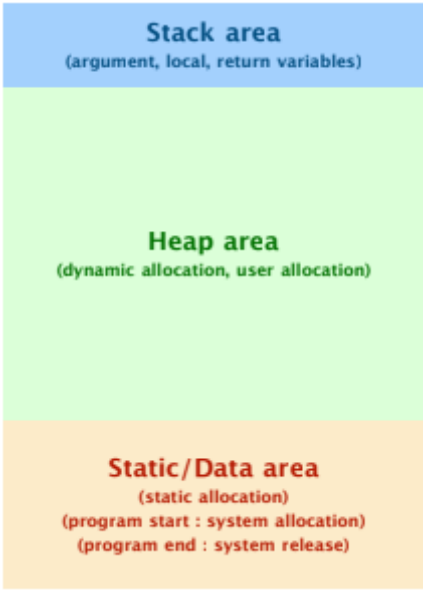
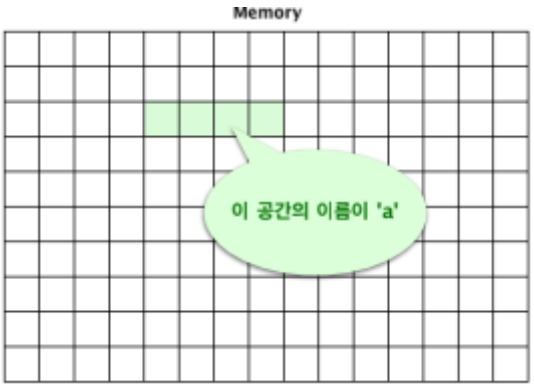
메모리 구조



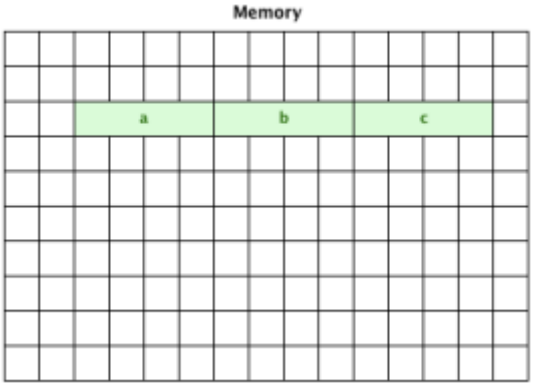
Stack Area ☝

- 자바에서 함수를 구성하다보면 함수 내에 수많은 변수를 생성하게 되는데. 함수(main) 내의 로컬 변수(지역변수), 그리고 반환값이 있을 경우 그 반환 변수까지 저장되는 공간이다.🐬
- 이 영역은 일시적인 저장공간이다. 즉, 공간의 생성도 시스템이 알아서 만들어 내고, 함수의 실행이 종료되면 해당 공간도 바로 없어져 버린다.🏃
- 휘발성이 강한 영역이므로, 해당영역의 크기가 크지 않다. 대략 1MB(1000000 byte)의 크기를 가진다고 생각하면 될듯하다. -> 따라서 지역변수로 크기가 큰 배열의 생성은 어렵다고 할 수 있다.🙂
- 시스템 설정에서 "스택 영역"을 보다 크게 설정할 수 있다.
- 스택 오버 플로우 : 설정된 스택 크기 이상의 메모리를 할당하려 한다면 에러가 발생하는데, 할당된 양이 흘러넘쳐서 'heap area'영역을 침범하게 된다. 이럴때 발생하는 에러가 그 유명한 **스택 오버플로우**이다. 🤞🤞



> 변수를 선언해서 스택영역에 공간을 생성한 모습.🔎

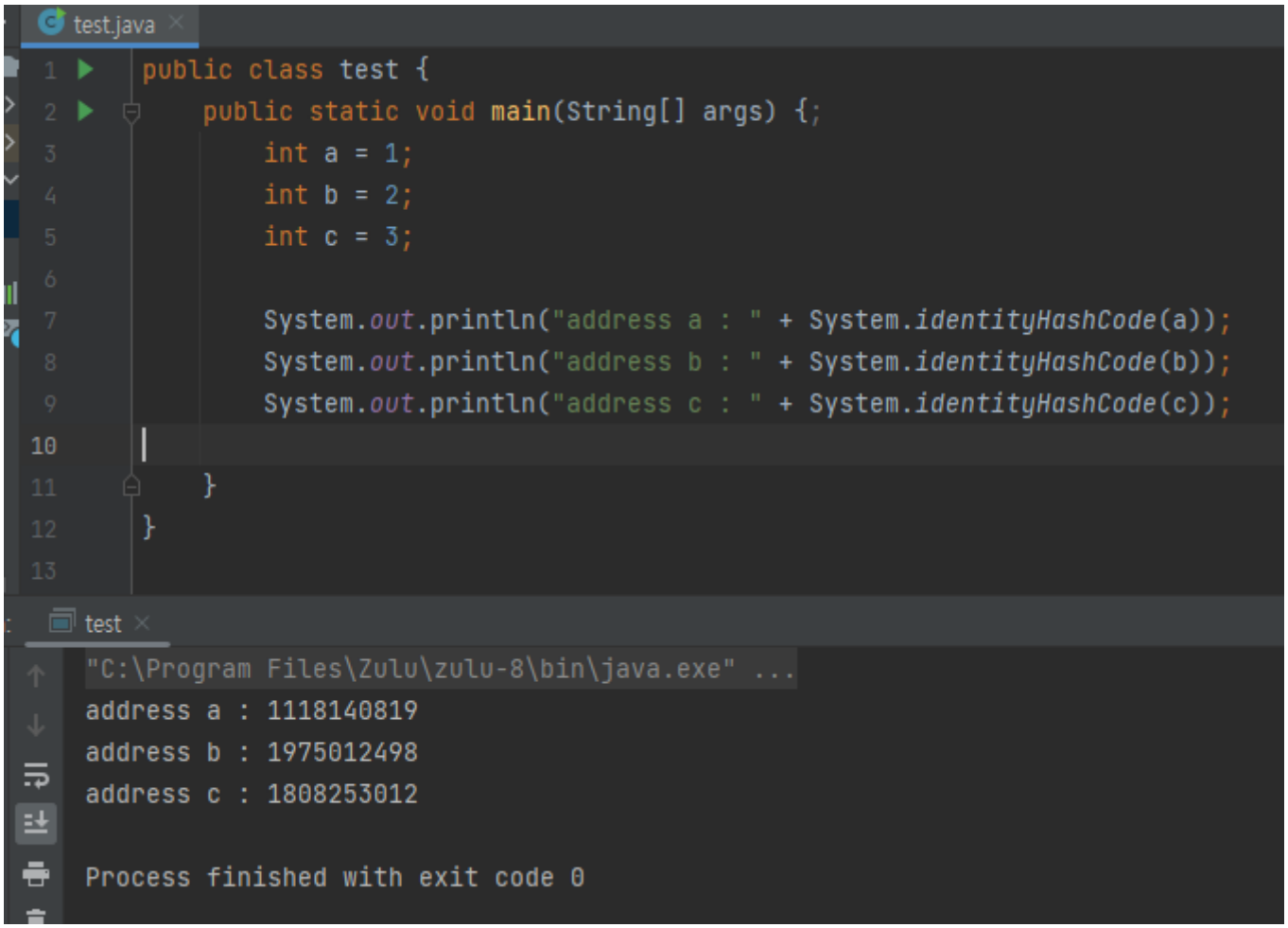
```
public class Main{
  public static void main(String args[]){
    int a;    //4byte 공간을 생>
  }
}
```



> 🔥메모리는 절대 이런식으로 연속적으로 잘 정리되어 우리가 보기 쉽게 공간을 형성하는 게 아니다!!!🔥

```
public class Main{
  public static void main(String args[]){
    int a;
    int b;
    int c;
  }
}
```

> 실제 저장은 저 메모리 속안에 흩어져서 저장이 되게 되고, 주소값을 알수 있다.



> 실제 주소값들을 뽑아보면 위와 같이 4byte의 공간 차이가 아닌 전혀 뜬금없는 주소에 저장된걸 알 수 있다.

Heap Area ☝️

- 메모리에서 가장 많은 영역을 차지하는 구역이다. 🥶
- 보통 **"동적 할당"** 이라고 하며, 이 영역에 공간을 만들게 된다.
- 사용자에 의해 아주 많은 공간이 필요하다고 판단되는 경우, 힙 영역을 사용하자 👌
- *사용자가 할당* 하였기 때문에 *사용자가 해제*해야 한다‼️

👍👍👍

> 하지만, 자바 언어의 특성상 사용되지 않는 메모리의 영역을 알아서 해제시켜주는 기능이 있다.
>
> 이런한 함수를 가비지 컬렉션(쓰레기 모음)이라고 하는데 이런 함수를 통해 시스템이 알아서 해제시키도록 처리함.

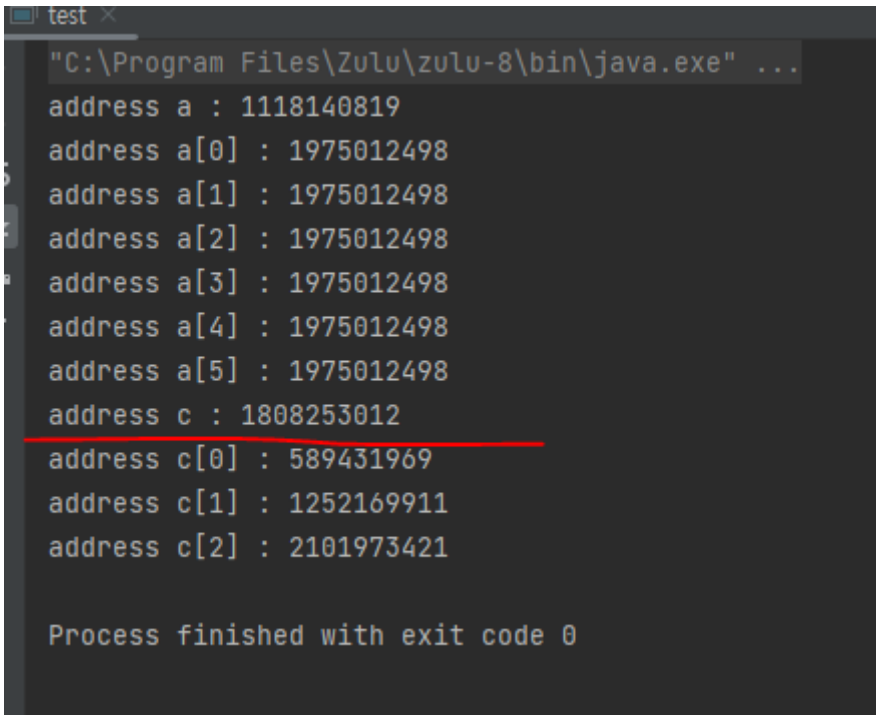
- "new"연산자를 이용해 동적 할당이 가능.

```java
public class test {
    public static void main(String[] args) {;
        int[] a = new int[6];     // 1차원 배열

        System.out.println("address a : " + System.identityHashCode(a));
        for(int i = 0; i< a.length; i++)
            System.out.println("address a[" + i + "] : " + System.identityHashCode(a[i]));

        int[][] c = new int[3][10];

        System.out.println("address c : " + System.identityHashCode(c));
        System.out.println("address c[0] : " + System.identityHashCode(c[0]));
        System.out.println("address c[1] : " + System.identityHashCode(c[1]));
        System.out.println("address c[2] : " + System.identityHashCode(c[2]));
    }
}
```



살펴보기 🔎🔎🔎

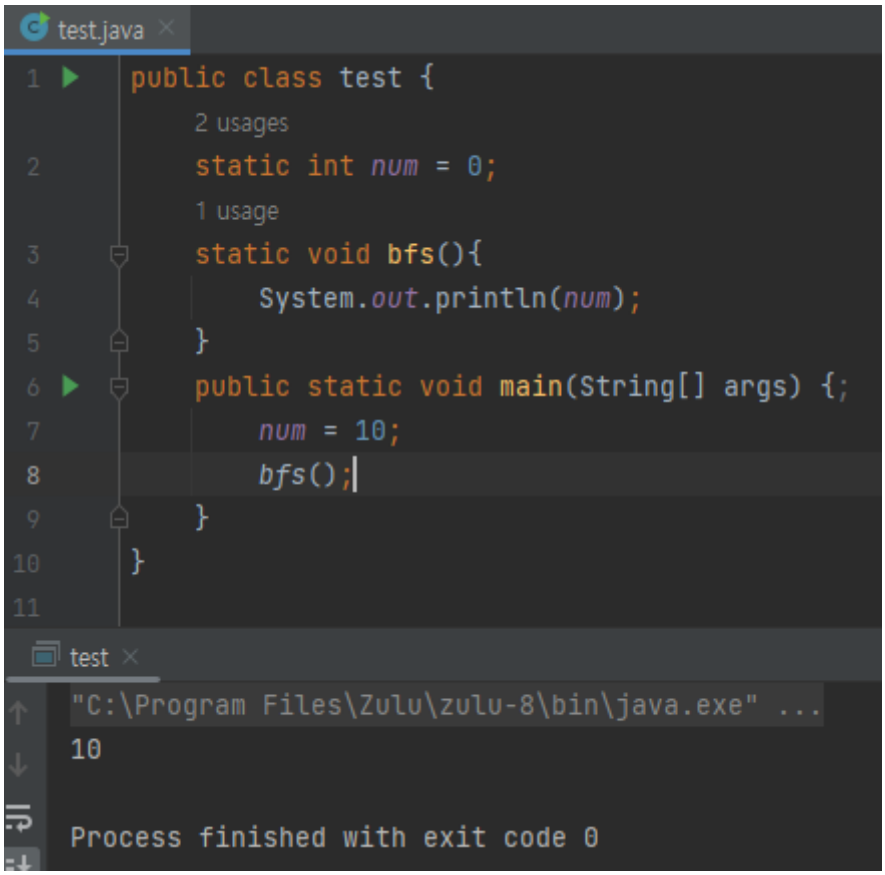
> 1차원 배열의 경우 주소값이 스택영역에 설정되고 그 주소를 따라가 **힙영역**에는 연속된 공간으로 6개의 공간이 생성되므로, 스택 영역에서 참조되는(참고하는) 주소의 값이 모두 동일한 것을 볼 수 있다.
>
> 2차원 배열의 경우에는 1차원 배열들의 모임이라고 생각하면 쉽다. 각각의 1차원 배열들의 주소를 모아논 공간이 있고, 그 공간의 주소가 애초에 **스택 영역**이 저장된다. (스택 -> 힙)

> 그 주소를 따라가면 힙 영역에 1차원 배열의 갯수만큼 주소들을 모아논 공간이 존재하고, 다시 그 주소를 참고하여 또 다른 **힙영역**에 존재하는 1차원 배열을 찾아가게 된다. (힙-> 힙)

Static/Data Area ☝

- 정적 변수(static variable), 전역 변수(global variable)로 구성된 변수들이 생성되는 공간.🐬
- 프로그램이 실행될 때 시스템에 의해 공간이 할당되고, 프로그램이 종료될 때 시스템이 개발자가 만든 공간을 해제시킴.
- 프로그램이 작동하는 동안에는 아 정적으로 끝날때까지 그 공간이 유지되는 영역이다.
- 해당 크기는 그림에서 보다시피. 스택영역보다는 살짝 크고, 힙 영역보다는 작다. 따라서 해당 영역의 편리성 때문에 마구잡이로 정적변수를 설정 하다가는 큰일 날 수 있으므로, 적당히 사용하는 게 좋다.💔

```
public class test {
    static int num = 0;     // 스택 영역의 변수
    static void bfs(){      // 스택 영역의 함수
        System.out.println(num);
    }
    public static void main(String[] args) {;
        num = 10;
        bfs();
    }
}
```



정리

stack area	heap area	static/data area
함수 호출시 시스템에 의해 생성	임의의 시점에 개발자에 의해 생성	프로그램이 시작될때 시스템에 의해 생
함수가 종료될 때 시스템에 의해 해제	idle 시점에 시스템에 의해 해제	프로그램이 종료될 때 시스템에 의해 해제