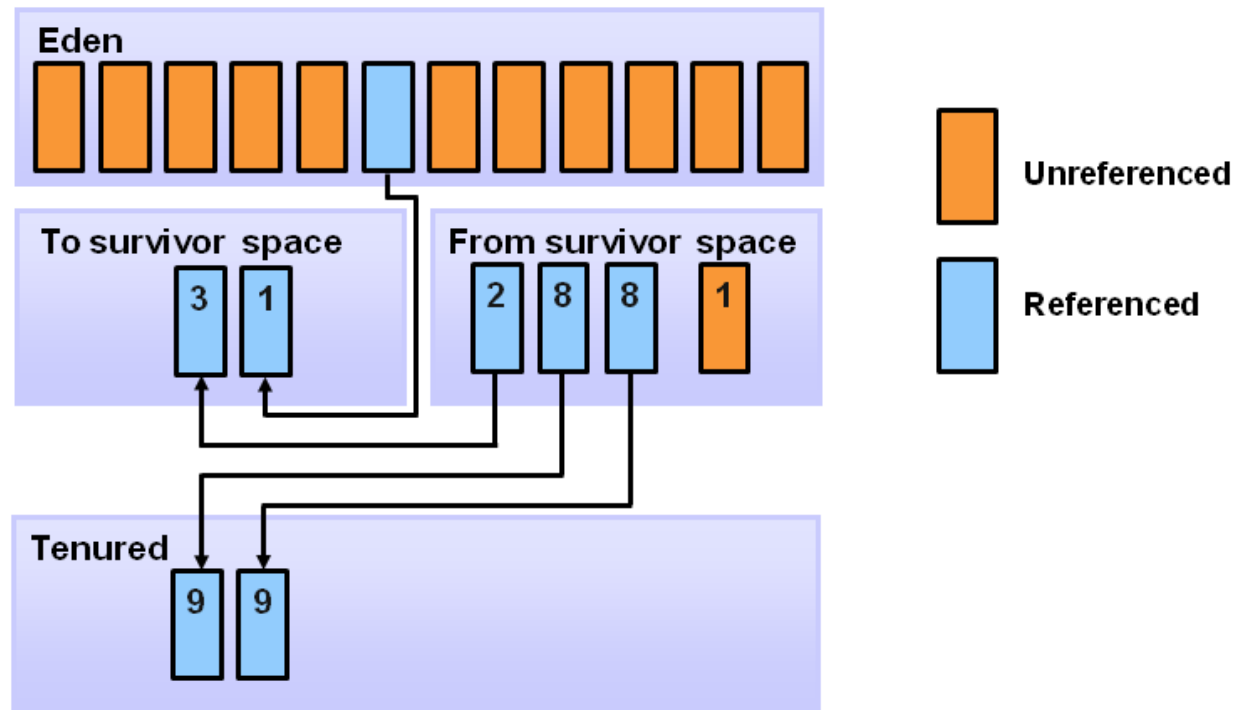
## Additional Aging



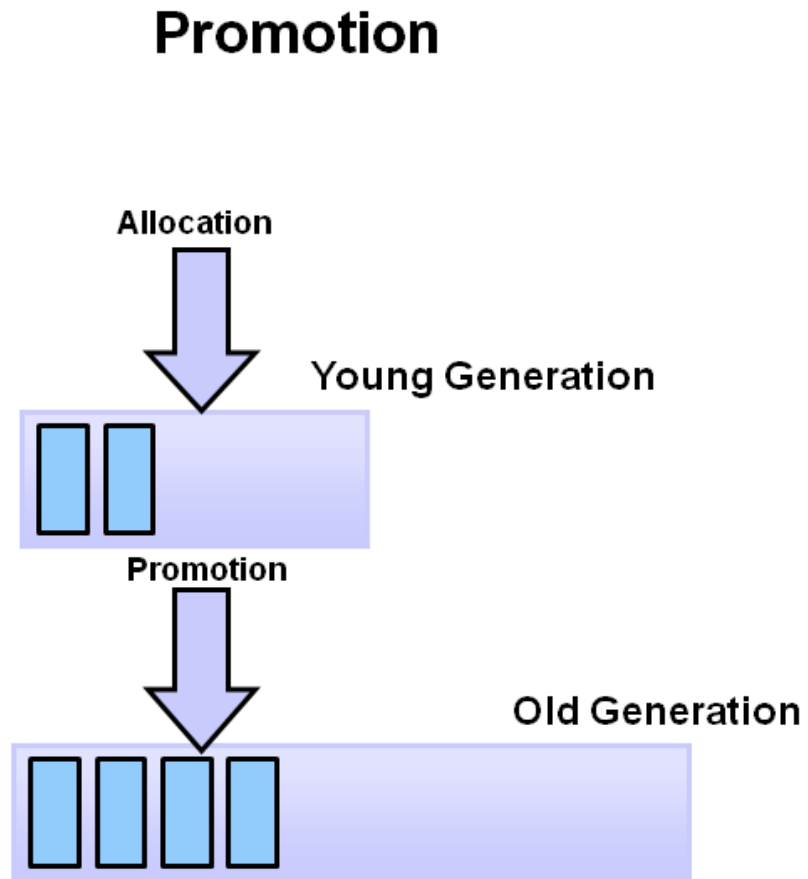


# Move to Tenured Storage

## Promotion

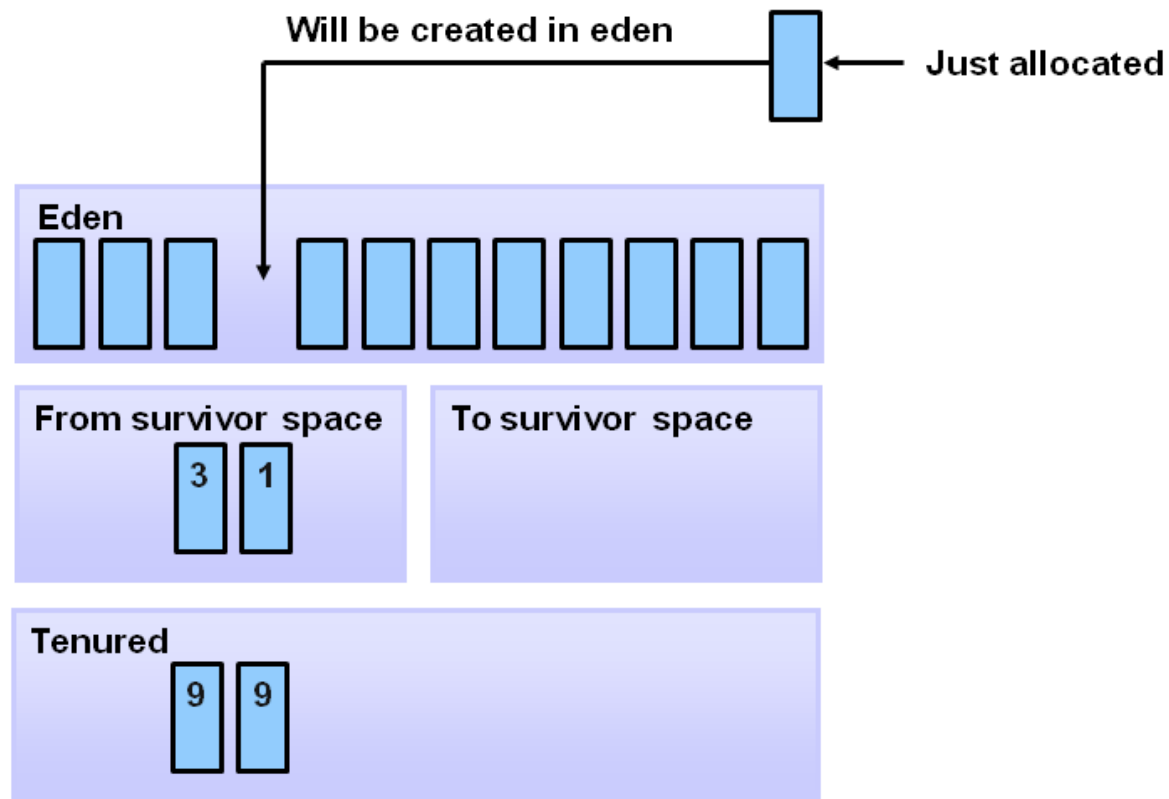



# Continued Promotion



# Major GC

## GC Process Summary



A top-down view of a wooden desk. In the top right corner, a portion of a silver laptop is visible, showing keys like 'ESC', '1', '2', '3', 'Q', 'W', 'E', 'tab', 'caps lock', 'A', 'S', 'Z', 'fn', 'control', and 'option'. Below the laptop, a pair of black-rimmed glasses lies horizontally. To the right of the glasses is a white ceramic cup filled with dark coffee, with a yellow handle. In the bottom right corner, the top edge of a black tablet is visible. At the top center, a small green succulent in a dark pot sits on the desk. A semi-transparent dark grey rectangle is positioned on the left side of the desk, containing the text 'Module End' in white.

**Module End**