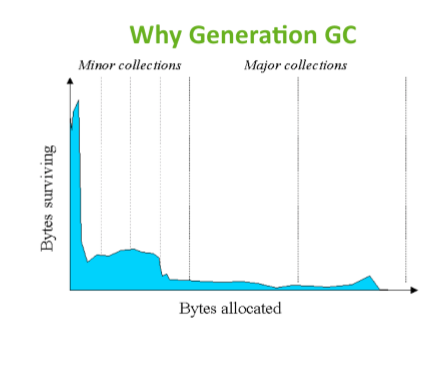
**Garbage Collector**

De **GC** ruimt unreferenced objecten aan.  
Manueel: De waarde van een variabele gelijkstellen aan null.

De GC werkt op de **Stack** (Houdt de referencies bij)  
**JIT Compiler** (Just In Time compiler) -> Zet byte code om in native code (Terwijl het runt)

Perfomance  
- **Responsiveness** (Hoe reageert een applicatie)  
 -> Responsive design  
 -> vB knop drukken -> Direct antwoord en data  
- **Throughput** (Hoeveel data wordt er verwerkt)  
 -> vB database query, 100mb direct en vlot verkrijgen

*Hoe werkt de GC?*1) Marking: Unreferenced objecten aanduiden  
2) Normal deletion: Deze deleten + gaten compacten

  
*Als je het programma opstart, gebruikt het veel CPU, en na een tijdje stabiliseert het zich*

Objecten die ontstaan verschijnen op de “**eden**” sector. (Veel GC)  
Als ze blijven bestaan, gaan ze naar de **Survivor** space (= objecten die overleven) (Weinig GC)  
Ze kunnen nu verhuisd worden tussen **S0** en **S1**.  
Als ze nog langer overleven -> **Tenured** (GC runt hier)  
Als ze nog langer blijven -> **Permanent** (Geen GC)

**Oef:**  
Console -> jvisualvm  
Visualvm opstarten en ga naar de Visual GC tab  
