Taller 1 – Instalar VisualVM

1. Instalar VisualVM (sólo es necesario descomprimirlo) y ejecutarlo por primera vez para que ejecute el proceso de calibración.
2. Instalar el plugin *VisualGC* a través el menú *Tools –> Plugins*. En la pestaña *Available* seleccionar *Visual GC* y presionar *Install*.
3. Reiniciar *VisualVM*.
4. Ejecutar cualquier programa con Java (por ejemplo Eclipse) y verificar que *VisualVM* funciona pasando por todas las pestañas y verificando que en todas muestra las métricas correspondientes.
5. *Tomar una imagen de VisualVM mientras realiza el monitoreo*.
6. ¿Alguna métrica no aparece? ¿Cuál puede ser la razón?

Taller 2 – Parámetros de Memoria

En este pequeño taller ejecutará el programa de ejemplo memtest.jar con diferentes configuraciones de memoria y con VisualVM comprobará que los valores se hayan asignado correctamente. Para cada ejecución tome imágenes del la información entregada por el plugin VisualVM y guárdelas como evidencia del taller.

Guarde imágenes de VisualGC para cada configuración.

Configuración 1:
- Heap mínimo: 200 MB.
- Heap máximo 350 MB.
- Tamaño fijo del Eden: 150 MB.
- Memoria permanente mínima: 64 MB.
- Memoria permanente máxima: 128 MB.

Configuración 2:
- Heap mínimo: 0.4 GB.
- Heap máximo 0.7 GB.
- Tamaño mínimo del Eden: 128 MB.
- Tamaño máximo del Eden: 256 MB.
- Memoria permanente fija: 0.25 GB.

Configuración 3:
- Heap fijo: 1 GB.
- Eden fijo: 45% del Heap.
- Memoria permanente fija: 128 MB.

Configuración 4:
- Heap fijo: 3 GB.
- Eden fijo: 45% del Heap.
- Memoria permanente fija: 512 MB.

Taller 3 – Recolección de Basura

En este taller ejecutará la aplicación de ejemplo y analizará los mecanismos de recolección de basura para optimizarlos posteriormente.

Ejecución 1:

1. Ejecute el programa `memtest.jar` con los siguientes parámetros:
    - Heap fijo de 512 MB.
    - Eden fijo de 64 MB.
    - Memoria permanente de 64 MB.
2. Abra VisualVM para monitorear el proceso y abra la pestaña de VisualGC.
3. Ejecutar una secuencia donde cree 4 objetos y elimine 1 (1, 1, 1, 1, 2) hasta que la JVM se caiga con un *OutOfMemoryError*.
4. Responda las siguientes preguntas:
    - ¿Cuántos objetos quedaron vivos al final?
    - ¿Cuantas recolecciones menores ejecutó?
    - ¿Cuántas recolecciones mayores?
    - ¿Cuánto tiempo tomó la recolección en cada sección de memoria?
5. *Adjunte un pantallazo de Visual GC en las respuestas del taller*.

Ejecución 2:

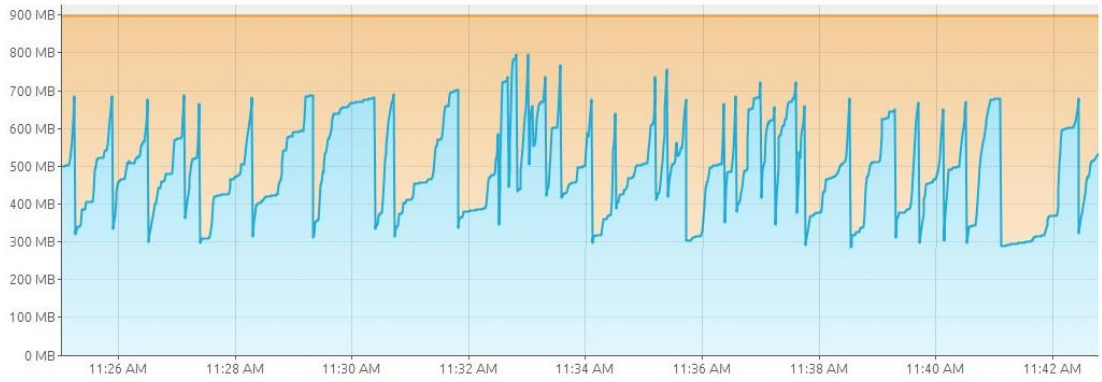
1. Ejecute el programa `memtest.jar` con los siguientes parámetros:
    - Heap fijo de 512 MB.
    - Eden fijo de 400 MB.
    - Memoria permanente de 64 MB.
2. Abra VisualVM para monitorear el proceso y abra la pestaña de VisualGC.
3. Ejecutar una secuencia donde cree 4 objetos y elimine 1 (1, 1, 1, 1, 2) hasta que la JVM se caiga con un *OutOfMemoryError*.
4. Responda las siguientes preguntas:
    - ¿Cuántos objetos quedaron vivos al final?
    - ¿Cuantas recolecciones menores ejecutó?
    - ¿Cuántas recolecciones mayores?
    - ¿Cuánto tiempo tomó la recolección en cada sección de memoria?
5. *Adjunte un pantallazo de Visual GC en las respuestas del taller.*

¿Cuál cree que debe ser la configuración óptima para que permita la creación de un gran número de objetos con recolecciones de basura que no impacten mucho el rendimiento?
Intente configuraciones y haga pruebas.

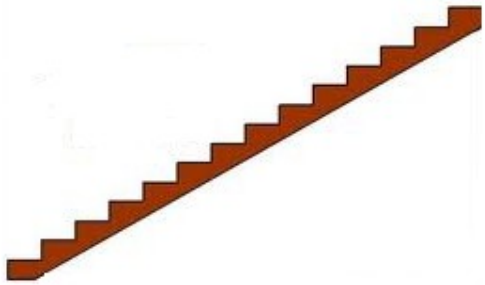
Taller 4 – Recolección de Basura

Usando el programa de ejemplo creará secuencias que simulen el comportamiento de una aplicación bajo 3 escenarios, al final adjunte pantallazos con los resultados obtenidos e indique cuál fue la secuencia de creación y eliminación de objetos utilizada.

1. Cree una secuencia que muestre un consumo de memoria regular, similar a la siguiente imagen.



2. Cree una secuencia que muestre un consumo en forma de escala o muy similar.



3. Cree una secuencia que simule un memory leak.