CURSO DESAROLLO WEB CON MEAN

WEB FULL STACK DEVELOPER

Germán Caballero Rodríguez
germanux@gmail.com

Garbage Collector

Recolector de basura

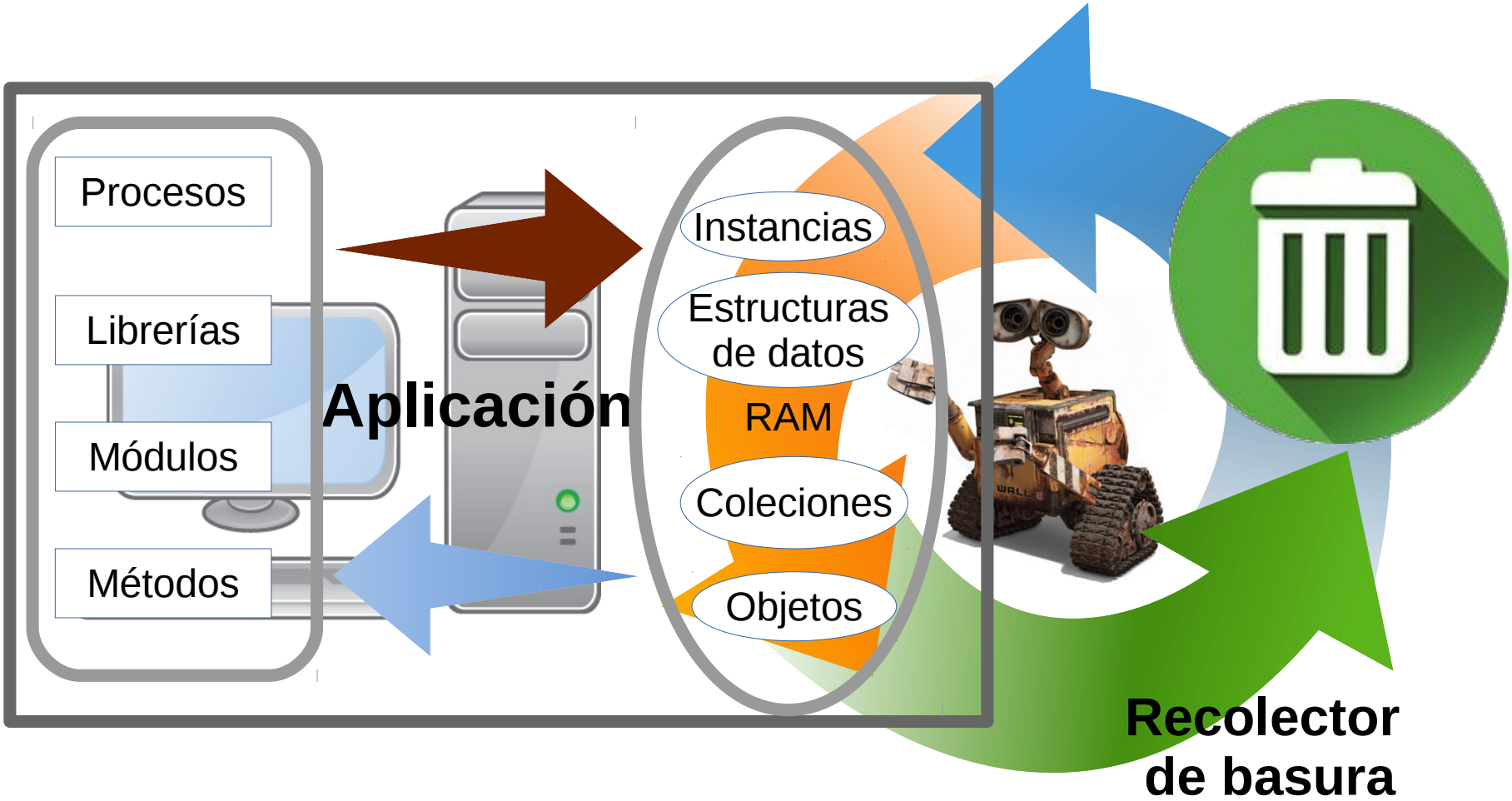
¿Qué es el recolector de basura?

Es un mecanismo implícito
de gestión de memoria
implementado en algunos
lenguajes de programación
de tipo interpretado o semiinterpretado.

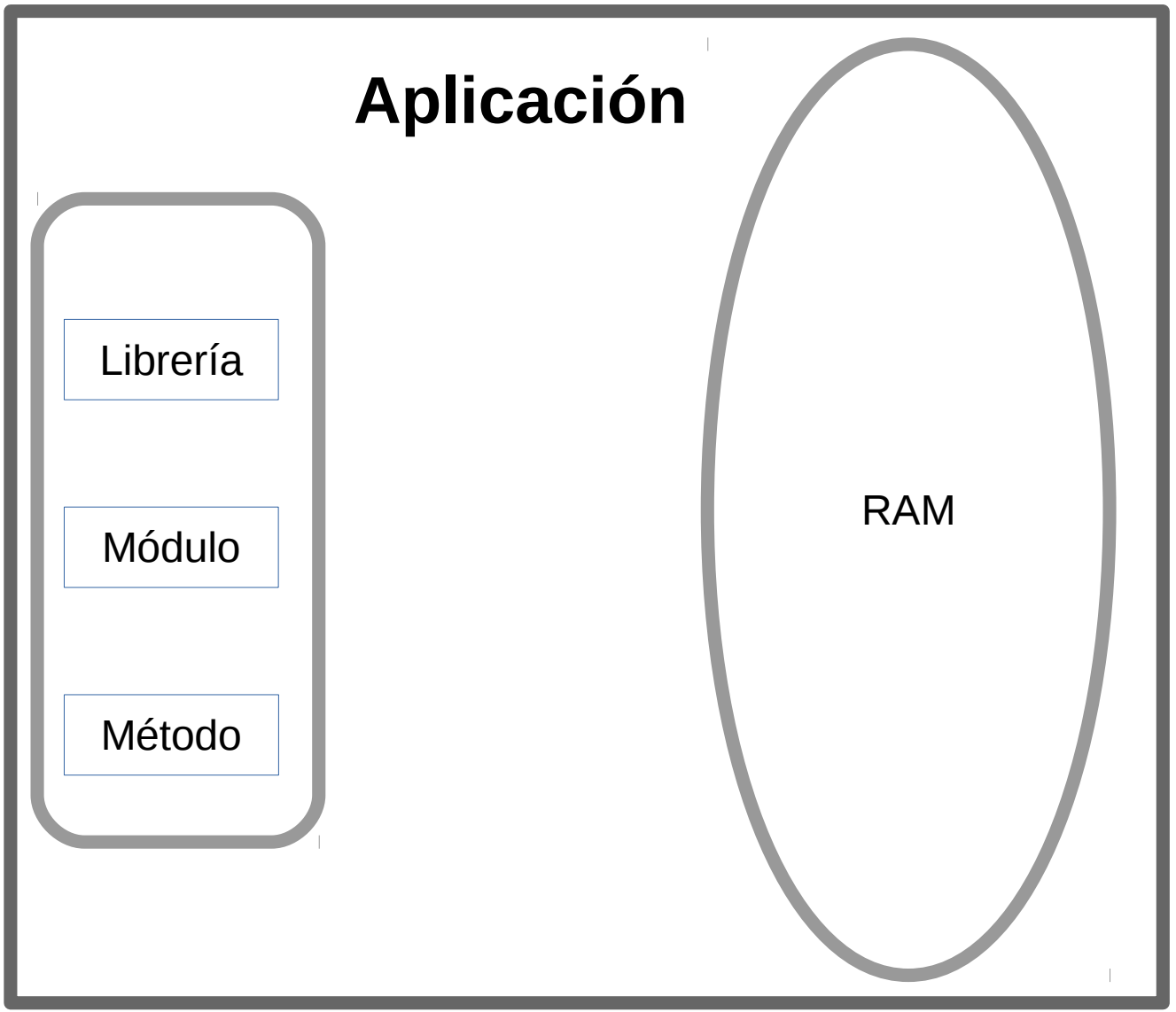
¿De donde viene?
¿A donde a llegado?

- El concepto de recolección de basura fue inventado por John McCarthy en 1958 para evitar la gestión manual de memoria en el lenguaje Lisp.
- Ejemplos de lenguajes con recolector de basura:
  - ALGOL 68
  - C#
  - Caml
  - Clean
  - D
  - Eiffel
  - Go
  - Haskell
  - Java
  - Prolog
  - Lua
  - Mercury
  - ML
  - Modula-3
  - Oberon
  - Oz
  - Objective C 2.0
  - **Pauscal**
  - Perl
  - PHP
  - Ruby
  - Smalltalk
  - SNOBOL
  - SuperCollider
  - Visual Basic .NET
  - Python
  - JavaScript
  - Lisp

¿Para qué sirve?

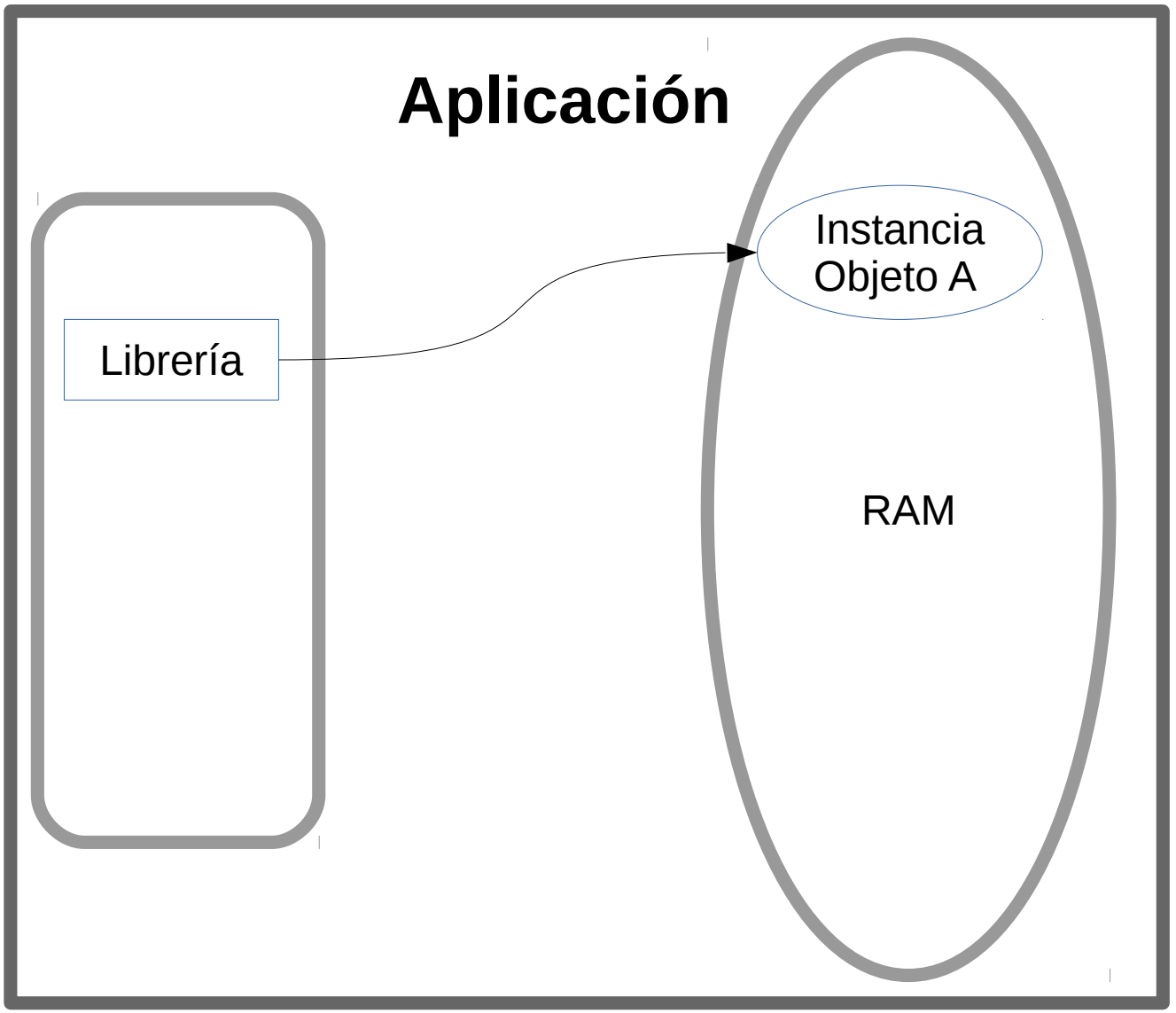


¿Como funciona?



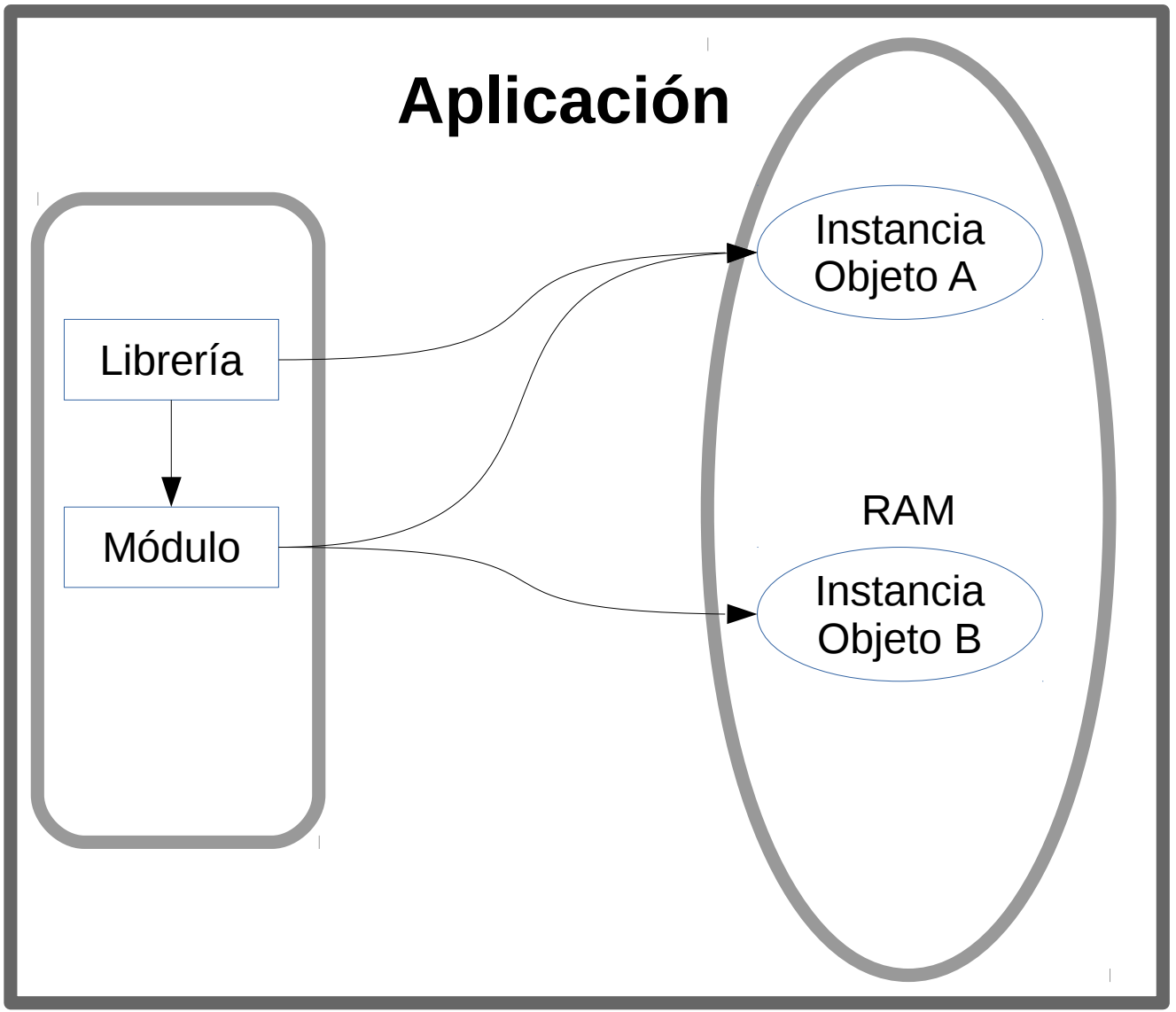
Memoria libre

¿Como funciona?



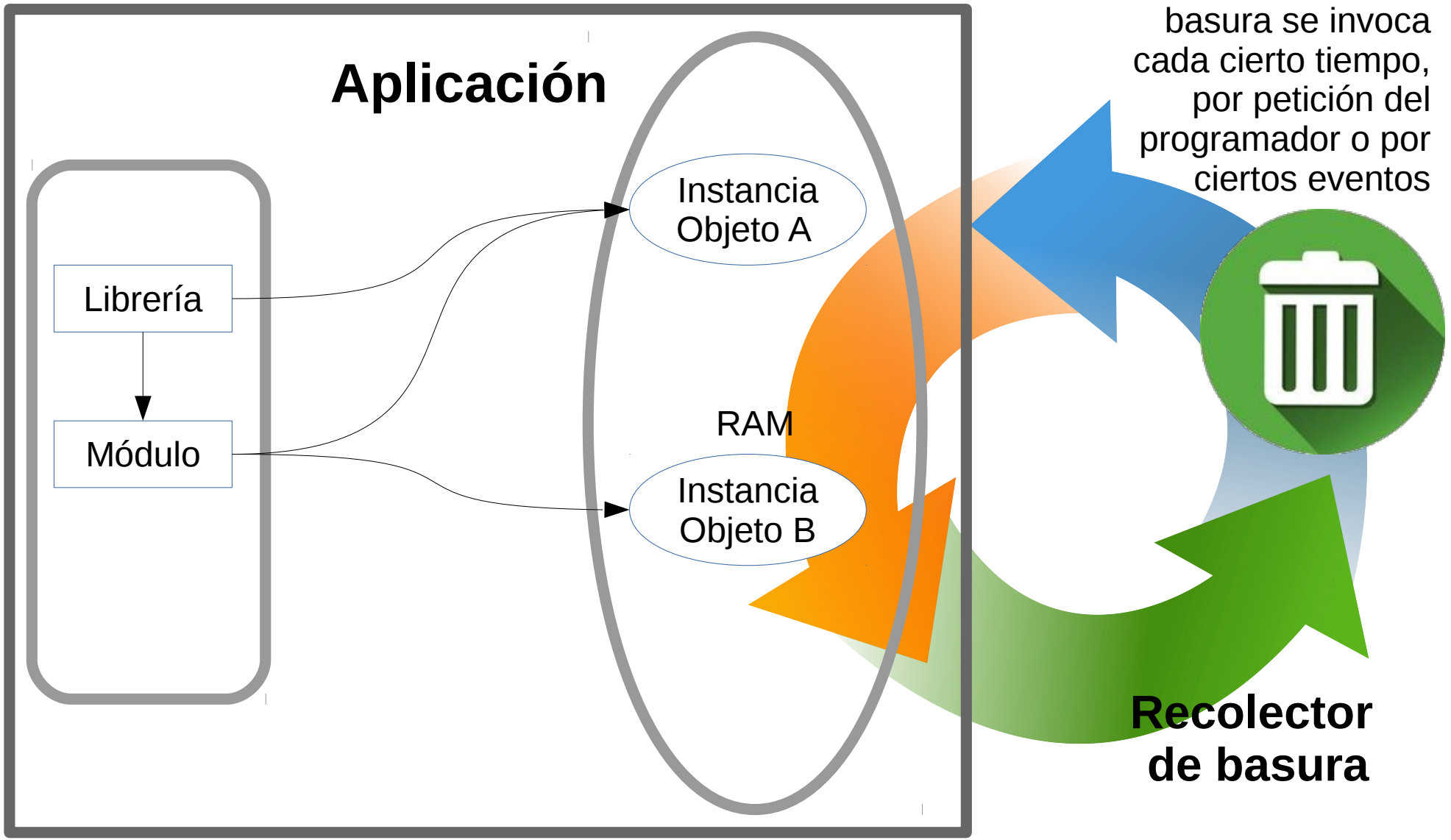
Cualquier instancia de un objeto, colección o dato ocupa memoria

¿Como funciona?

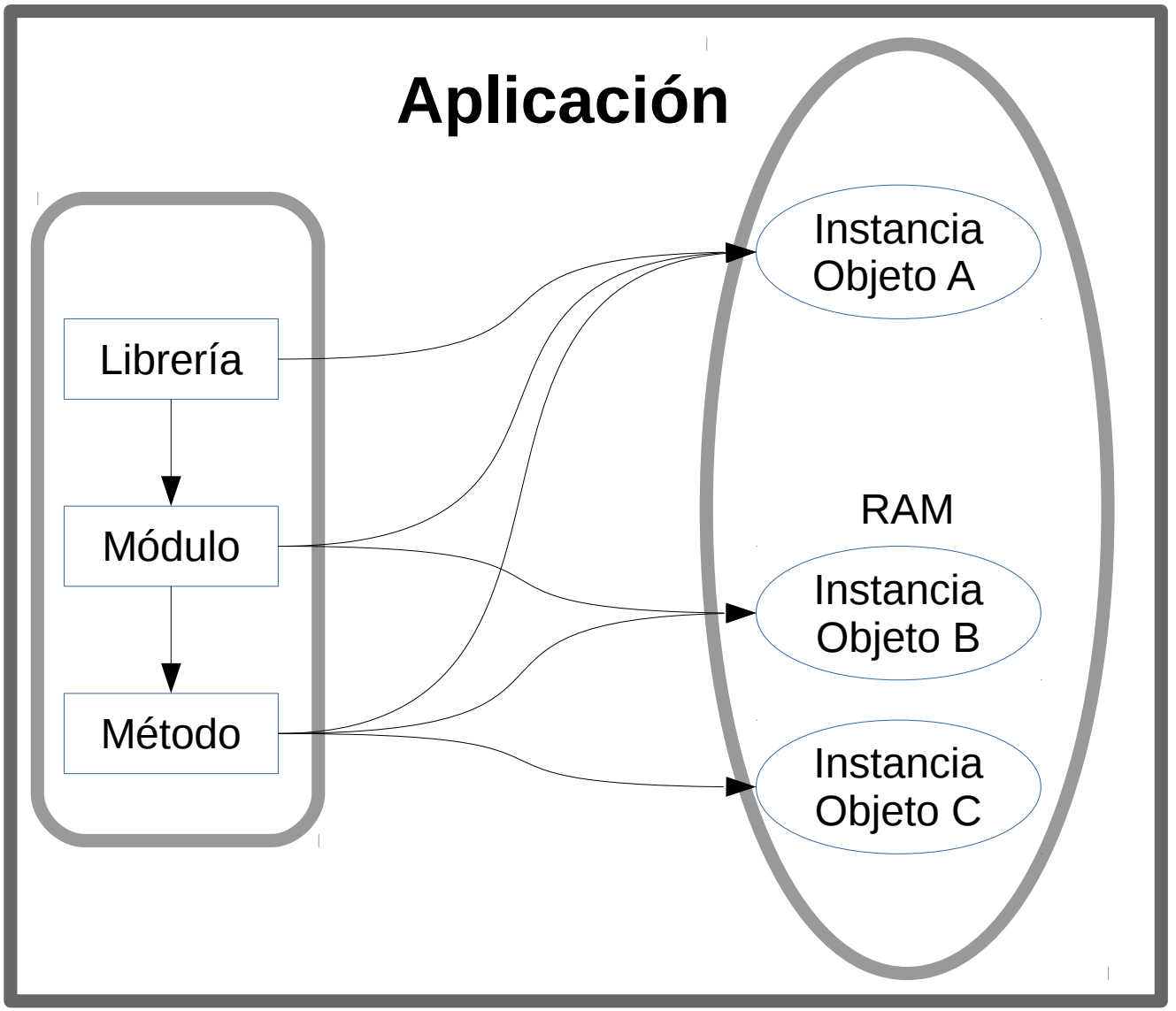


Por cada proceso u objeto que use una instancia de otro objeto, se crea una referencia (puntuero en C)

¿Como funciona?



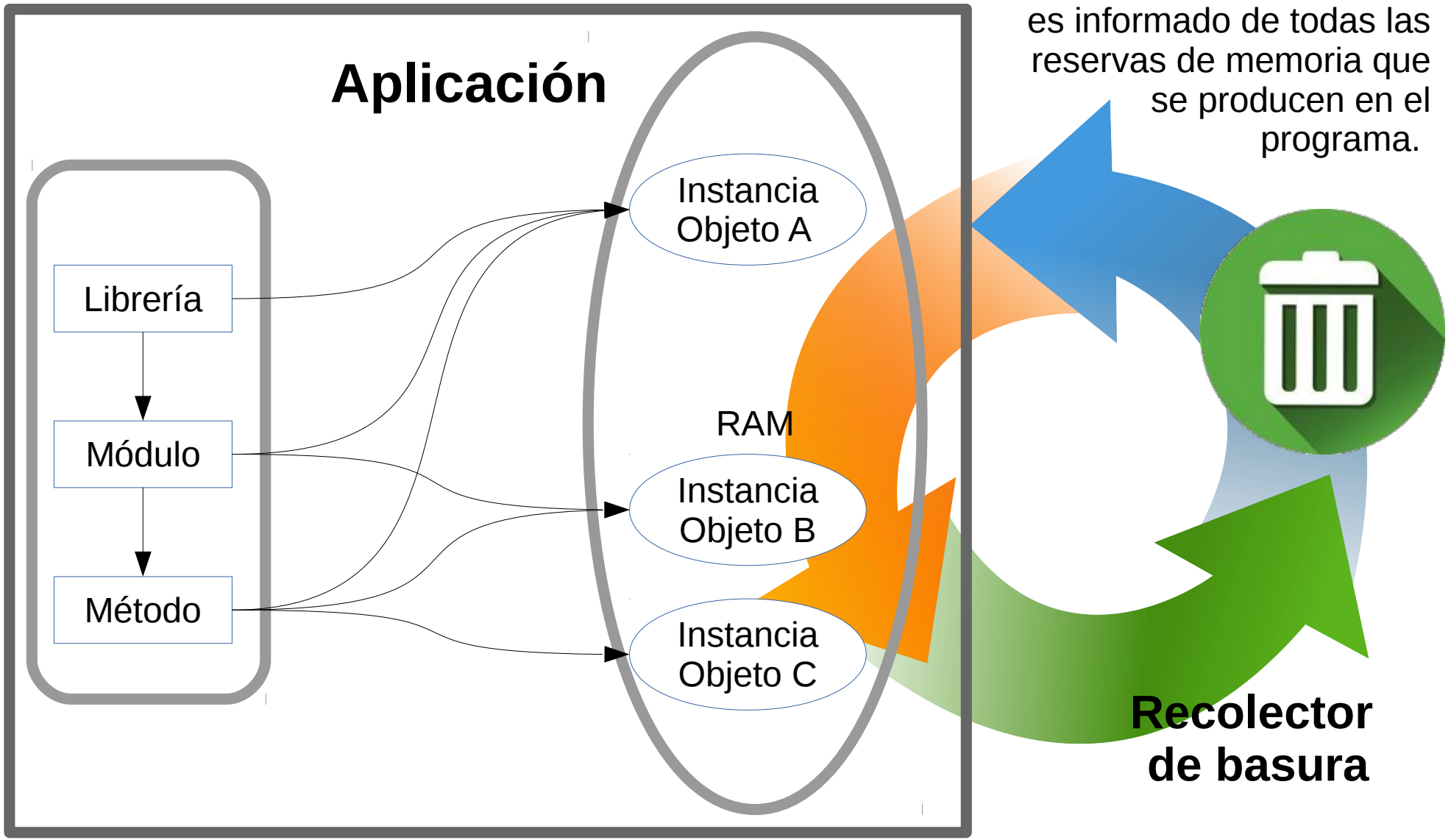
¿Como funciona?



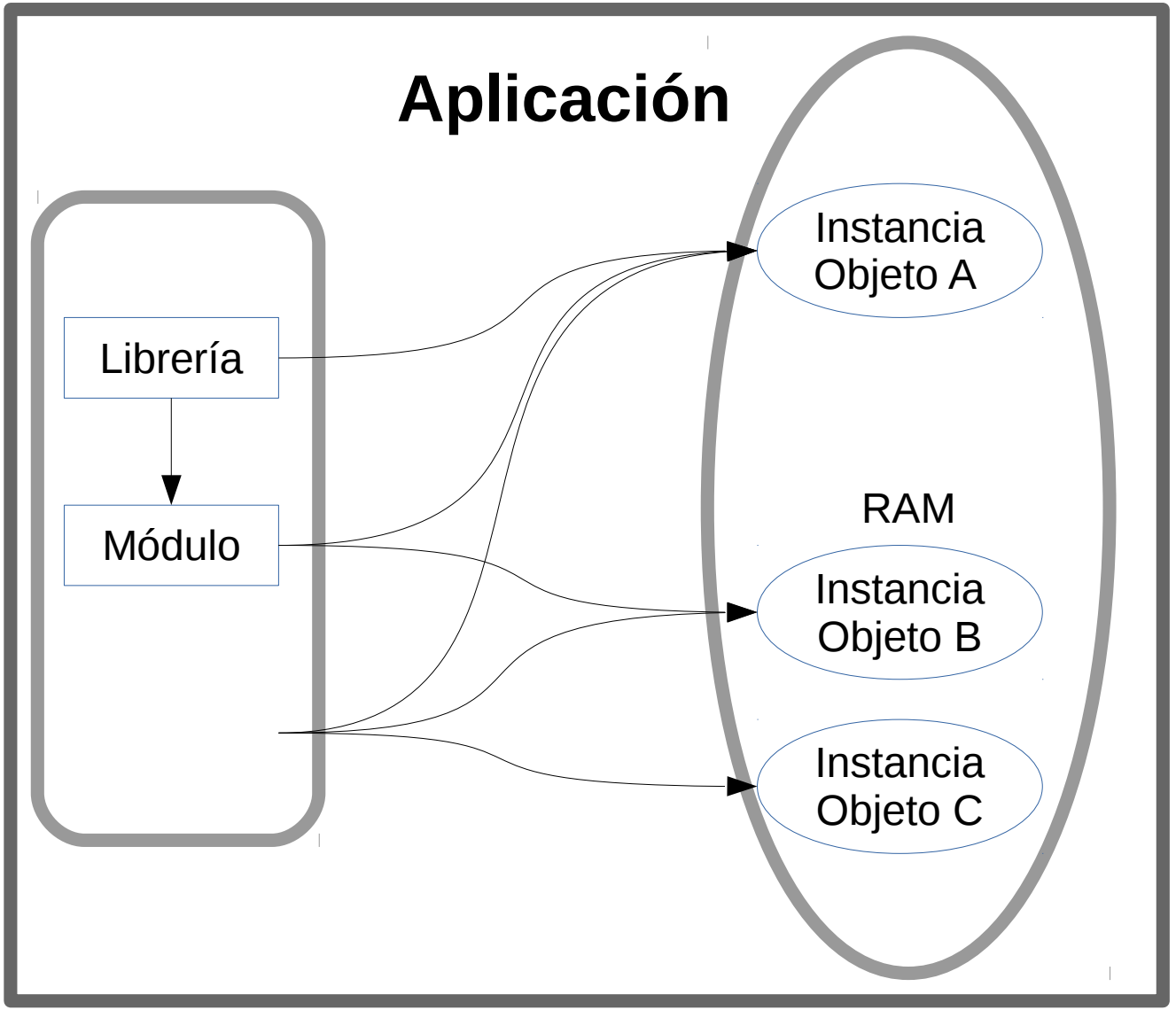
En una aplicación hay miles o millones de referencias que deben ser correctamente gestionadas

¿Como funciona?

El recolector de basura es informado de todas las reservas de memoria que se producen en el programa.

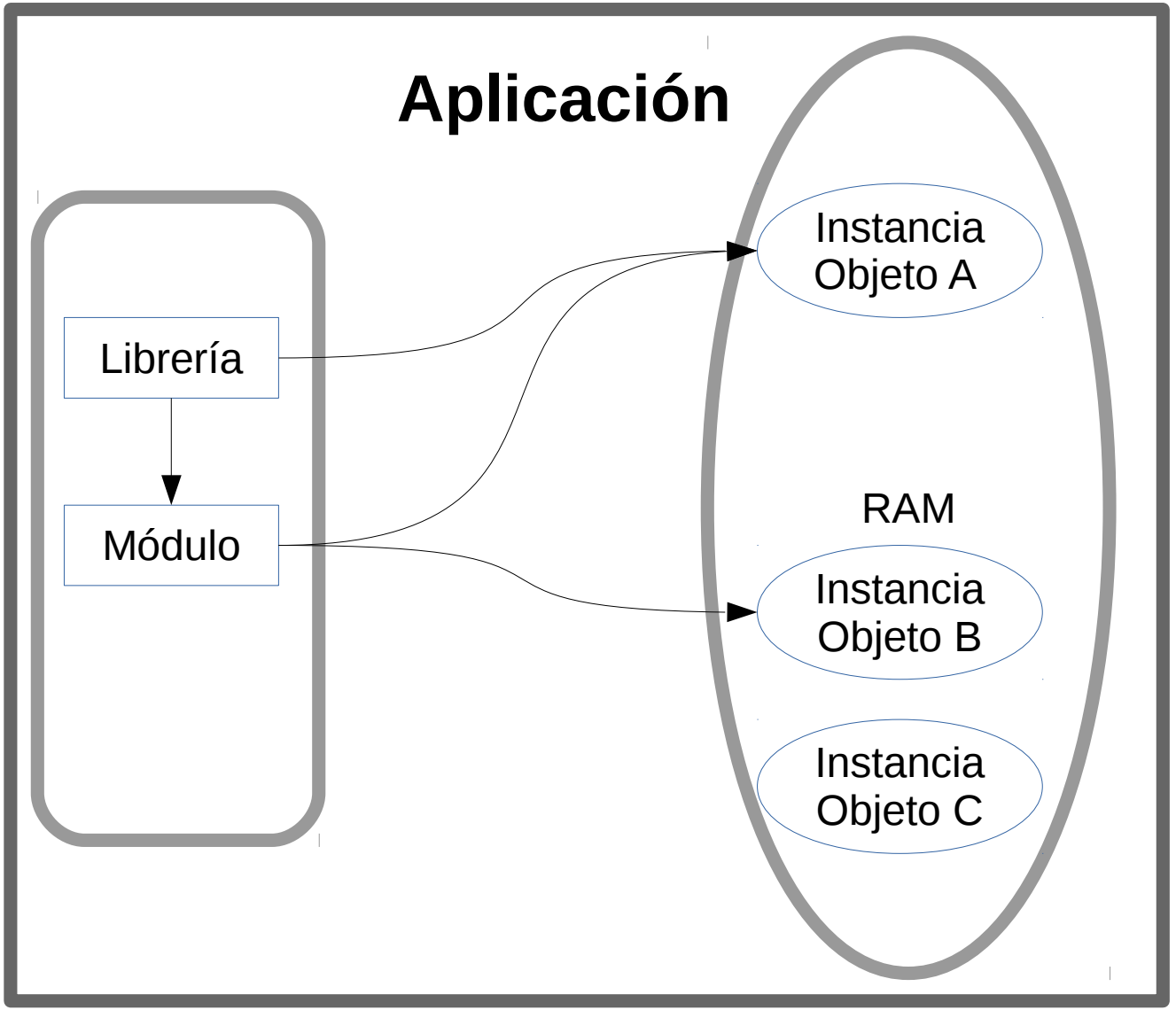


¿Como funciona?



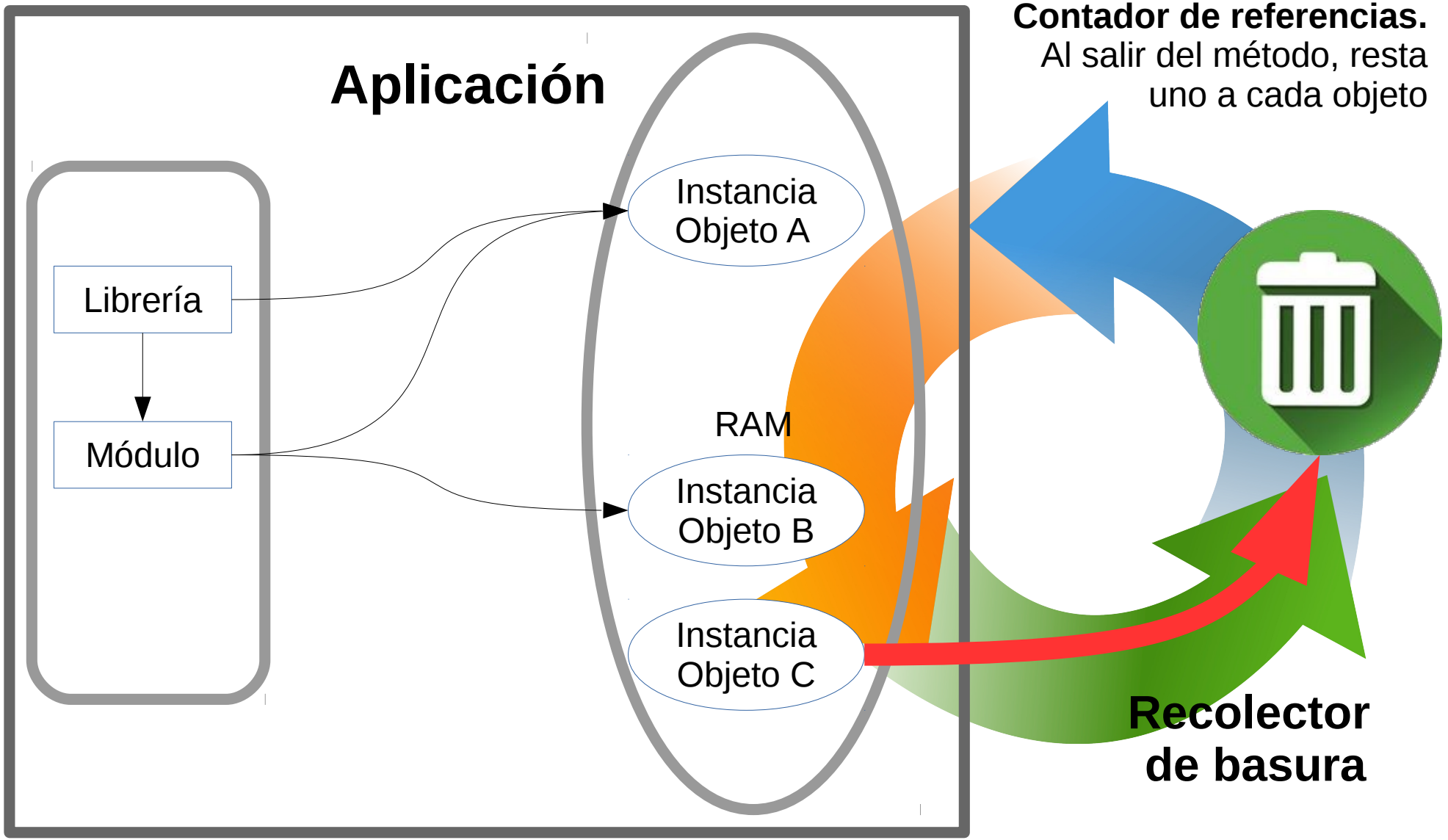
El compilador colabora para que sea posible llevar una cuenta de todas las referencias que existen a un determinado espacio de memoria reservado.

¿Como funciona?

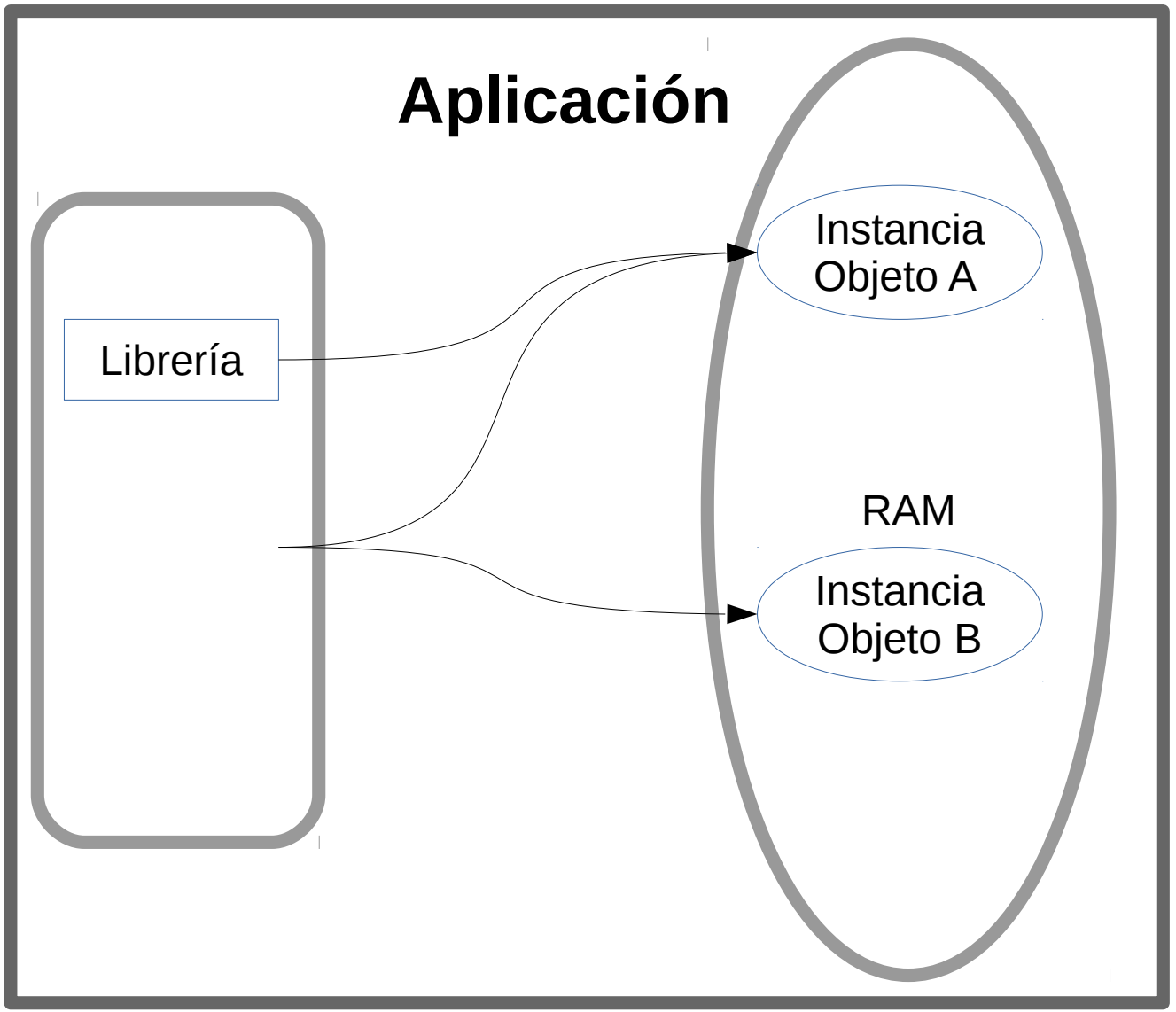


Las referencias se eliminan, pero el objeto no (a menos que el programador lo solicite al compilador)

¿Como funciona?

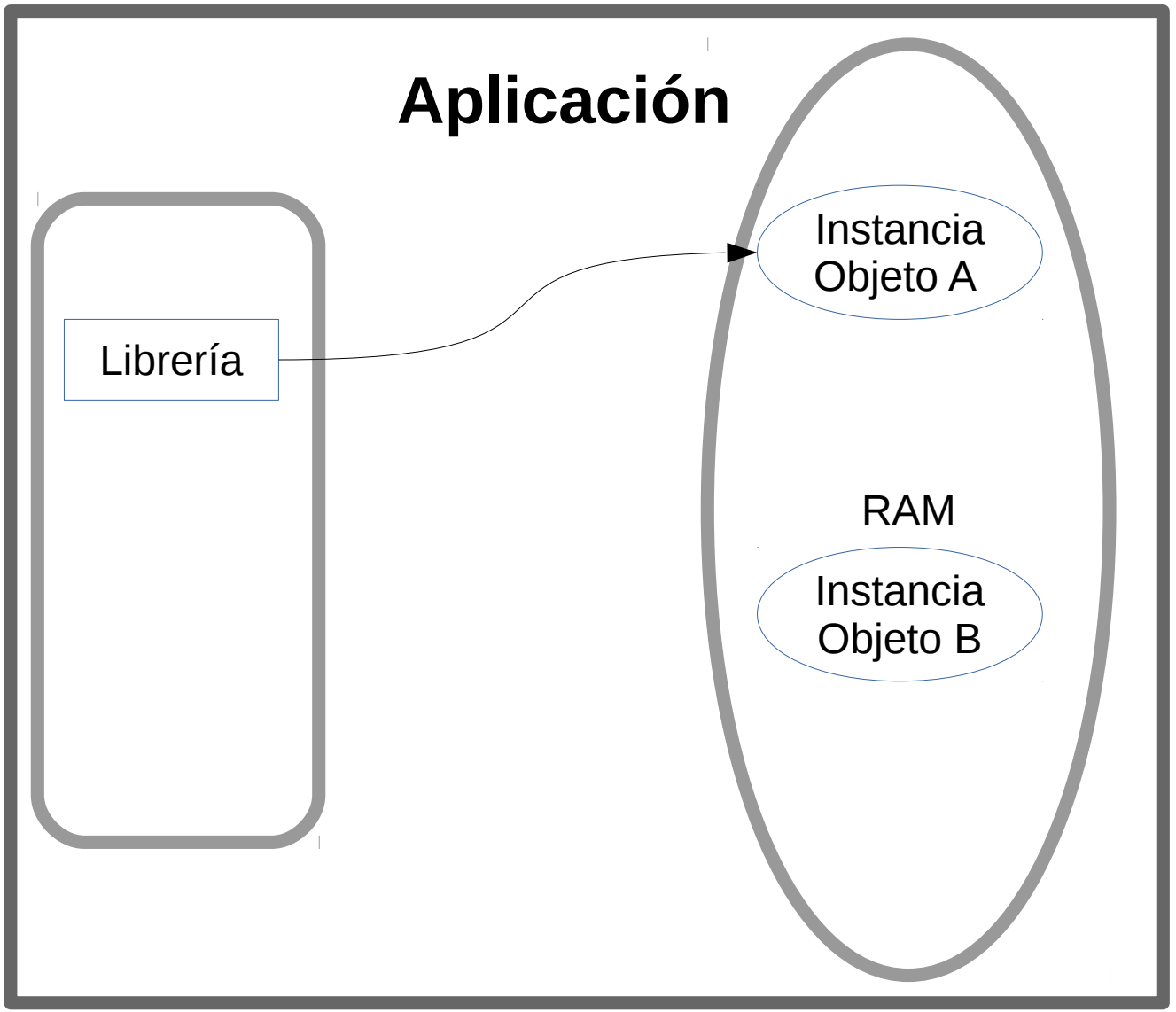


¿Como funciona?



Cuando se invoca el recolector de basura, recorre la lista de espacios reservados observando el contador de referencias de cada espacio.

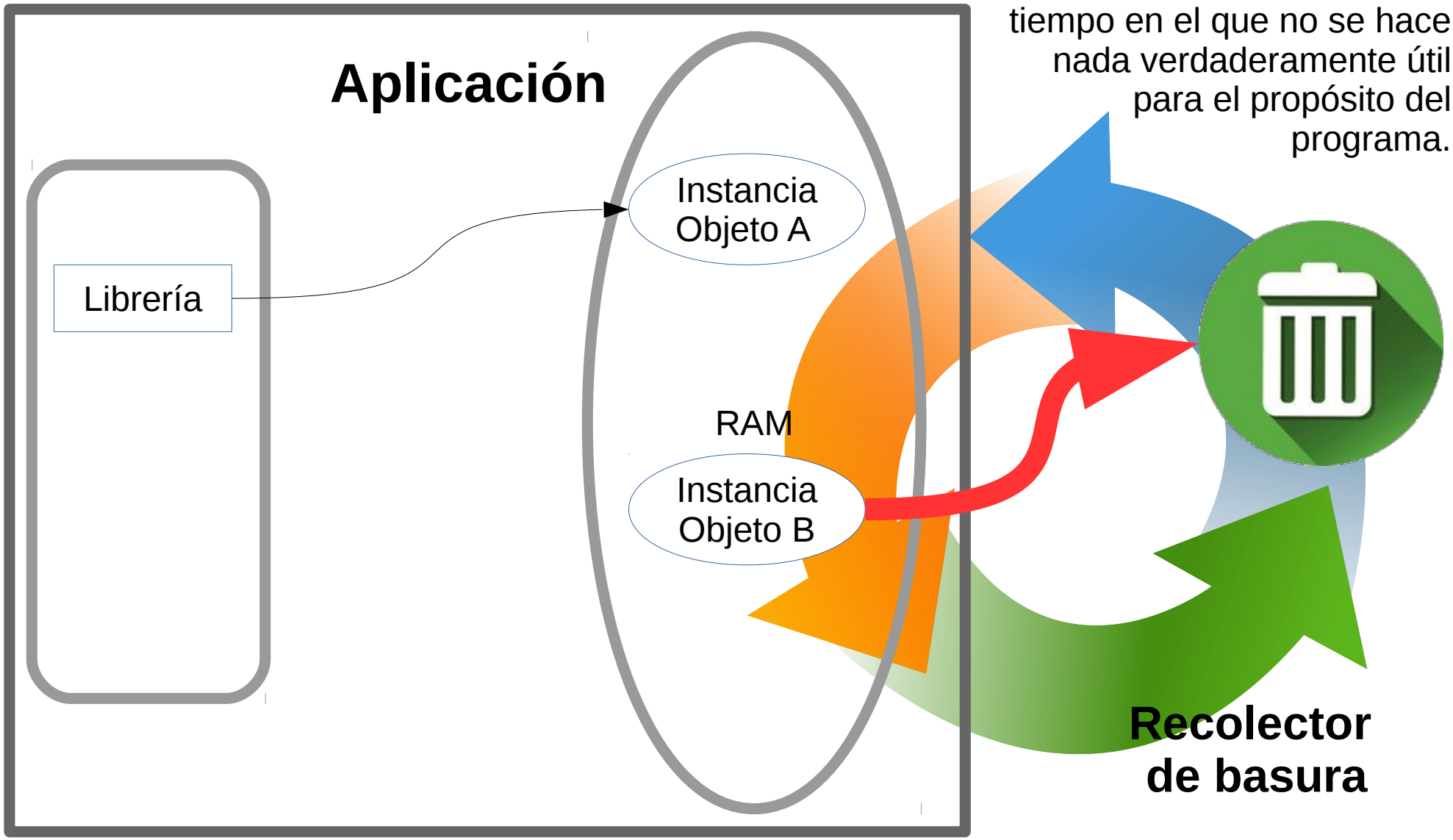
¿Como funciona?



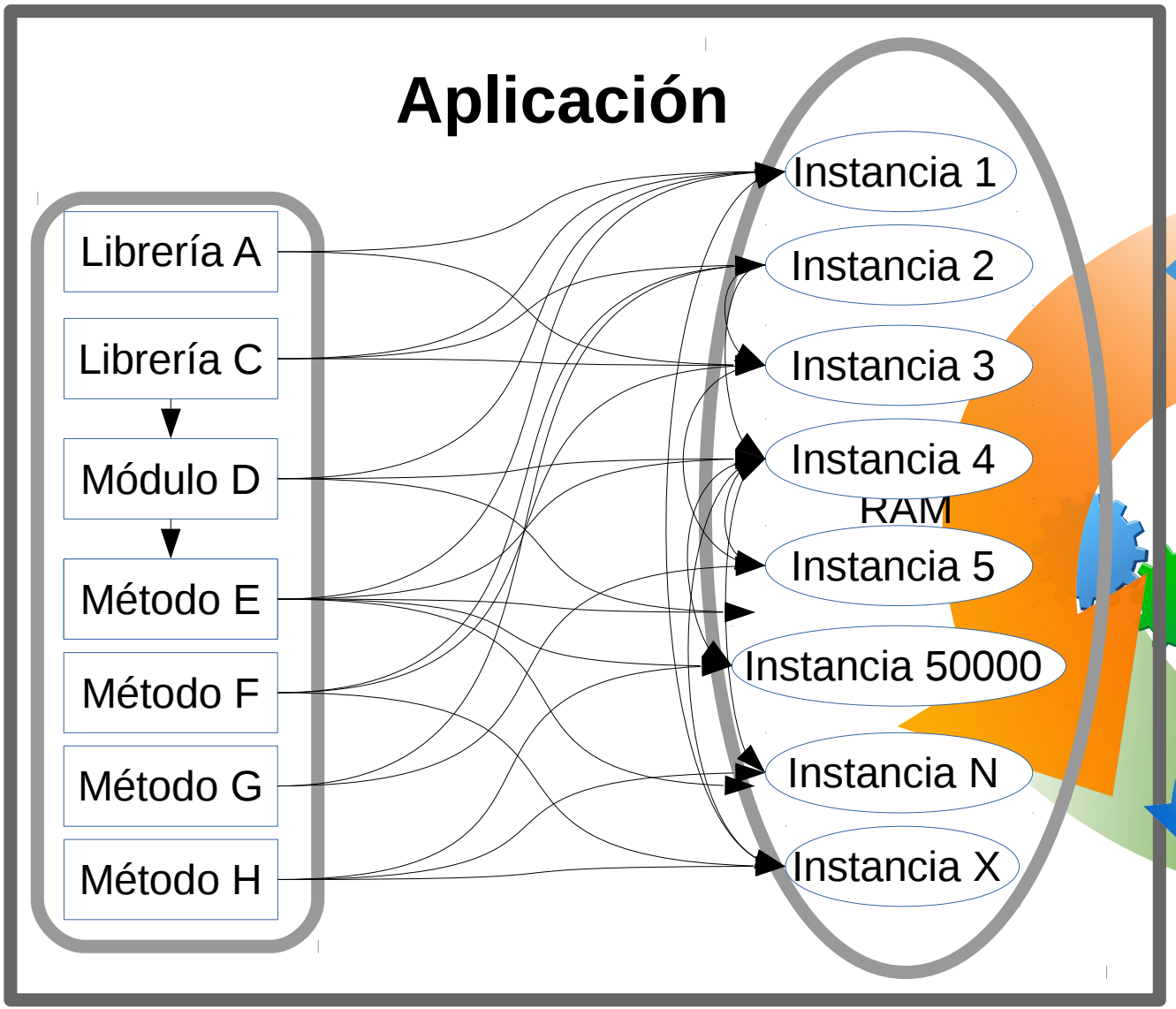
Si un contador ha llegado a cero significa que ese espacio de memoria ya no se usa y, por tanto, puede ser liberado.

¿Como funciona?

Este proceso consume cierto tiempo en el que no se hace nada verdaderamente útil para el propósito del programa.



¿Como funciona?



La realidad

Recolector de basura

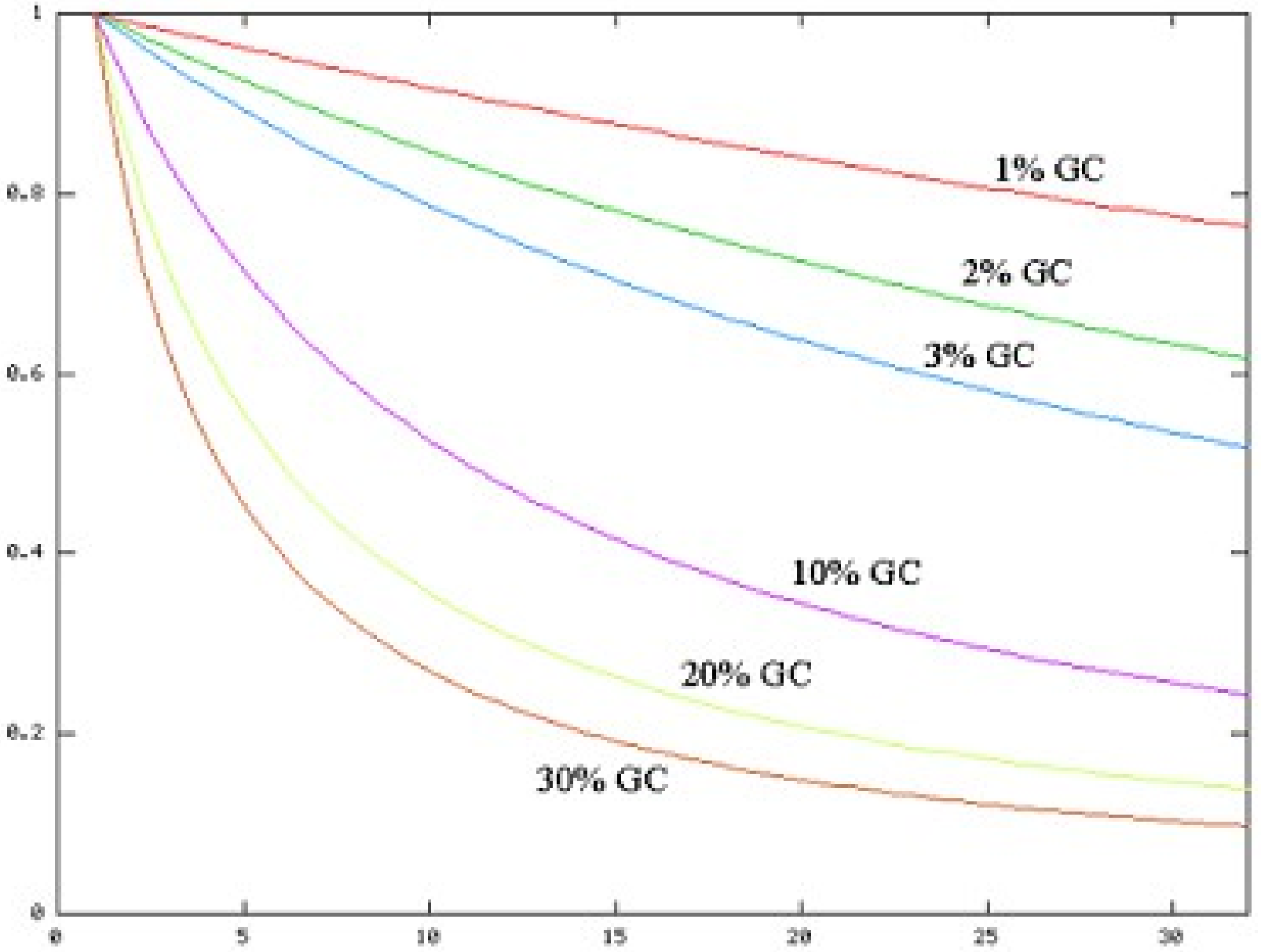
Funcionamiento peligroso para el rendimiento

- No puede ser invocado con demasiada frecuencia.
- En consecuencia, el único inconveniente a este mecanismo es determinar cuándo se tiene que ejecutar el recolector de basura
- Existen varios algoritmos para hacerlo, pero no todos son igual de eficientes.

Algoritmos de recolección de basura

- Esperar a que no quede memoria libre, y entonces, ejecutar el recolector de basura.
- Fijar un umbral de ocupación de la memoria libre y ejecutar el recolector de basura cuando se supere dicho umbral.
- Ejecutar el recolector de basura a intervalos regulares (no siempre es posible).
- Ejecutar el recolector de basura justo antes de cada reserva de memoria.
- Permitir al programador que invoque explícitamente al recolector de basura cuando quiera.

Impacto en el rendimiento



Un poco más de cómo funciona

- La reserva de memoria también es más o menos automática sin la intervención del programador.
- En los lenguajes orientados a objetos: se reserva memoria cada vez que el programador crea un objeto, pero éste no tiene que saber cuánta memoria se reserva ni cómo se hace esto.
- Cuando se compila el programa, automáticamente se incluye en éste una subrutina correspondiente al recolector de basura.

Implementación

- Existe la posibilidad de implementar la recolección de basura como una biblioteca de código más, pero por norma general no es así.
- El propio diseño de ciertos lenguajes de programación hace necesaria la existencia del recolector de basura, como en Java.
  - Que el compilador proporcione la información necesaria para el recolector de basura (el contador de referencias).
  - Que el entorno de ejecución o máquina virtual implemente la subrutina del recolector de basura, como en Java.

El recolector de basura de Java. Ejemplo 1

- Marcar objetos para ser elegibles (para el recolector de basura)
- Un objeto es elegible cuando no existe un hilo vivo en el que exista alguna referencia hacia el.

```
public class Objeto{
    public static void main(String[] args){
        Objeto o = new Objeto();
       /* algunas cosas*/
        o = null;
        //otras
    }
}
```

El recolector de basura de Java. Ejemplo 2

```
public class Objeto{
    public static void main(String[] args){
        Objeto o = new Objeto();
        objeto o1=o;
        /* algunas cosas*/
        o = null;
        //otras cosas
    }
}
```

El recolector de basura de Java. Ejemplo 3

```
public class Objeto{
    public static void main(String[] args){
        Objeto o = new Objeto();//1
        /* algunas cosas*/
        o = new Objeto();//2
                //algunas cosas mas
    }
}
```

El recolector de basura de Java. Invocación

El método
System.gc();

Invocar al recolector de basura

El recolector de basura de Java. Evento de destrucción

El método **finalize()**

Es un método heredado de Object y es llamado cuando un objeto es “recolectado”

```
1  package ejemFinalize;
2
3  public class ObjetoPrueba {
4
5      private static int cantInstancias = 0;
6
7      public ObjetoPrueba(){
8          cantInstancias++;
9          System.out.println(cantInstancias);
10     }
11
12     public void finalize(){
13         cantInstancias--;
14     }
15
16 }
17
```

El recolector de basura de Java.

Diferencias de funcionamiento entre estructuras de datos diferentes.

- Interfaz Set
  - Un conjunto en Java es una colección de elementos que no permite elementos duplicados
  - **HashSet** es una clase que implementa la interface SET basada en una tabla hash
- Interfaz List
  - La clase **ArrayList** en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays
  - Permite referencias duplicadas al mismo objeto