

**UNIVERSIDAD TECNOLÓGICA DEL PERÚ**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Sección: 40579**

**Tema:**

S2.s1-Tarea

**INTEGRANTES:**

Miguel Angel Velasquez Ysuiza U23231519

Roberto Agustín Mejía Collazos U23254461

Manuel Ángel Pecho Santos U23201694

Ivan Sulca Palacios U23220872

Josué Carmelo Murga Guimaray U23238660

**DOCENTE:**

EVELYN MARILYN RIVEROS HUAMAN

01 de abril del 2024



**¿Cómo funciona el Garbage Collector en Java?**

**y explicar con ejemplos.**

El [***Garbage Collector***](http://coco://sendMessage?ext=%7B%22s%24wiki_link%22%3A%22https%3A%2F%2Fes.wikipedia.org%2Fwiki%2FRecolector_de_basura%22%7D&msg=Garbage+Collector)(recolector de basura) en Java es un componente del entorno de ejecución de Java ([JRE](http://coco://sendMessage?ext=%7B%22s%24wiki_link%22%3A%22https%3A%2F%2Fes.wikipedia.org%2Fwiki%2FJava_Runtime_Environment%22%7D&msg=JRE)) que se encarga de gestionar la memoria utilizada por los objetos creados en un programa Java. Su función principal es liberar la memoria ocupada por los objetos que ya no son accesibles y que no se utilizan más, para que esa memoria pueda ser reutilizada por otros objetos.

El Garbage Collector funciona de manera automática y se ejecuta en segundo plano mientras el programa está en ejecución. Utiliza un algoritmo de recolección de basura llamado "Mark and Sweep" (marcar y barrer) para determinar qué objetos son alcanzables y cuáles no.

El proceso de recolección de basura consta de los siguientes pasos:

Marcar: El ***Garbage Collector*** comienza marcando todos los objetos que son accesibles desde el punto de entrada del programa (como los objetos referenciados por variables locales o estáticas). Utiliza un algoritmo de recorrido de grafo para determinar qué objetos están conectados y son alcanzables.

Barrer: Una vez que se han marcado los objetos accesibles, el ***Garbage Collector*** recorre toda la memoria y "barre" los objetos no marcados, es decir, los que no son alcanzables. Estos objetos se consideran basura y se liberan de la memoria.

Compactar: En algunos casos, después de la recolección de basura, puede haber fragmentación de la memoria, es decir, espacios vacíos dispersos entre los objetos. Para optimizar el uso de la memoria, el Garbage Collector puede realizar una compactación, moviendo los objetos vivos juntos y liberando un bloque continuo de memoria.

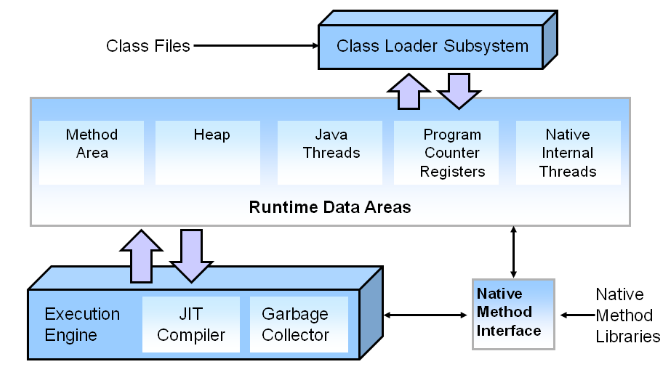
Aquí tienes un ejemplo para ilustrar cómo funciona el Garbage Collector en Java:

public class GarbageCollectorExample {  
 public static void main(String[] args) {  
 // Crear un objeto  
 MyClass obj1 = new MyClass();  
  
 // Crear una referencia al objeto  
 MyClass obj2 = obj1;  
  
 // Liberar la referencia al objeto original  
 obj1 = null;  
  
 // En este punto, el objeto original no es alcanzable  
 // y se considera basura  
  
 // Forzar la ejecución del Garbage Collector  
 System.gc();  
  
 // El Garbage Collector liberará la memoria ocupada por el objeto original  
 }  
}  
  
class MyClass {  
 // Algunos campos y métodos de ejemplo  
}

En este ejemplo, creamos un objeto obj1 de la clase MyClass y luego creamos una referencia obj2 que apunta al mismo objeto. Luego, establecemos obj1 en null, lo que significa que ya no hay referencias al objeto original. Al llamar a System.gc(), le estamos indicando al Garbage Collector que realice una recolección de basura. En este momento, el Garbage Collector detectará que el objeto original no es alcanzable y lo liberará de la memoria.

Es importante destacar que, aunque podemos sugerir al Garbage Collector que realice una recolección de basura llamando a System.gc(), no podemos garantizar cuándo o si realmente se ejecutará. El Garbage Collector de Java es altamente eficiente y se ejecuta de manera automática cuando considera que es necesario liberar memoria.

**Garbage Collector en Java**



**Referencias:**

*https://performancestack.in/garbage-collection-in-java/*