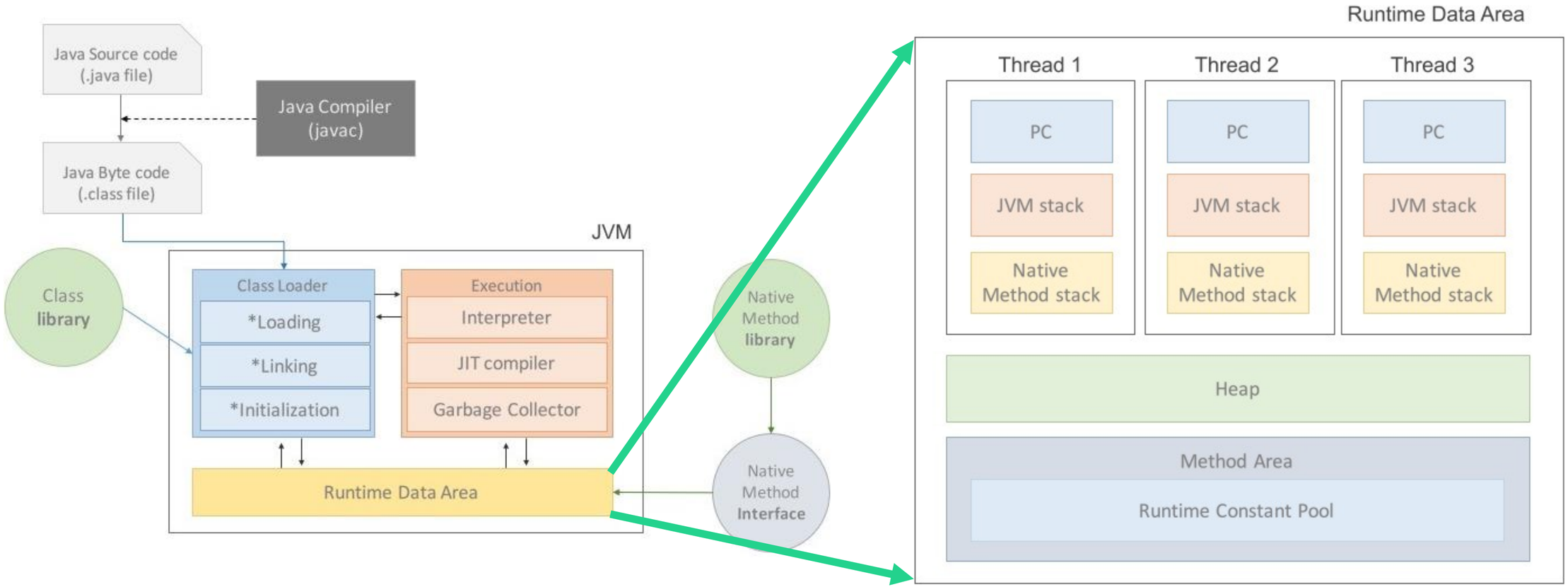
# CS 스터디 19주차

## 복습 2

### JAVA 가비지와  SSR & CSR

# Java의 **가비지 컬렉션**

## JVM > Runtime Data Area > Heap Area

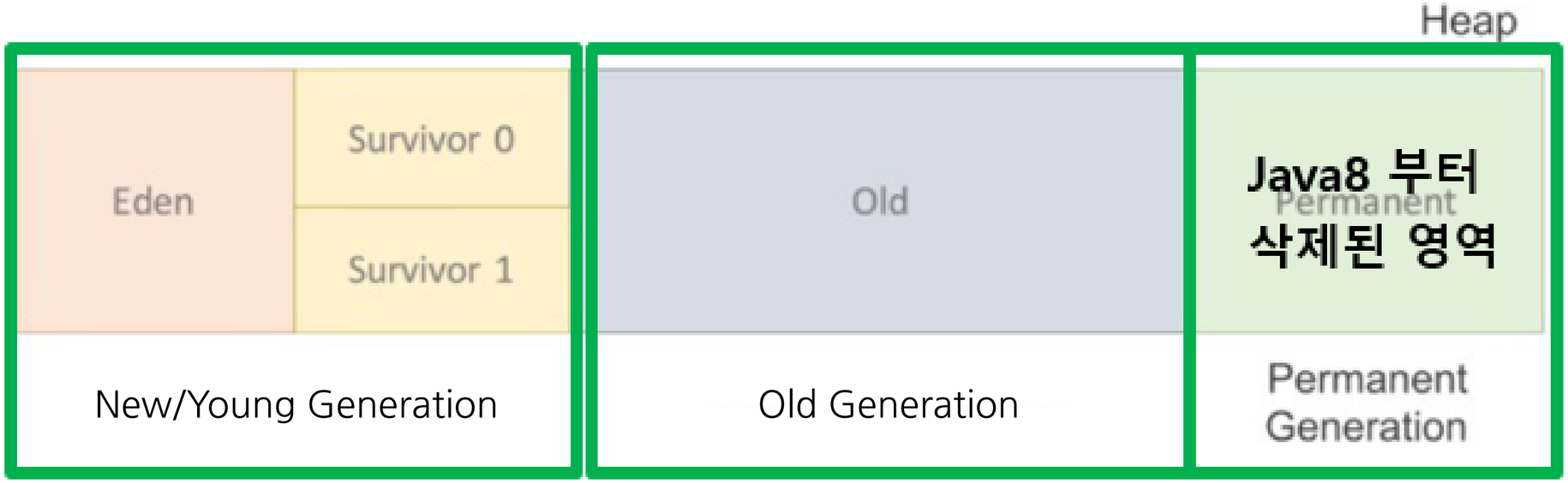


# Java의 **가비지 컬렉션**

## Heap Area

**객체를 저장하는 가상 메모리 공간**

**'객체는 대부분 일회성이며, 메모리에 오랫동안 남아있는 경우는 드물다'** 라는 전제로
객체의 생존 기간에 따라 Heap의 영역을 나눔



# Java의 **가비지 컬렉션**

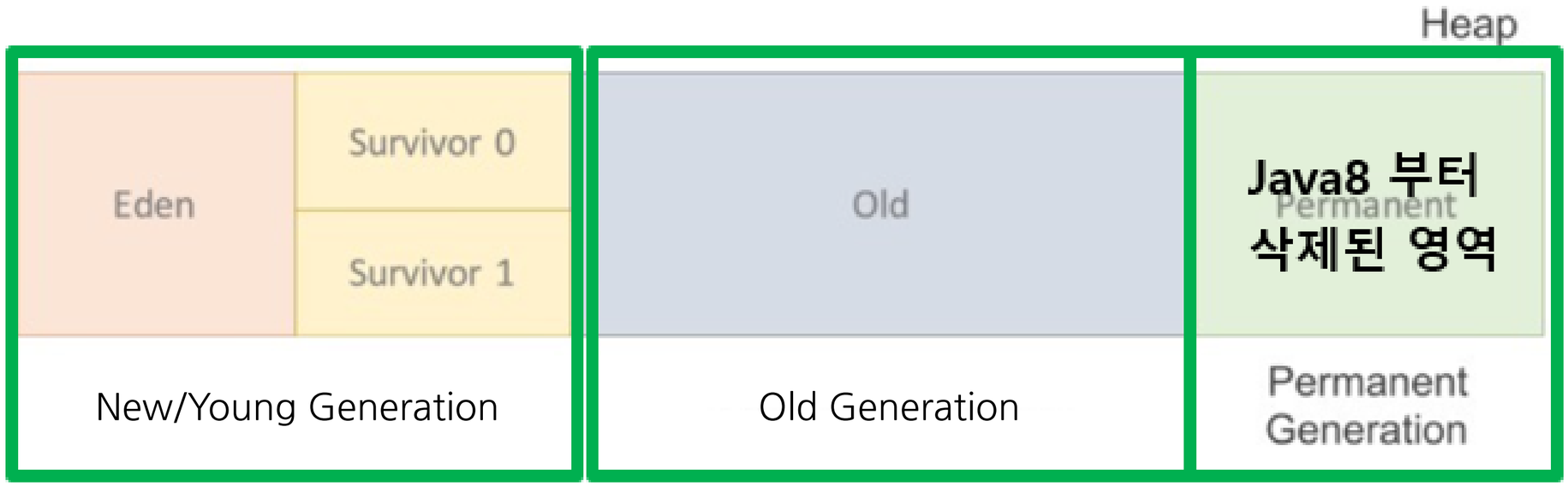
## Heap Area > New/Young Generation

**새롭게 생성된 객체가 할당되는 영역**

Eden : 객체들이 최초로 생성되는 공간

Survivor 0,1 : Eden에서 참조되는 객체들이 저장되는 공간, 최소 1번의 GC 이상 살아남은 객체가 존재하는 영역

New/Young 부분에 대한 GC를 Minor GC라 함

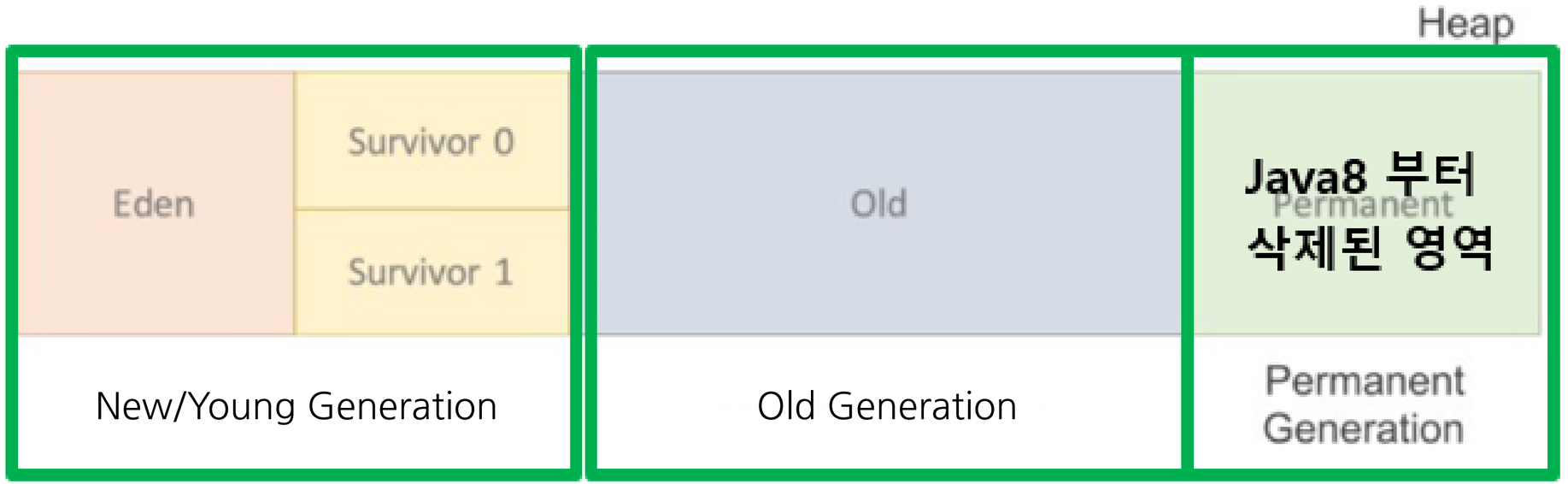


# Java의 **가비지 컬렉션**

## Heap Area > Old Generation

**일정시간 이상 참조되고 있는, 살아남은 객체들이 저장되는 공간**

Old 부분에 대한 GC를 Major 또는 Full GC라 함



# Java의 **가비지 컬렉션**

## Garbage Collection

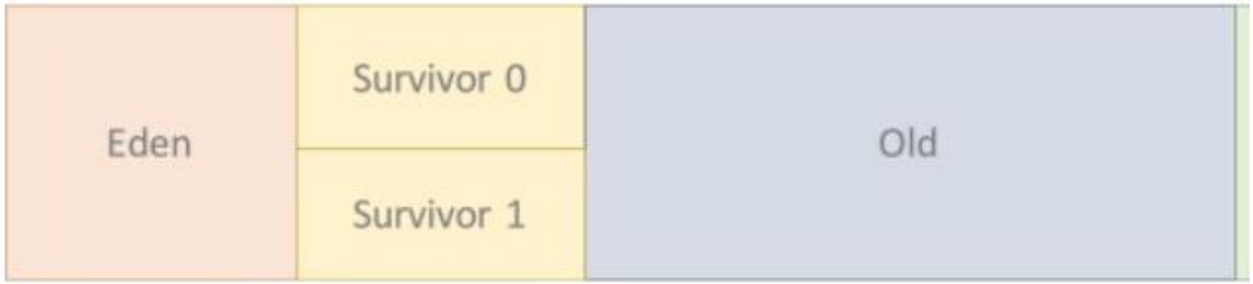
**참조되지 않는 객체들을 탐색 후 삭제,**
**삭제된 객체의 메모리를 반환해 Heap 메모리의 재사용을 할 수 있게 해줌**

## 가비지 컬렉터가 하는 일

1. 메모리 할당
2. 사용중인 메모리 인식
3. 사용하지 않는 메모리 인식

# Java의 **가비지 컬렉션**

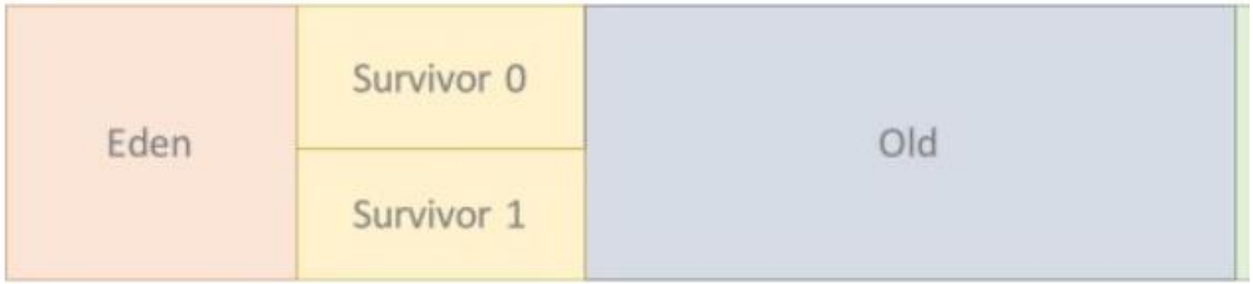
## 객체의 이동 순서와 Minor GC & Major GC




1. 객체가 생성되면 Eden 영역에 객체가 지정됨
2. Eden에 데이터가 어느정도 쌓이면 Survivor 0, 1로 객체가 옮겨짐
3. 더 큰 객체가 생성되거나 더이상 **New/Young Generation** 에 공간이 남지 않으면 객체들은 **Old Generation** 영역으로 이동

# Java의 **가비지 컬렉션**

## 객체의 이동 순서와 Minor GC & Major GC




**1. Old Generation**이 꽉 찼을 경우 **Old Generation**에 있는 모든 [illegible]

2. 참조되지 않는 객체를 한꺼번에 삭제

**Stop-the-world 발생**

# Java의 **가비지 컬렉션**

## Garbage Collection의 과정


**1. Heap내의 객체 중 가비지를 찾아냄 (Mark)**

모든 오브젝트를 탐색하기 때문에 시간 오래걸린다.

**2. 가비지를 제거하고, 메모리를 반환 (Sweep)**

반환되어 비어진 메모리의 참조 위치를 저장해 두었다가, 새로운 객체가 선언되면 이곳에 할당되도록 한다.

**3. 남은 참조되는 객체를 묶음**

새로운 메모리 할당 시 더 쉽고 빠르게 할당 진행 가능

# Java의 **가비지 컬렉션**

## 가비지 컬렉션의 한계

1. 어떤 방식의 알고리즘을 사용하던 실행 시간에 작업을 하는 이상 성능 하락을 피할 수 는 없다.

2. 가비지 컬렉터가 존재하더라도 더 이상 접근이 불가능한 객체만 회수하기 때문에 메모리 누수는 발생할 수 있다. (완전히 모든 메모리 재사용 불가)

# SSR & CSR

## 렌더링?

서버에 요청해서 받은 내용을 브라우저 화면에 표시하는 것

# SSR & CSR

## 서버 사이드 렌더링


사용자에게 보여 줄 페이지를 서버에서 모두 구성하여 보여주는 방식

# SSR & CSR

## 클라이언트 사이드 렌더링


서버는 처음에 하나의 빈 HTML 파일을 주고, 이후에 클라이언트에서 페이지를 구성해 사용자에게 보여주는 방식

# SSR & CSR

## 서버 사이드 렌더링의 장점

- CSR보다 첫 페이지 로딩이 빠르다.
- SEO (검색 엔진 최적화)을 쉽게 구성할 수 있다.

# SSR & CSR

## 클라이언트 사이드 렌더링의 장점

- 구동 속도가 빠르다.
- TTV와 TTI 사이의 간극이 없다.
- 서버 부하 분산

# SSR & CSR

## SEO

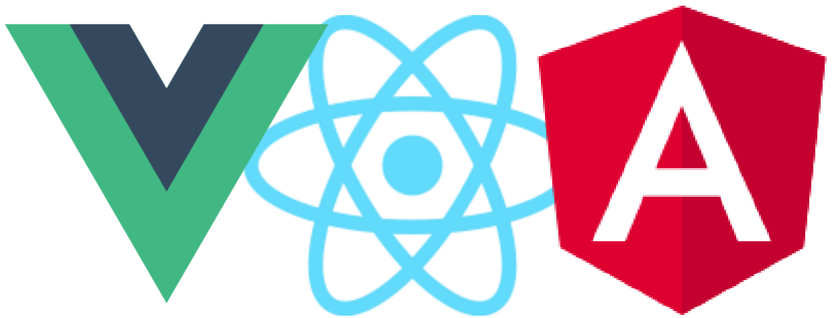
```html
<html>
<head>
  <title>Single Page Application</title>
  <link rel="stylesheet" href="app.css" type="text/css">
</head>
<body>
  <div id="app"></div>
  <script src="app.js"></script>
</body>
</html>
```

검색 로봇 입장에 보는 CSR로 렌더링 된 페이지

# SSR & CSR

## SPA

한 개의 페이지로 구성된 어플리케이션
하나의 페이지에 웹사이트의 전체 페이지를 담아야 함
CSR 방식으로 렌더링



# SSR & CSR

## MPA

페이지가 여러 개로 구성되어 있는 어플리케이션
SSR 방식으로 렌더링함

# SSR & CSR

## CSR+SSR

Express.js, nest.js 사용 (node.js기반 웹 프레임워크)

NEXT.js, Gatsby.js, NUXT.js, Universal

# SSR & CSR

## 서비스의 성격에 따라 달라지는 렌더링 방법

유저와의 상호작용 多, 고객의 개인정보가 많이 필요한 웹 ： CSR

누구에게나 항상 같은 내용, 매주 업데이트 되며 회사 홈페이지와 같은 웹 : SSR

사용자에 따라 페이지 내용이 달라지고, 빠른 인터렉션과 화면 깜빡임이 없어야 하고
검색의 상위에 노출되고 싶다면 CSR + SSR

꿋 :)