[dev-rary(데브러리)] 자바 토막강의

1. 변수에 대해 잘 이해하기.

강의자 소개 : dev-rary

前) 광고데이터 분석, SEO 최적화, SEM 세팅 및 관리

前) 기초프로그래밍, 웹개발, 앱개발 강의

前) 머신러닝을 활용한 데이터 분석

前) 기아자동차 광주공장 외주

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

現) 유튜브 데브러리의 개발도서관 운영중
https://www.youtube.com/channel/UCJJsnLh62qx00sN3yzRcW3Q

변수 끝장내기

변수는 정말 무조건적으로 사용하게 되는 것임에도 불구하고

많은분들이 그저 “감” 으로 사용하는 경우가 많은것 같습니다.

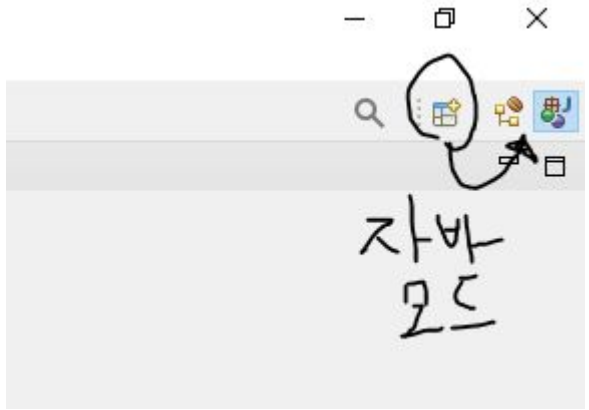
특히 메서드의 파라미터나, 리턴구문에 대한 이해가 부족한 분들은

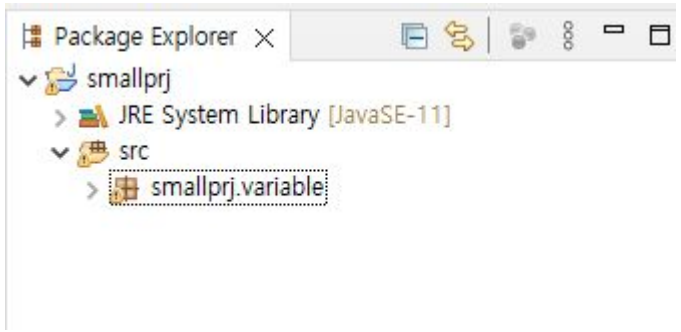
변수를 어떻게 해석해야 하는지 모르고

단지 특정한 값이 저장된 지점이나 명칭 정도로 생각하는 경우가 많습니다.

변수를 제대로 읽기 위한 포인트를 짚어보겠습니다.

프로젝트 구성



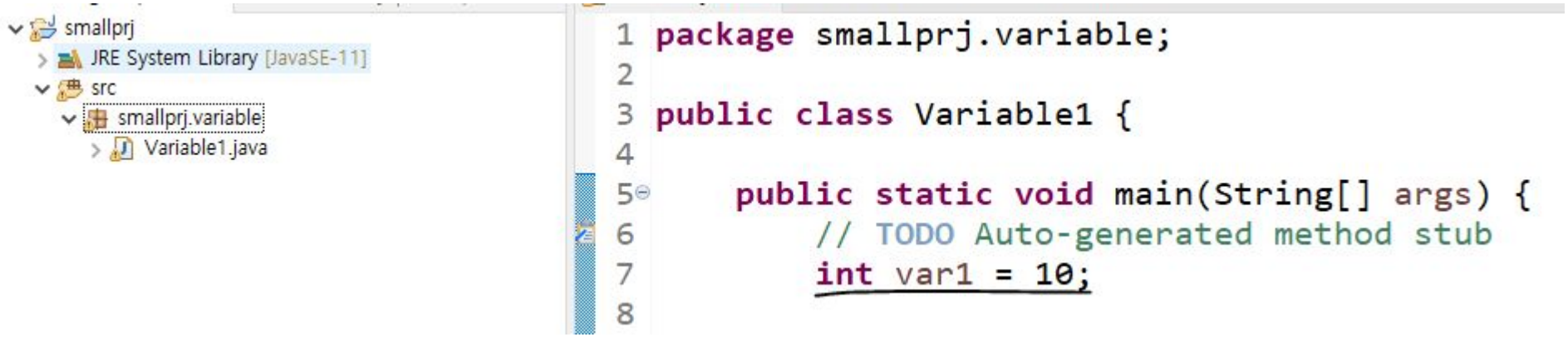


프로젝트 구성입니다.

smallprj 하위 패키지에 토막강의 주제별로 패키지를 생성할 예정입니다.

우측 상단을 클릭해 자바모드로 바꿔주세요.

변수를 해석하는 방법



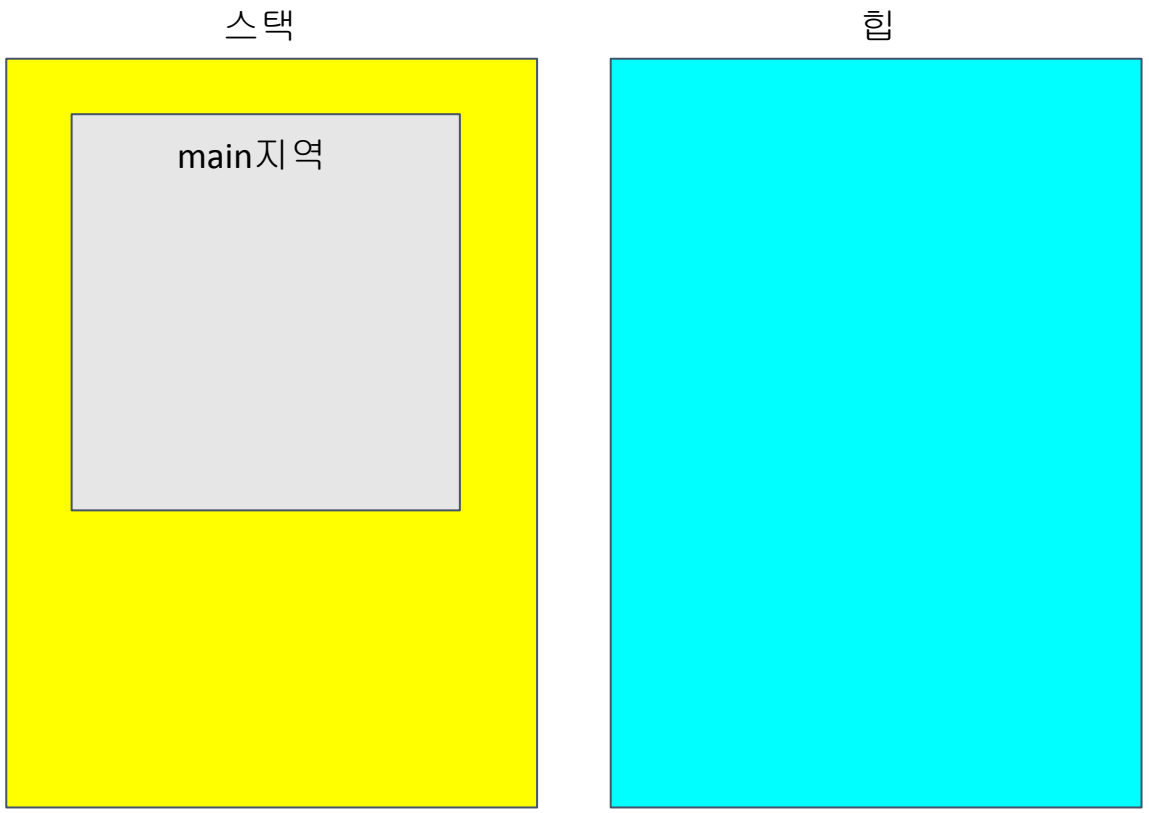
당연하다면 당연한것이지만 변수는 자료를 저장한 공간에 붙는 이름입니다.

자바를 배우면서 거의 처음으로 **“Hello World”** 구문을 찍은 뒤부터

여러분들은 위와 같이 변수를 선언해서 쓰셨을겁니다.

변수를 해석하는 방법

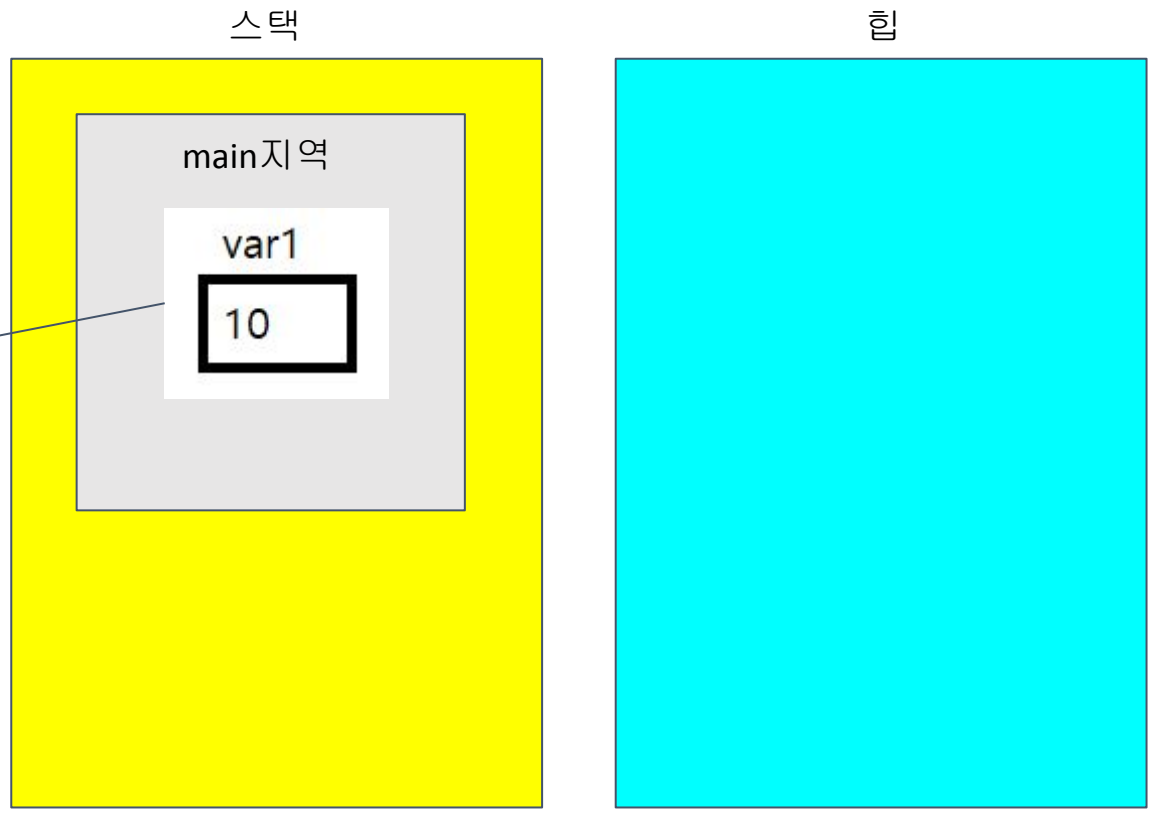

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8
9      }
10
11 }
```



여기서 유념하실 점은 우리가 코드를 실행하면 확정적으로 실행되는 코드는 위와같이 **5~9번라인(main 메서드의 중괄호 여는부분~닫는부분)** 까지이며 우측 메모리영역에는 스택이라는 공간에 **main지역**이 생깁니다.

변수를 해석하는 방법

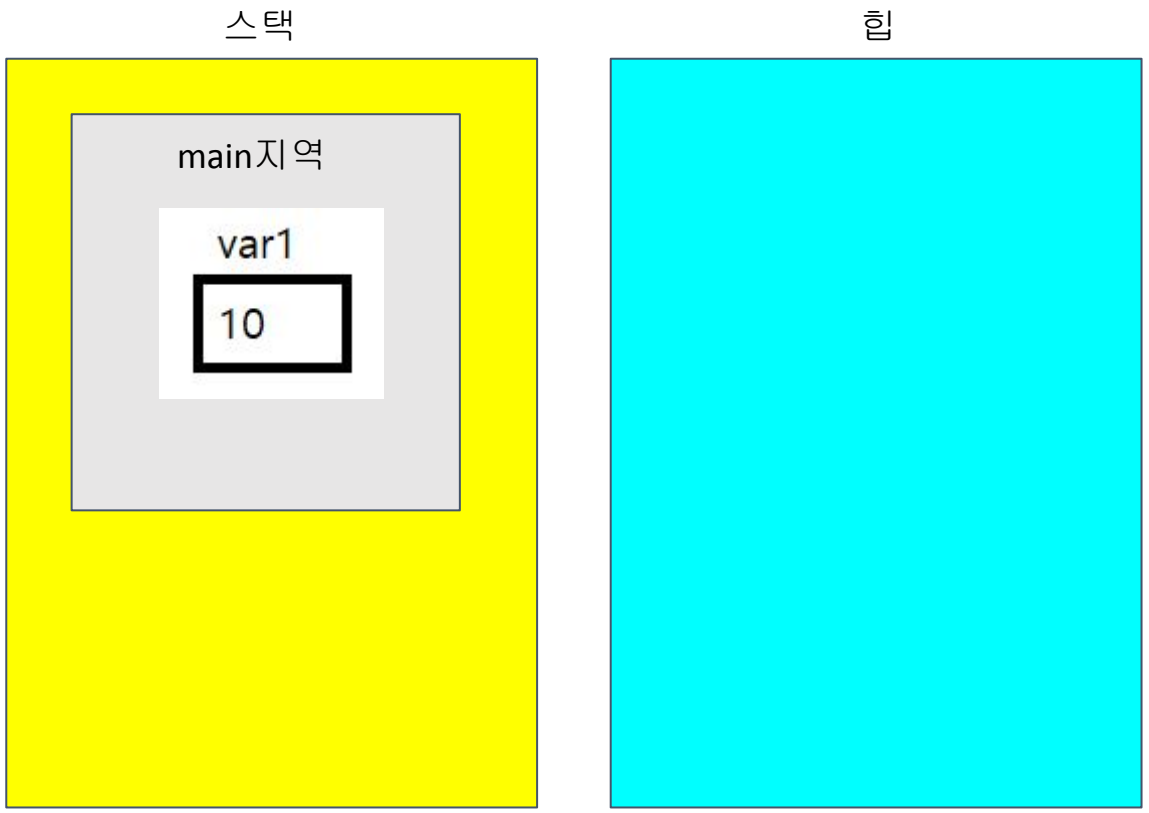

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8
9      }
10
11 }
```



main 지역 내부의 코드는 위에서부터 순차적으로 한 줄 씩 실행됩니다.
즉 컴퓨터는 먼저 5번라인을 인지해서 main영역을 스택에 생성한 다음
6번은 주석이니 건너뛰고 7번을 인식해 main영역(중괄호 여닫는사이)에
var1 이라는 변수를 만들고 거기 10이라는 값을 저장합니다.

변수를 해석하는 방법


```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8
9      }
10
11 }
```

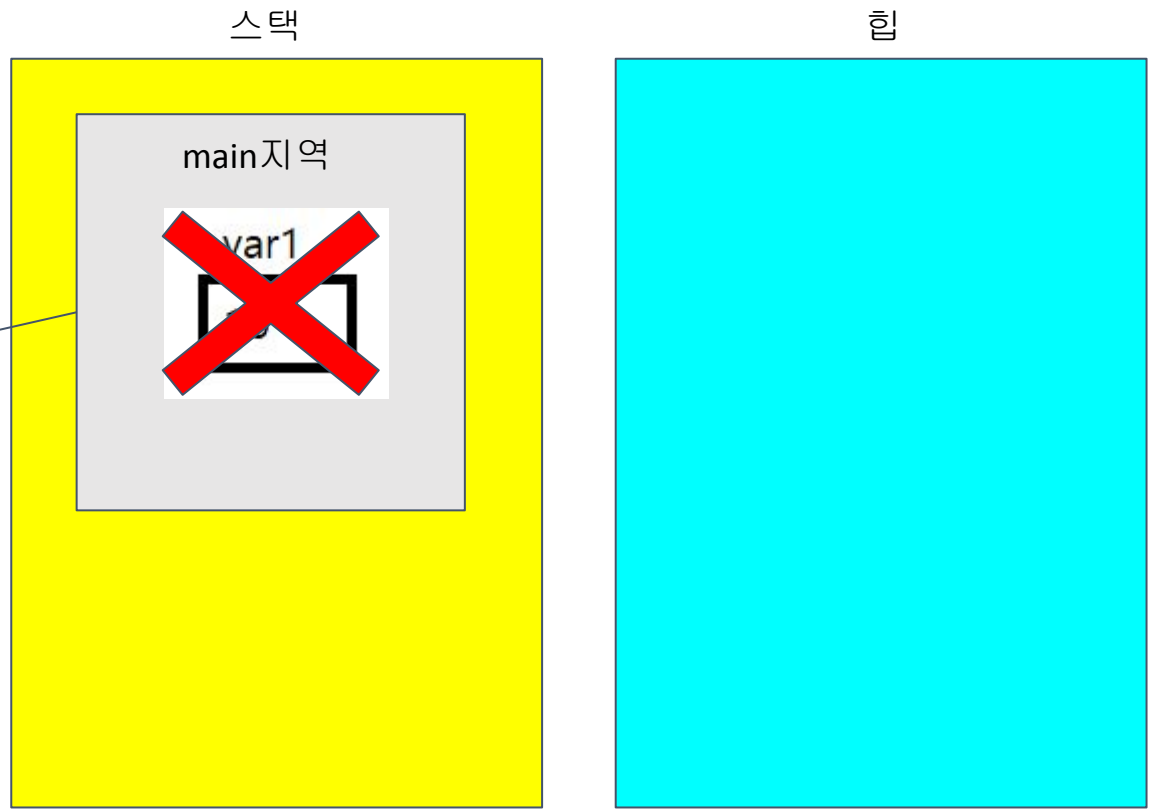


마지막으로 8번라인에 코드가 없으므로 9번 닫는라인으로 넘어갑니다.

이제 닫는부분을 만났다는것은 더 이상 실행할 코드가 없다는 뜻이기도 합니다.

변수를 해석하는 방법

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8
9      }
10
11 }
```

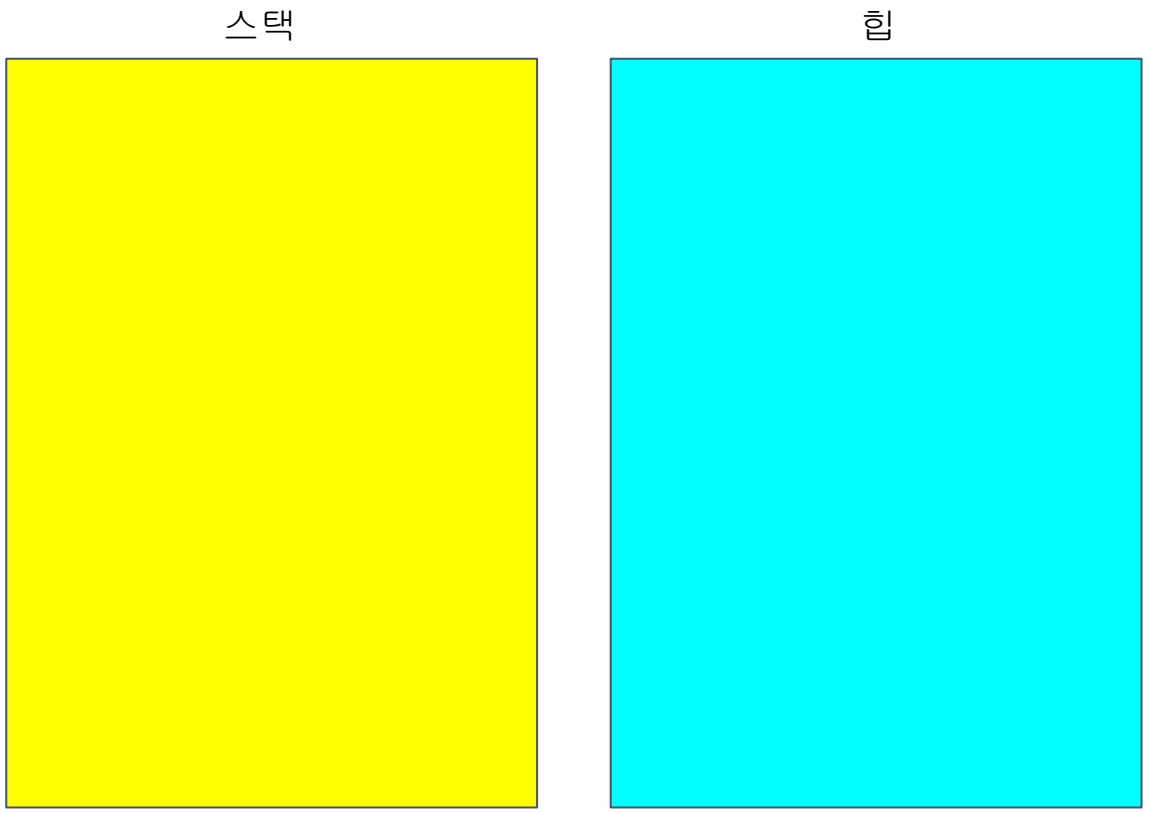


9번라인에서 중괄호가 닫히는게 확인되면

main지역은 그대로 사라집니다.

변수를 해석하는 방법


```
3 public class Variable1 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int var1 = 10;
8
9     }
10
11 }
```



그리고 스택이 비워진 채로 프로그램이 끝나게 됩니다.

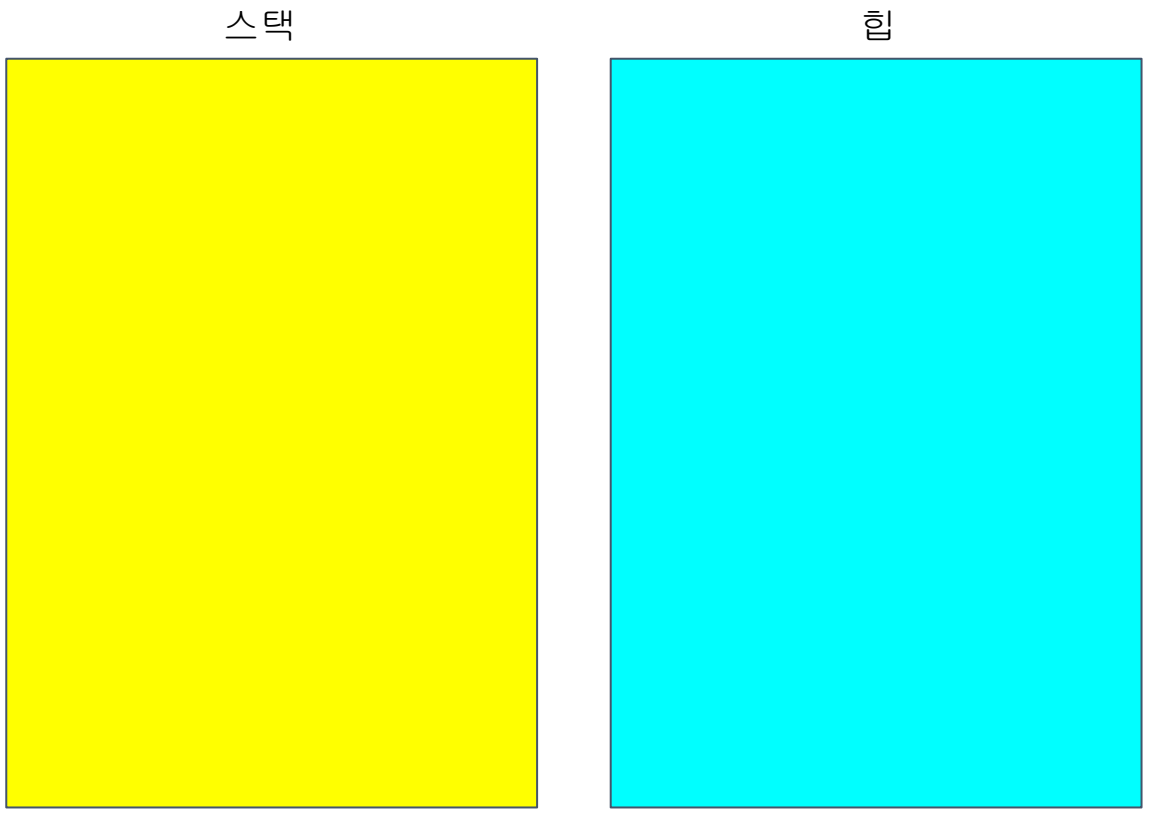
변수를 해석하는 방법

변수가 저장되는 기본적인 원리부터 봤고

다음은 변수를 호출하는 부분을 다뤄보겠습니다.

변수를 해석하는 방법

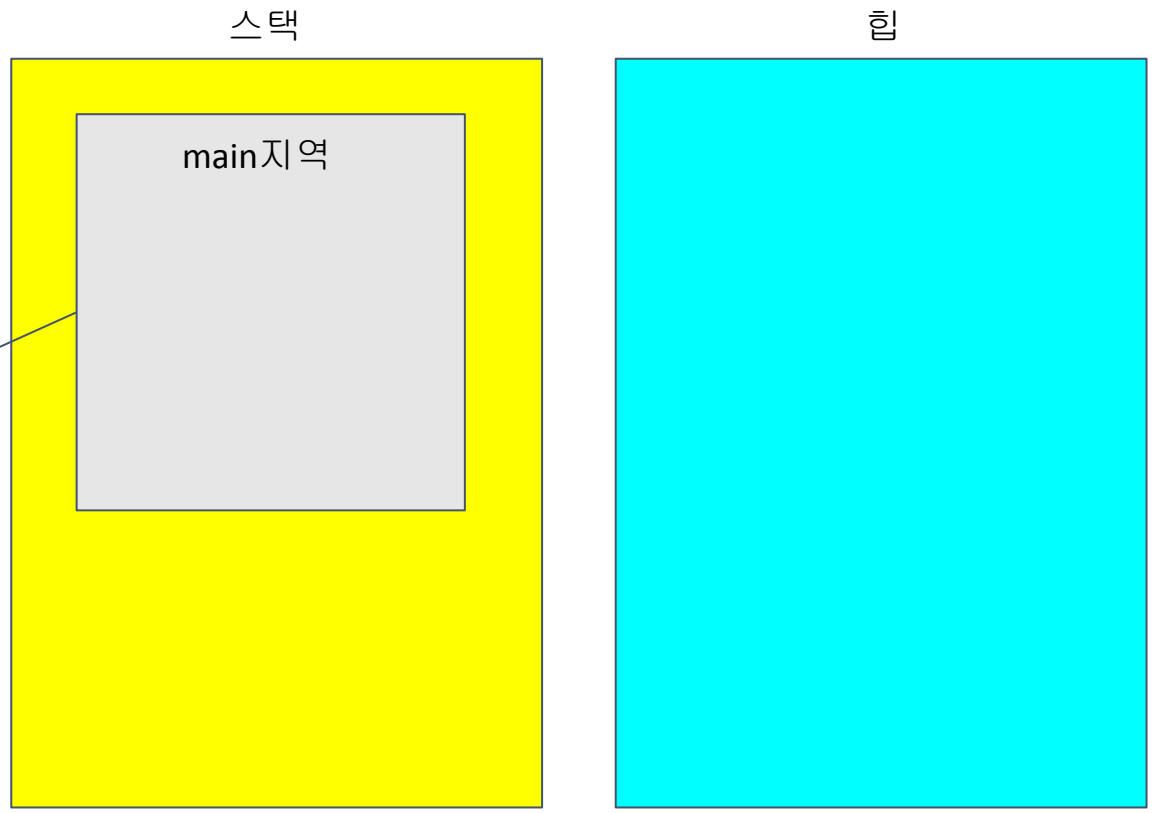

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



이번에는 9번라인에 **var1**에 저장된 변수를 콘솔에 출력하는 내용을 추가합니다.

변수를 해석하는 방법

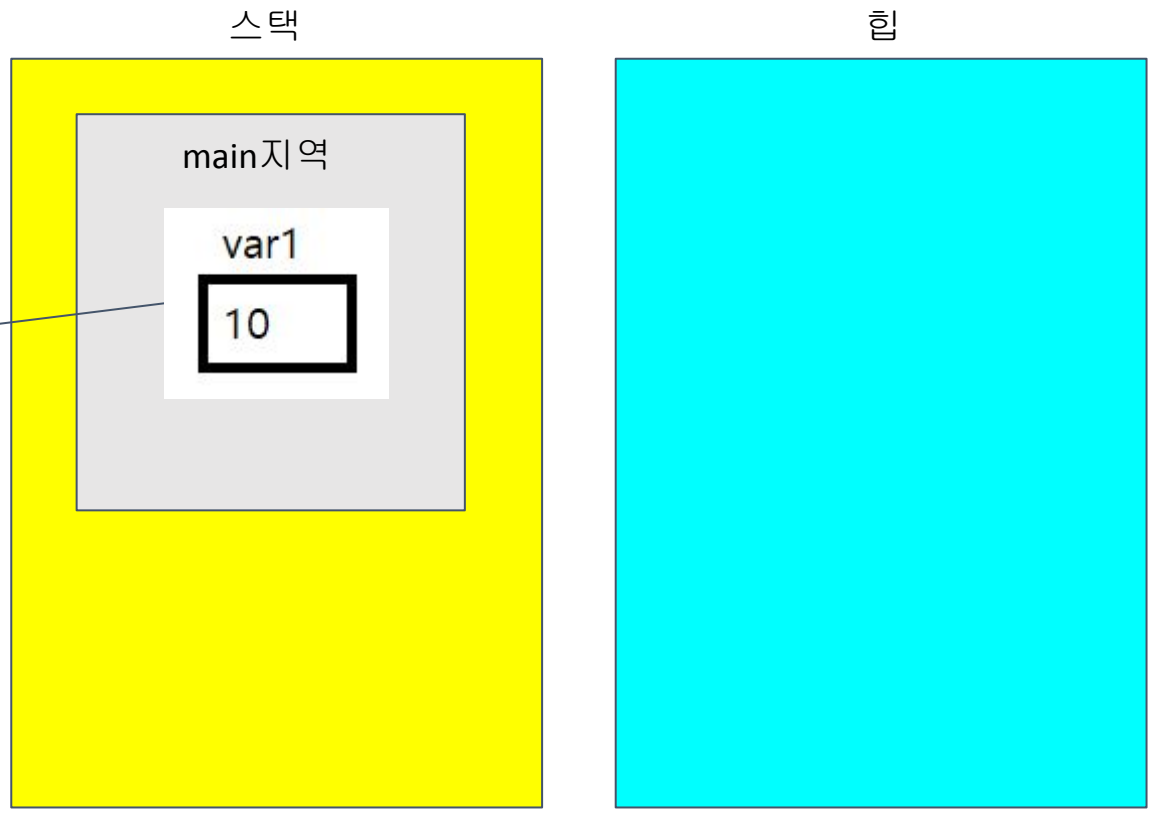

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



위 코드에서는 실행 타겟이 먼저 **main method**의 영역인 **5~10**번라인입니다.
역시 **5**번**~10**번 견적이 나오면 컴퓨터는 제일 먼저 스택에 **main**지역을 생성합니다.

변수를 해석하는 방법

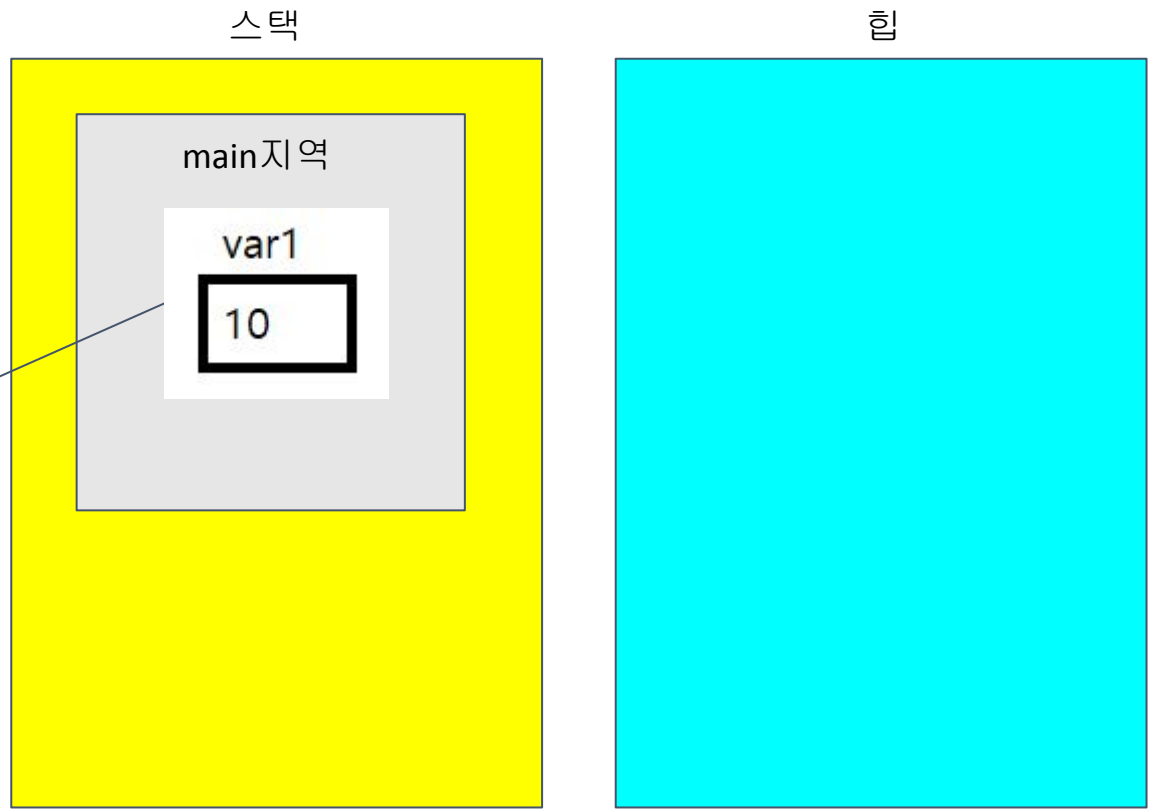

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



순서에 의해 5번 다음은 6번을 실행하는데 6번은 주석이니 넘기고
7번라인에 의해 var1 변수에 10을 저장합니다.

변수를 해석하는 방법


```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



다음 8번은 주석이니 역시 거르고 9번으로 넘어오게 되는데
앞으로 저런 구문이 보이면 제일 먼저 변수를 다시 원래 자료로 치환해야 합니다.
var1이라고 적힌 부분은 실제로는 10이라고 들어가 있다고 인식하셔야 합니다.

변수를 해석하는 방법


```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```

```
<terminated> Variable1 [Java Application] C:₩Program Files₩Eclipse Adoptiu
10
```

치환

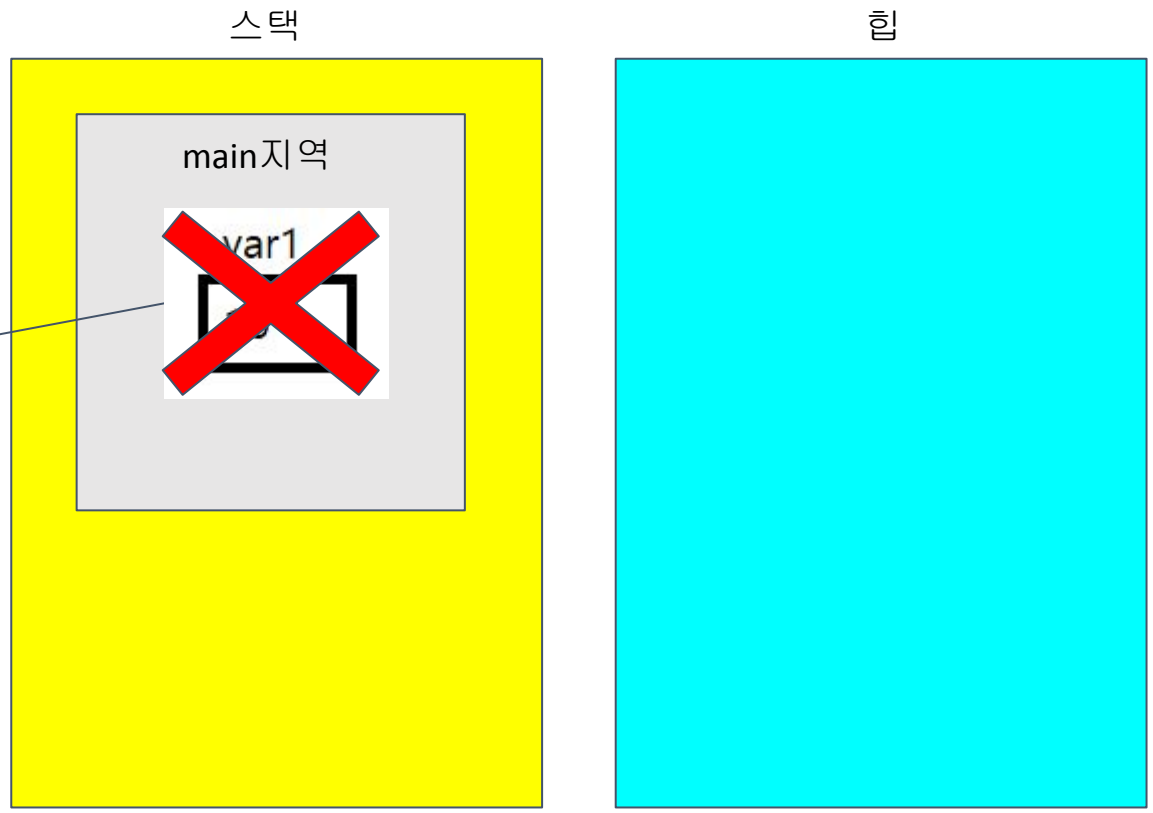
```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(10);
10     }
11
12 }
```

```
<terminated> Variable1 [Java Application] C:₩Program Files₩Eclipse Adoptiu
10
```

너무 당연하다고 생각하실지도 모르지만 이런 절차를 제대로 안 밟기 때문에
정수나 일반 자료에 대해서는 기가막히게 해석하시지만
메서드가 조금 나오거나 구문이 꼬이면 바로 포기해버리게 됩니다.

변수를 해석하는 방법

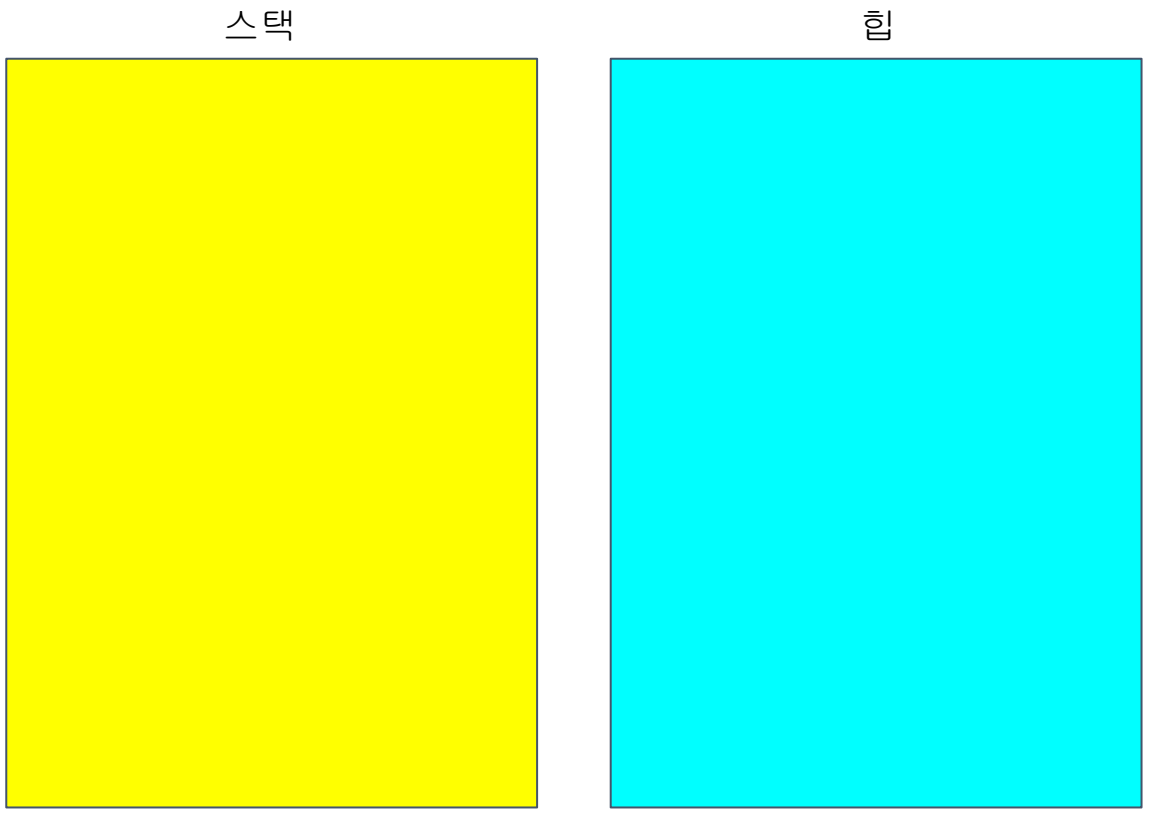

```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



이제 **10**번라인을 만나면 역시 스택에서 **main**지역이 소거됩니다.

변수를 해석하는 방법


```
3  public class Variable1 {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int var1 = 10;
8          // System.out.println(10); 과 동일함
9          System.out.println(var1);
10     }
11
12 }
```



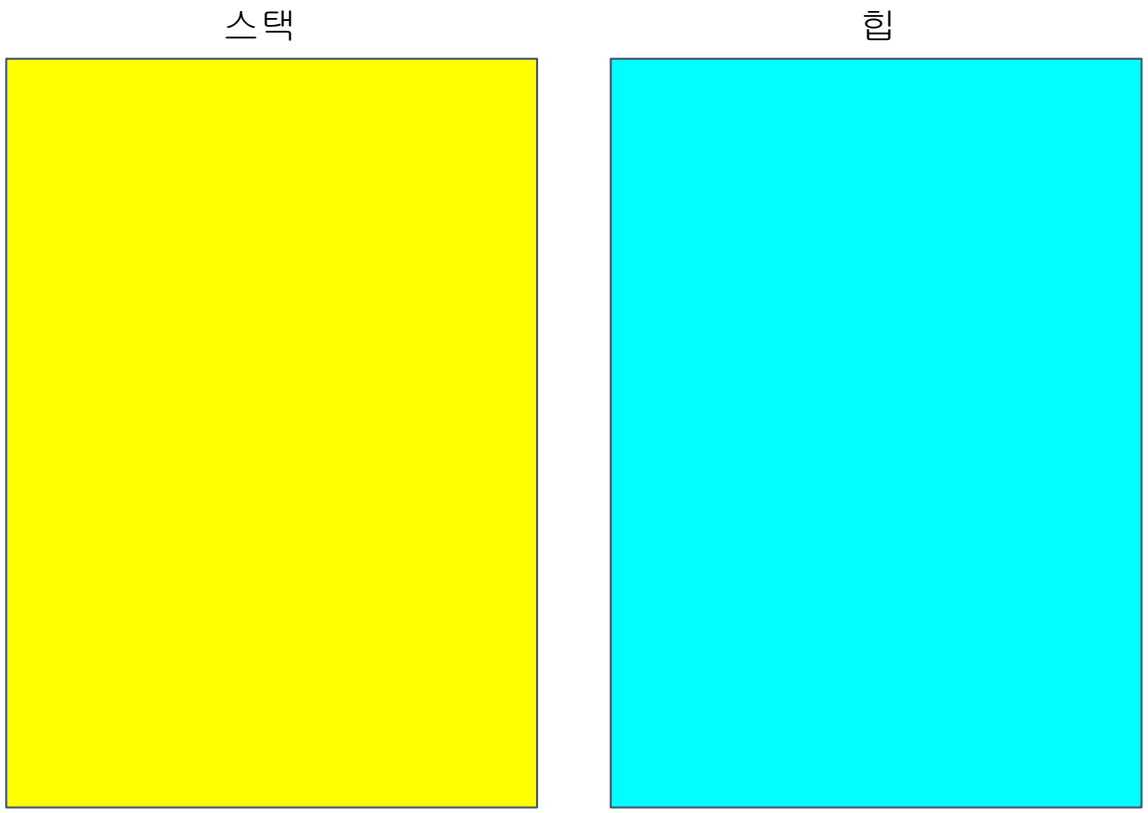
그리고 스택이 완전히 비워지고 프로그램이 종료됩니다.

변수를 해석하는 방법


```java
public class Variable1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int var1 = 10;
        // System.out.println(10); 과 동일함
        System.out.println(10);
        // int var2 = 10; 과 동일함
        int var2 = var1;
    }

}
```



문제 : 위 코드를 보고 메모리에 지역과 변수가 생성되었다가 소멸되는 과정

그리고 그 과정에서 변수 **var1**을 저장값으로 치환해서 코드를 해석해 보세요.

변수를 해석하는 방법

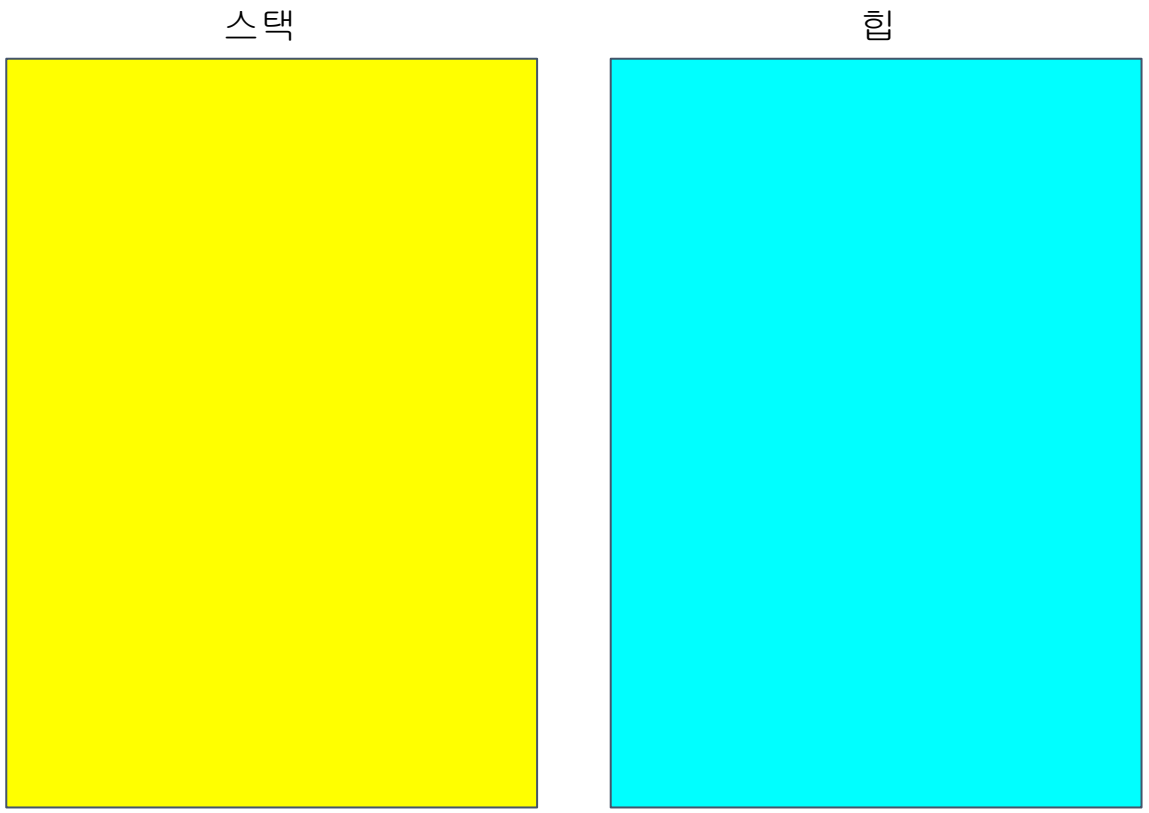
이렇듯 자료 자체가 그냥 저장되는 기본형 자료는

그냥 눈에 보이는대로 해석하면 끝나기 때문에 별로 어렵지 않습니다.

다음으로 배열을 이용해서 참조형 자료의 해석에 대해 보겠습니다.

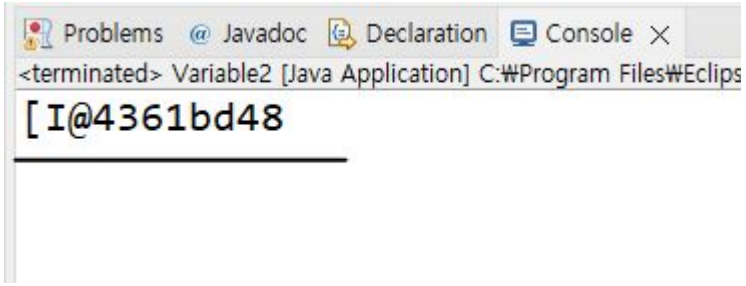
변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8
9      }
10
11 }
```



이번엔 대표적으로 해석을 혼란스럽게 하는 참조형 변수에 대해 보겠습니다.
배열같은경우 위와같이 코드를 두고 실행하면 결과값이 뭐라고 나올까요?

변수를 해석하는 방법



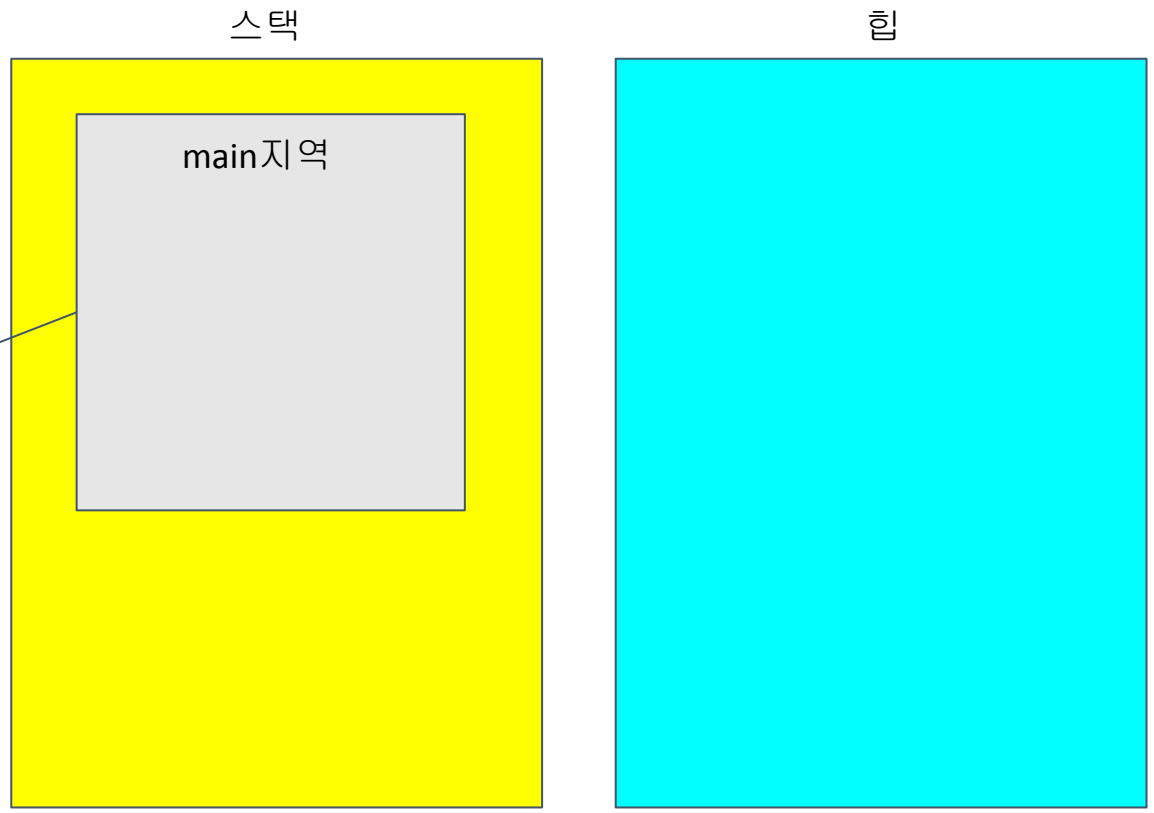
[1, 2, 3, 4, 5] 와 같이 내부에 저장한 데이터가 그대로 나올것이라고 생각했다면

메모리에 대한 이해가 부족한 상태이므로

기본형 변수, 참조형 변수 라는 키워드로 먼저 공부를 하고 오셔야 합니다.

변수를 해석하는 방법

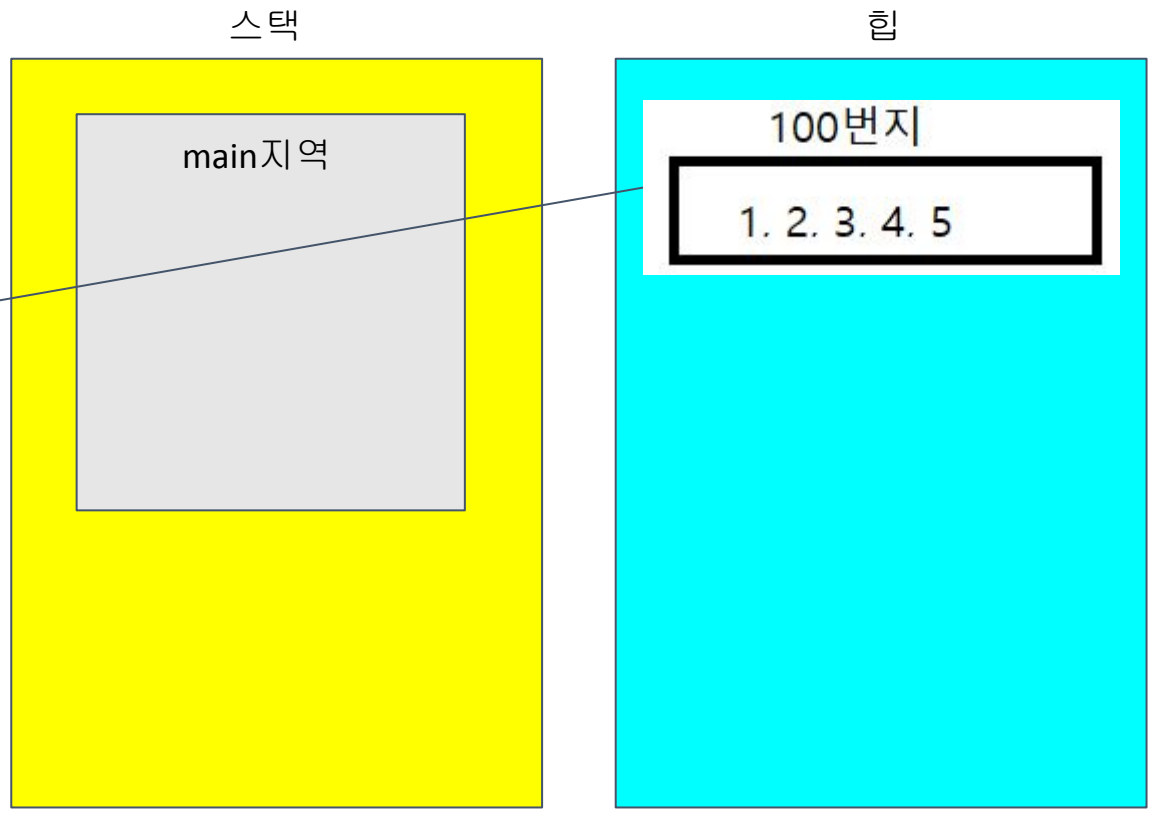

```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8
9      }
10
11 }
```



이제 해당 코드의 타겟 실행범위인 **main method** 영역 **5~9**라인을 보겠습니다.
역시 **main**영역이 생깁니다.

변수를 해석하는 방법

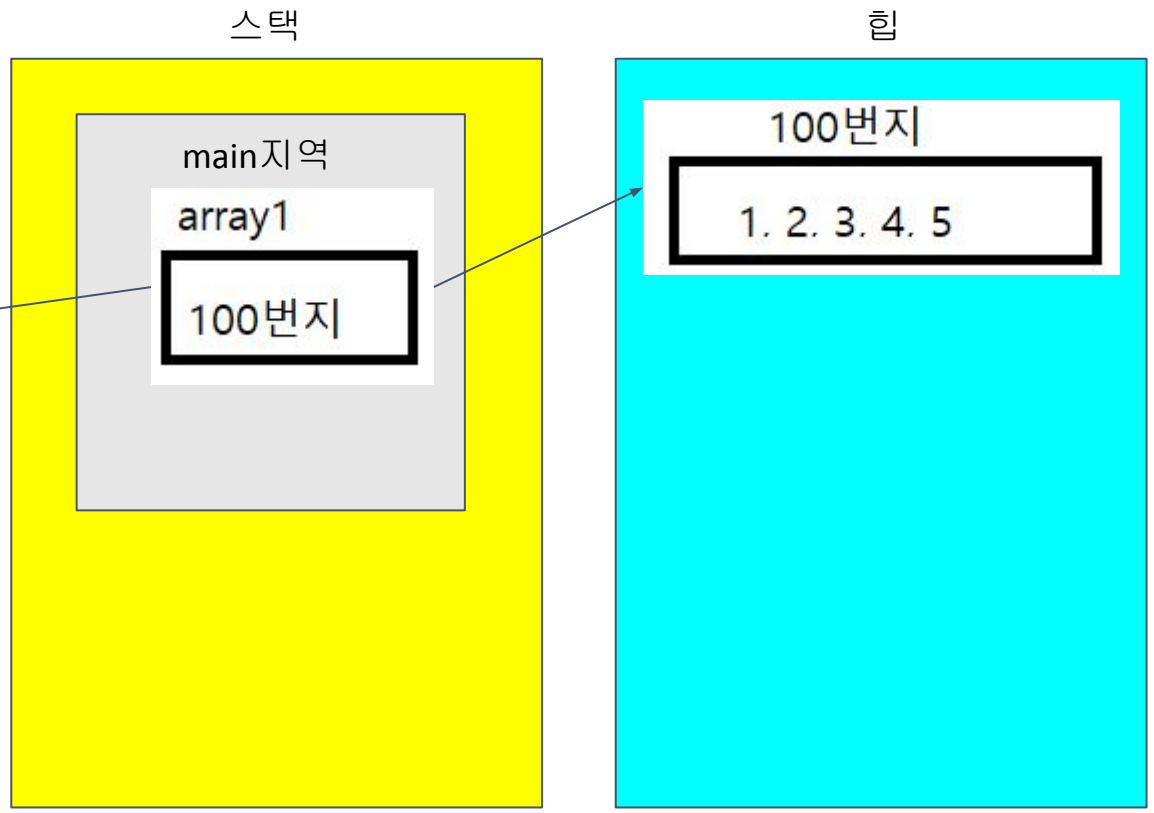

```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8
9      }
10
11 }
```



여기서, 명목상으로는 **array1** 변수에는 여러개의 값**(1, 2, 3, 4, 5)**이 저장되지만
스택 내부에서는 **기본적으로 하나의 값만 저장**할 수 있습니다.
따라서 여러 데이터를 저장하기 위해 **실제로는 힙에 1, 2, 3, 4, 5를 저장**합니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8
9      }
10
11 }
```



주소 100번지는 실제로는 랜덤하게 붙는 주소이며
스택의 main지역 array1 변수에는 “100번지”라는 주소만 저장하면
값을 하나만 저장하고도 5개를 저장한 것과 같은 효과가 생깁니다.

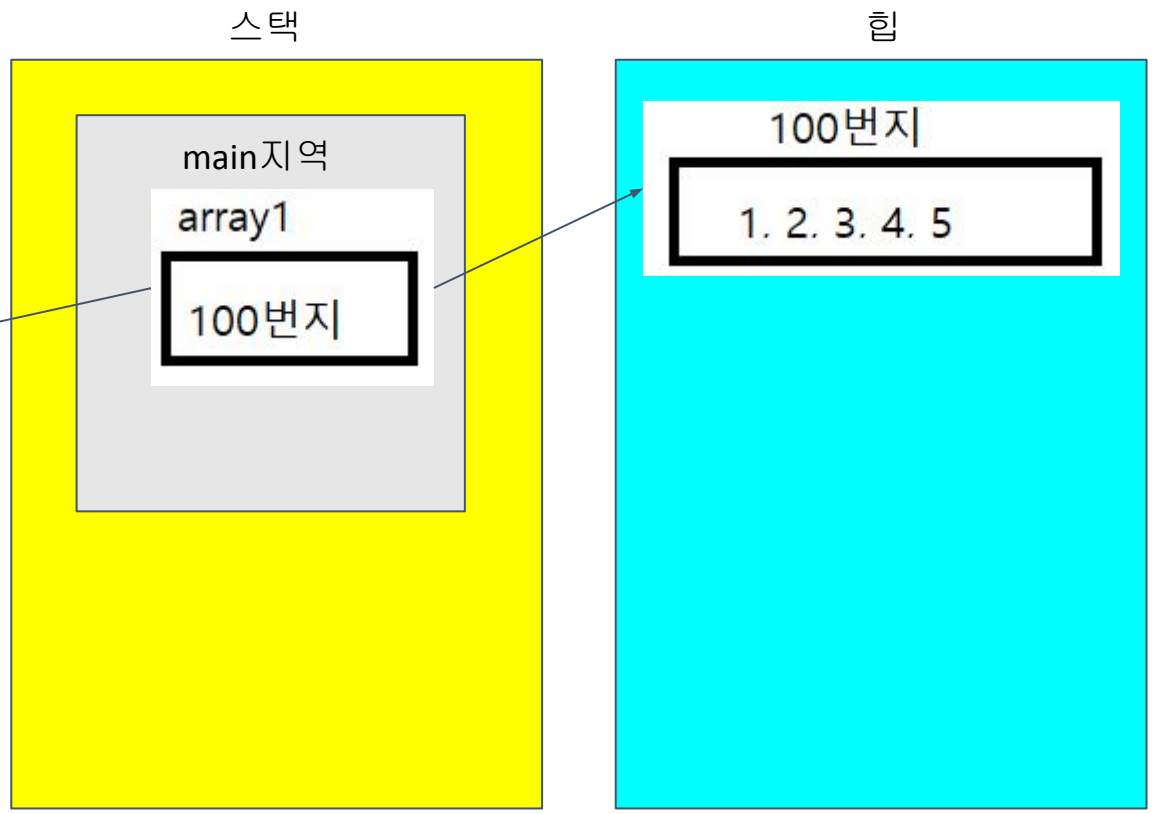
변수를 해석하는 방법

```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8
9      }
10
11 }
```

Problems @ Javadoc Declaration Console

<terminated> Variable2 [Java Application] C:₩Program Files₩Eclips

[I@4361bd48



이제 여기서 **System.out.println();** 구문을 이용해 **array1** 내부 자료 조회시
당연히 **1, 2, 3, 4, 5**를 직접 저장하고 있지 않기 때문에
100번지(실제로는 콘솔에 뜨는 무작위의 16진수 주소)라고 나오는겁니다.

변수를 해석하는 방법

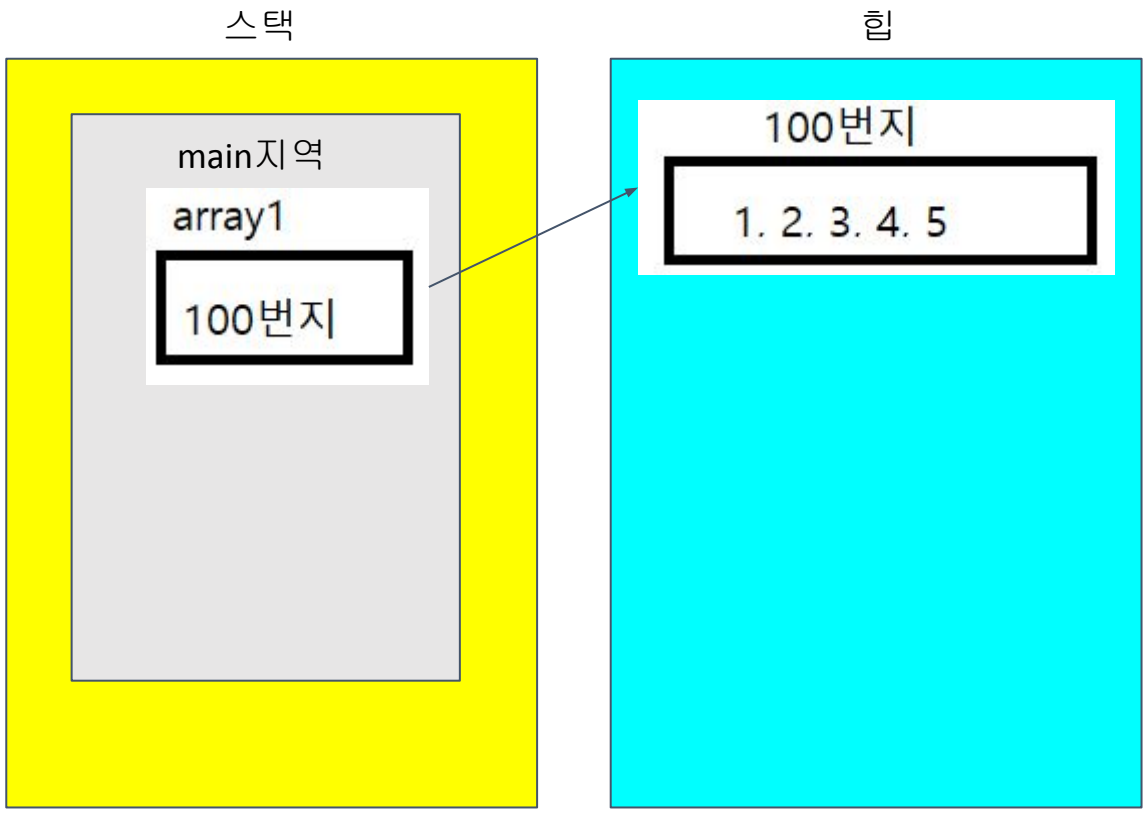
우리가 **var1**에 들어있던 값을 **var2**에 저장해줄 때

실제로는 **var2**에 **var1**에 저장된 값인 **10**을 저장하는 효과가 있었듯

array2를 선언해 **array1**을 대입받아 저장해도 같은 식으로 저장됩니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```



다음 단계로 대입에 따른 파급효과를 한 줄 한 줄 보겠습니다.

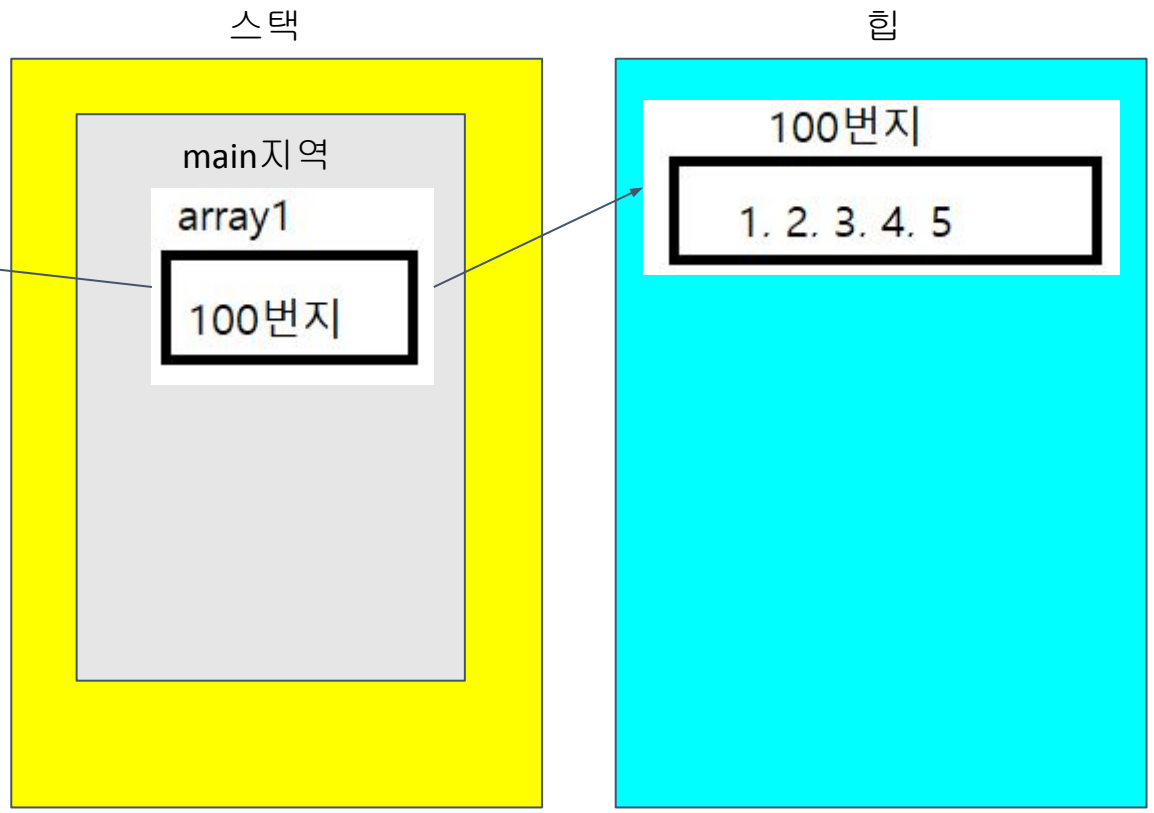
코드를 위와같이 **8~11**번 라인을 추가해주세요.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

```
int[] array2 = 100번지;
```

변수는 무조건 변수에 든 값으로 치환한다!



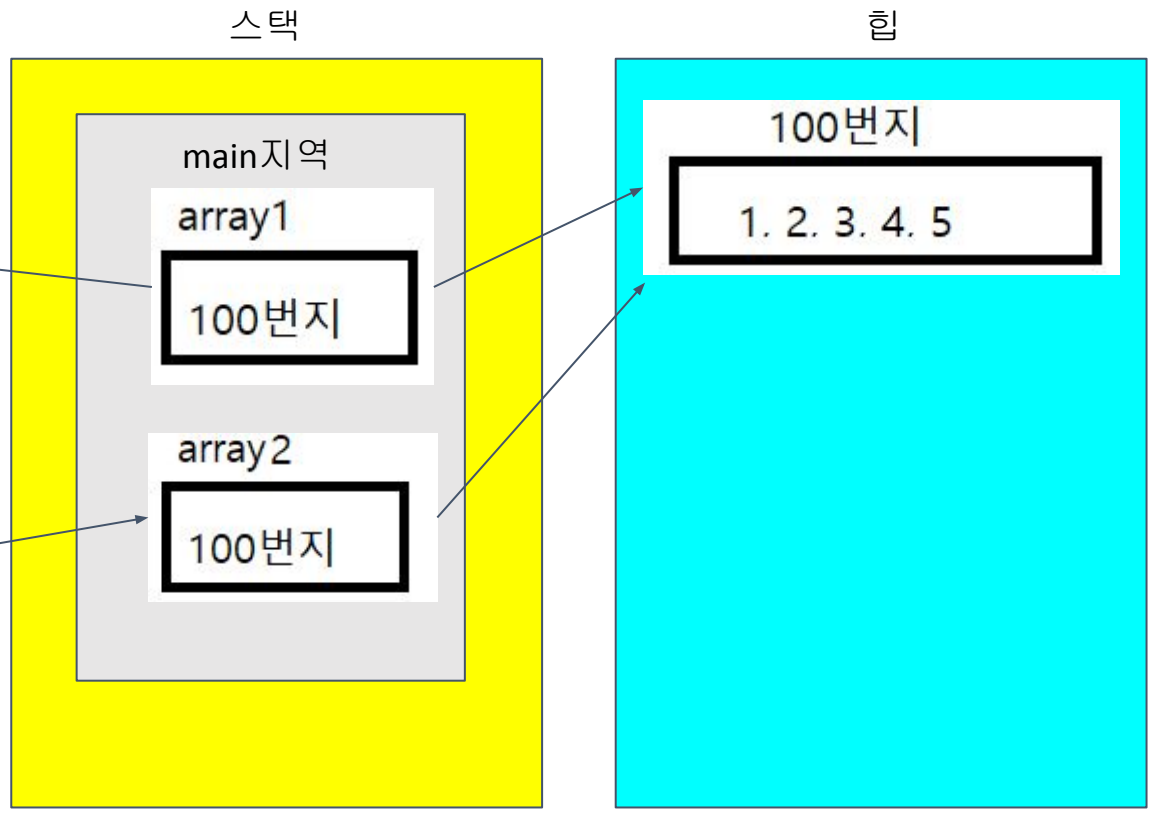
이제 8번부터 11번까지 순차적으로 보면 아까 말씀드렸듯
8번라인은 우측 array1이 들어오는 자리에
실제로는 저장된 값으로 치환을 해야 하므로
100번지를 array2에 대입받고 있다는 점을 인지하는것이 중요합니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

```
int[] array2 = 100번지;
```

변수는 무조건 변수에 든 값으로 치환한다!



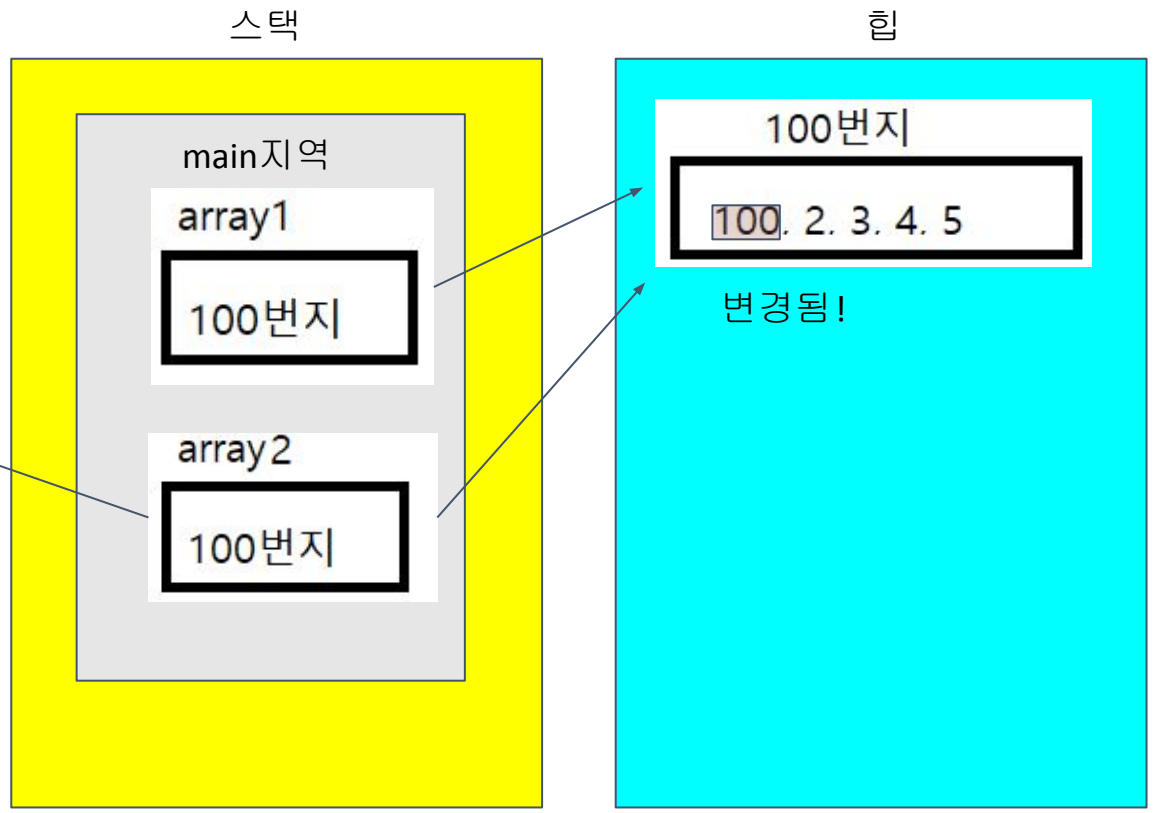
그래서 단순하게 **array1**을 **array2**에 대입해줬네?
가 아닌 위의 사고과정을 거쳐서
array1에 저장되어있던 주소를 **array2**에 저장해줬고
그 결과 두 개의 변수가 같은 주소의 값을 조회하게 되었다고 이해해야 합니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

100번지[0] = 100;

변수는 무조건 변수에 든 값으로 치환한다!



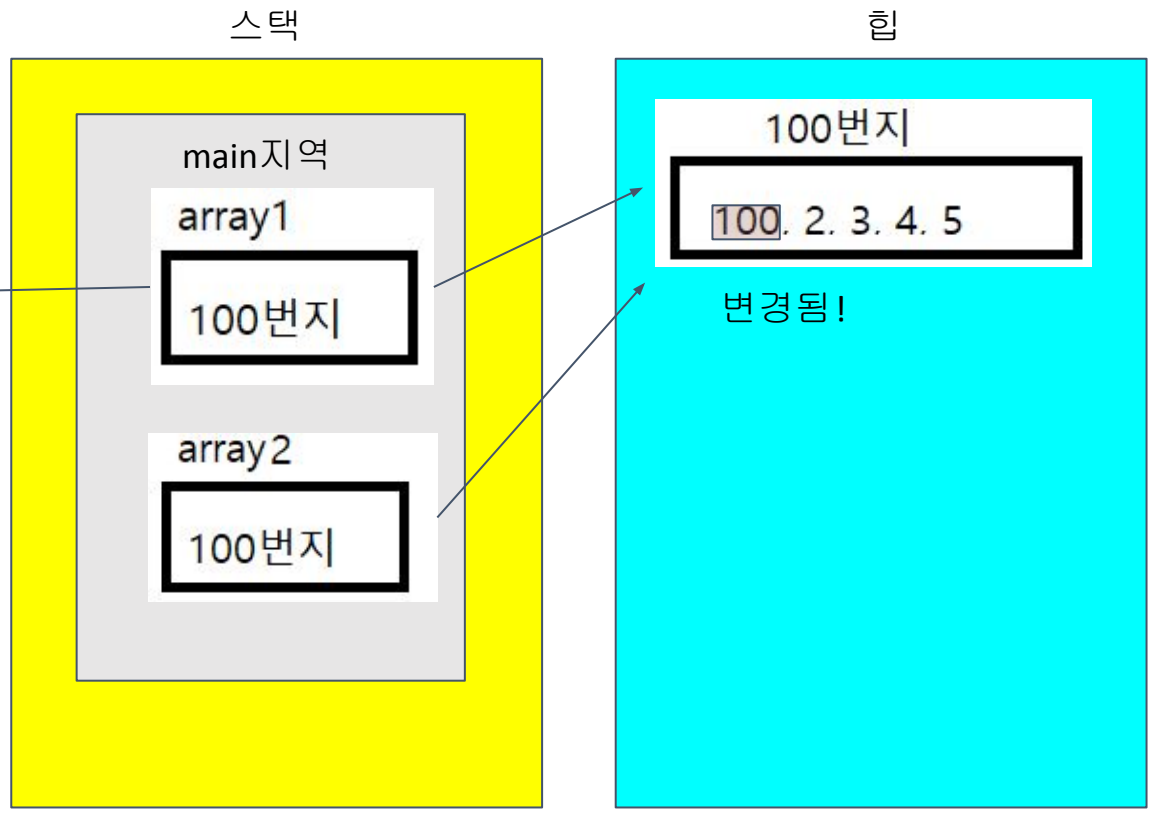
이제 9번 라인을 해석할때도, 이미 main지역에 array2는 8번라인에 의해서 100번지라는 값이 저장되어 있기 때문에, 실제로는 array2[0]은 100번지의 0번 인덱스로 생각하고 처리해야 합니다.
100번지에는 1, 2, 3, 4, 5가 저장되어 있었지만 0번째를 100으로 교체합니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

```
10         System.out.println(100번지[0]);
```

변수는 무조건 변수에 든 값으로 치환한다!



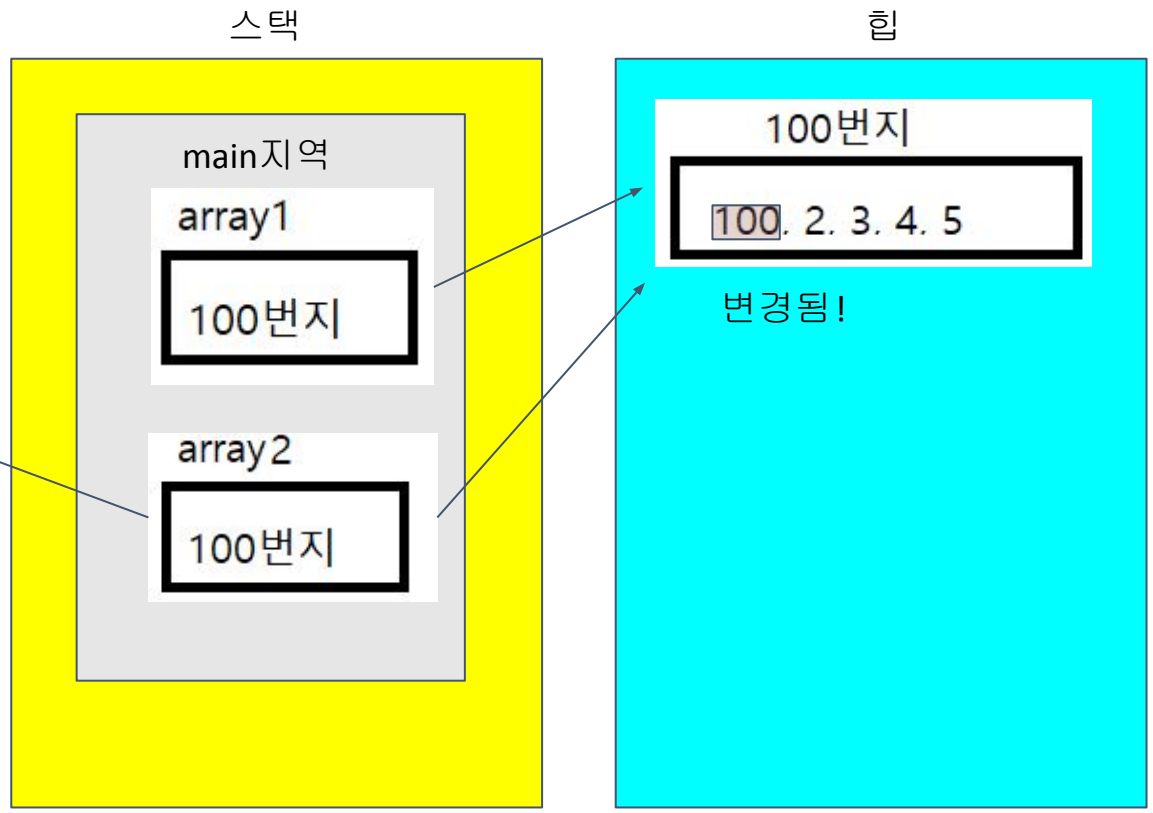
같은 맥락으로 array1[0]을 보는 순간 여러분은 스택의 main지역에 array1이 저장되었는지 먼저 확인해야 하고, 100번지라는 값을 저장하고 있으니 array1[0]은 100번지[0]으로 생각해야 합니다.
100번지의 0번째는 그리고 9번라인에 의해서 100으로 변경되었습니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

```
11         System.out.println(100번지[0]);
```

변수는 무조건 변수에 든 값으로 치환한다!



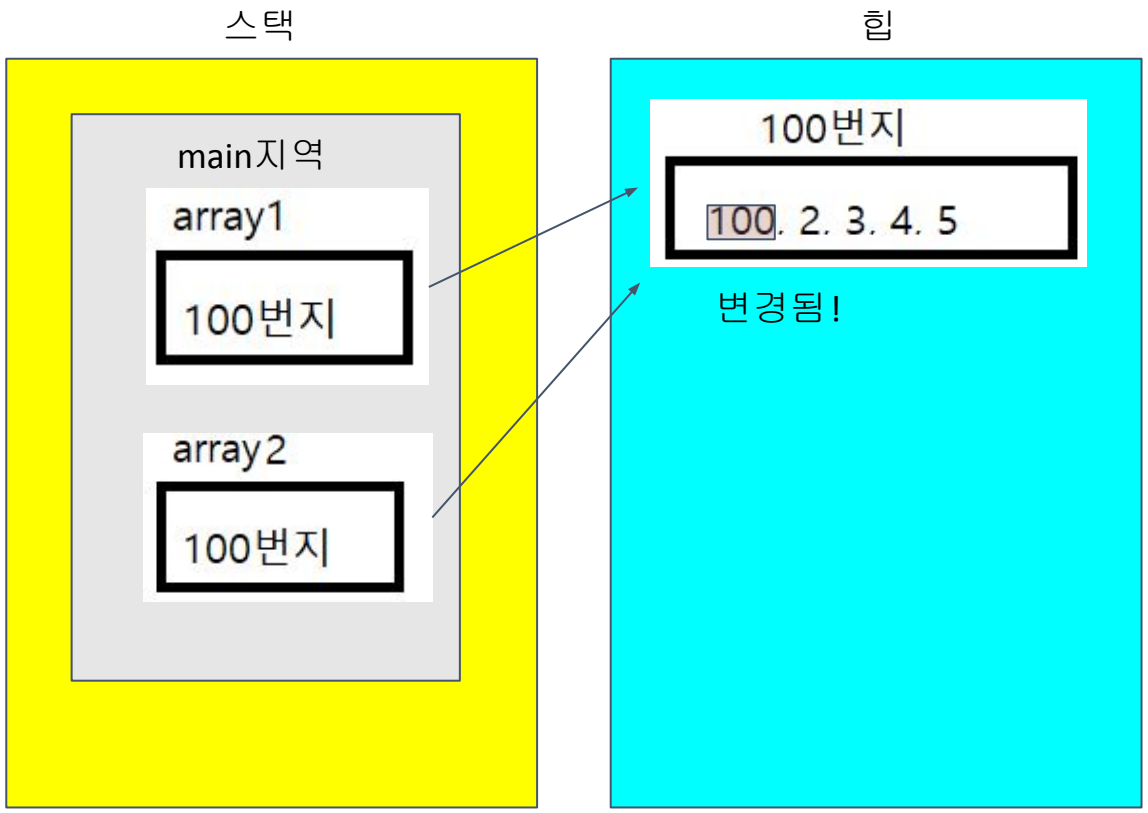
마찬가지로 array2도 array1과 마찬가지로 실제 저장하고 있는값은 100번지이니

100번지의 0번째 요소인 100을 조회하는것입니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```

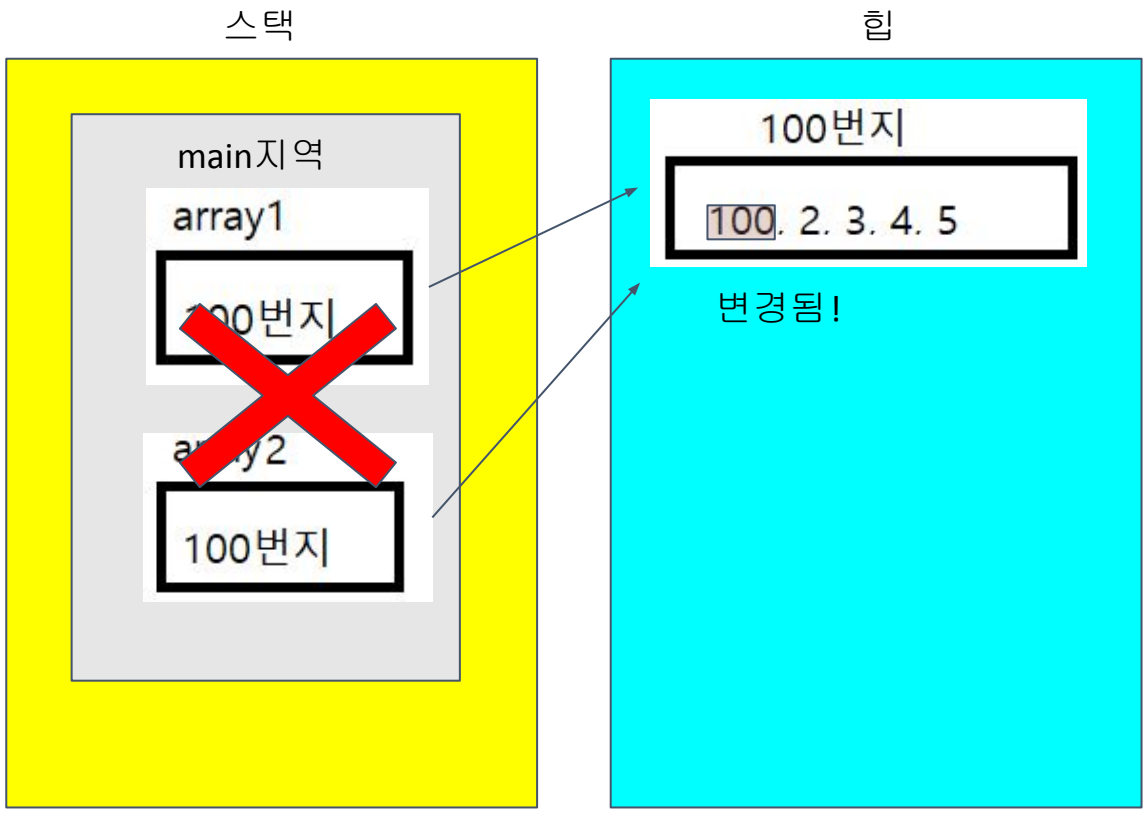
```
[I@4361bd48
100
100
```



따라서 결과는 위와 같이 **array2[0]** 만 교환한것처럼 보이지만
100번지를 저장중인 모든 요소가 영향을 받아
array1[0] 역시 **100**으로 나오는 것입니다.

변수를 해석하는 방법

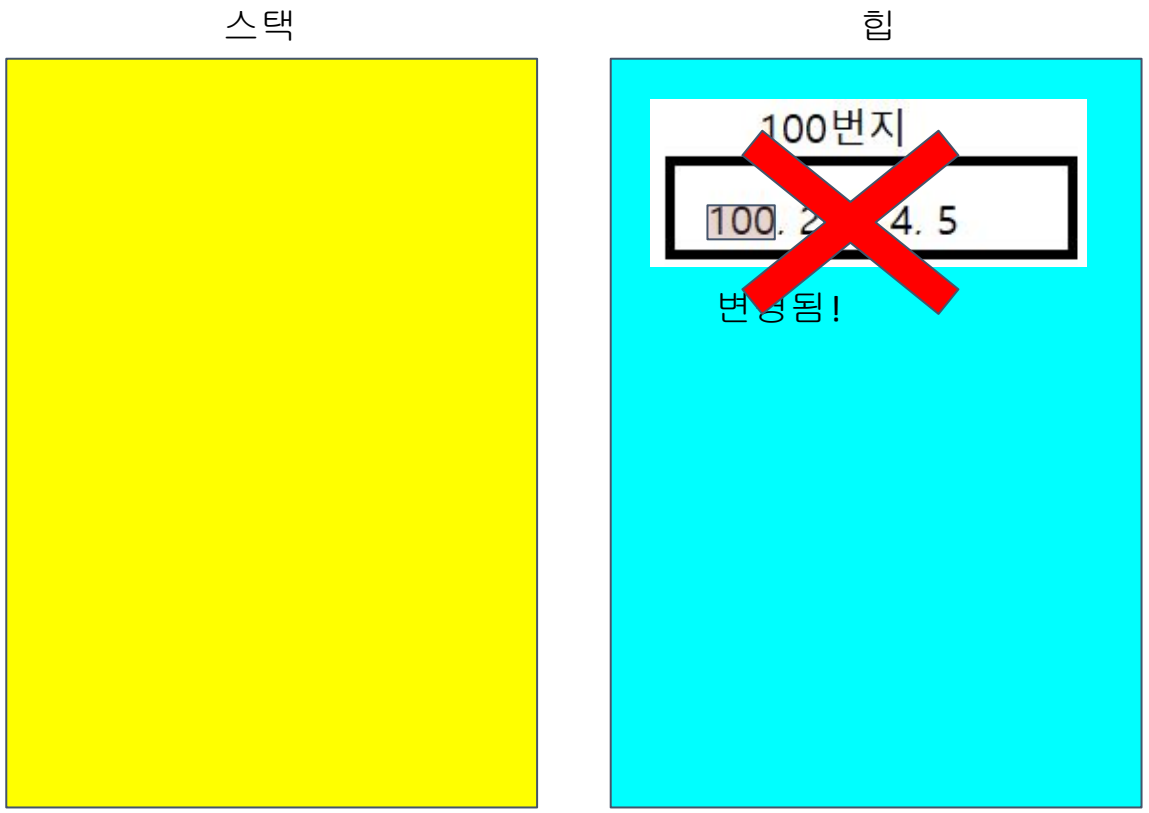

```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```



마지막으로 12번 라인에 도달하면 main 영역이 삭제됩니다.

변수를 해석하는 방법


```
3  public class Variable2 {
4
5      public static void main(String[] args) {
6          int[] array1 = {1, 2, 3, 4, 5};
7          System.out.println(array1);
8          int[] array2 = array1;
9          array2[0] = 100;
10         System.out.println(array1[0]);
11         System.out.println(array2[0]);
12     }
13
14 }
```



그 후 스택에 있던 모든 지역이 사라지면 마지막으로 힙에 있던 데이터도 삭제되고 프로그램이 끝납니다.

변수를 해석하는 방법

이번 교안은 여기까지입니다.

마지막으로 여러분들이 공부하는데 썼던 코드 중 배열관련 코드를 하나 열어서

지금처럼 절차에 따라 순서대로 그림을 그려주시고,

그 와중에 변수는 변수에 든 값으로 치환하여 확인하는 연습을 꼼꼼히 해주세요.

논리적으로 설명할 수 있는 순간부터 코드를 한층 더 정확하게 이해할 수 있습니다.