호출 스택

스택 vs 큐

- **스택(Stack)**

  쌓아 올리는 형태를 가진 자료구조이며 후입선출(LIFO, Last In First Out)의 구조

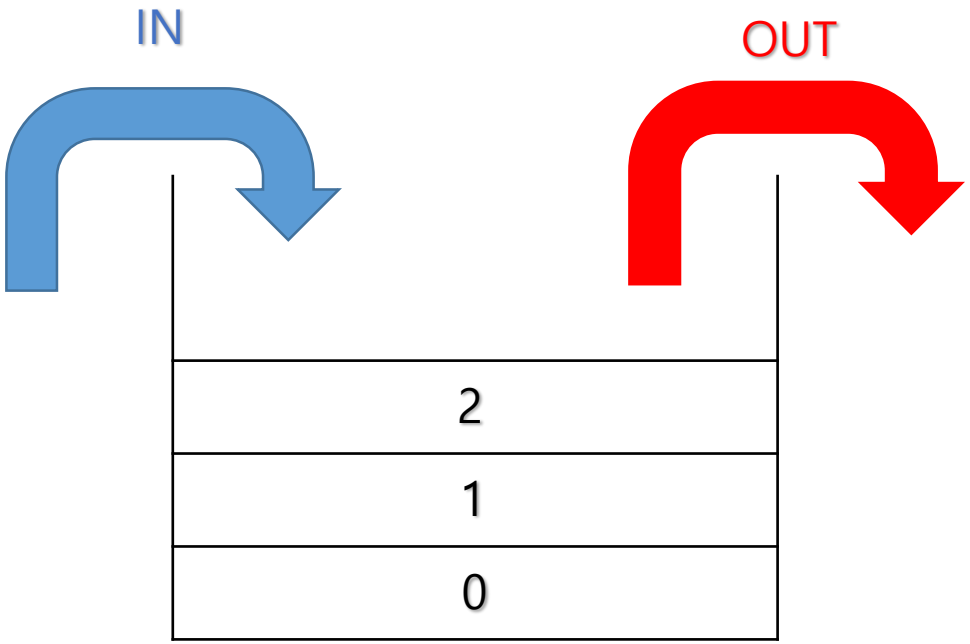
- **큐(Queue)**

  줄을 서서 기다리는 형태를 가진 자료구조이며 선입선출(FIFO, First In First Out)의 구조

스택 vs 큐

- **스택(Stack)**

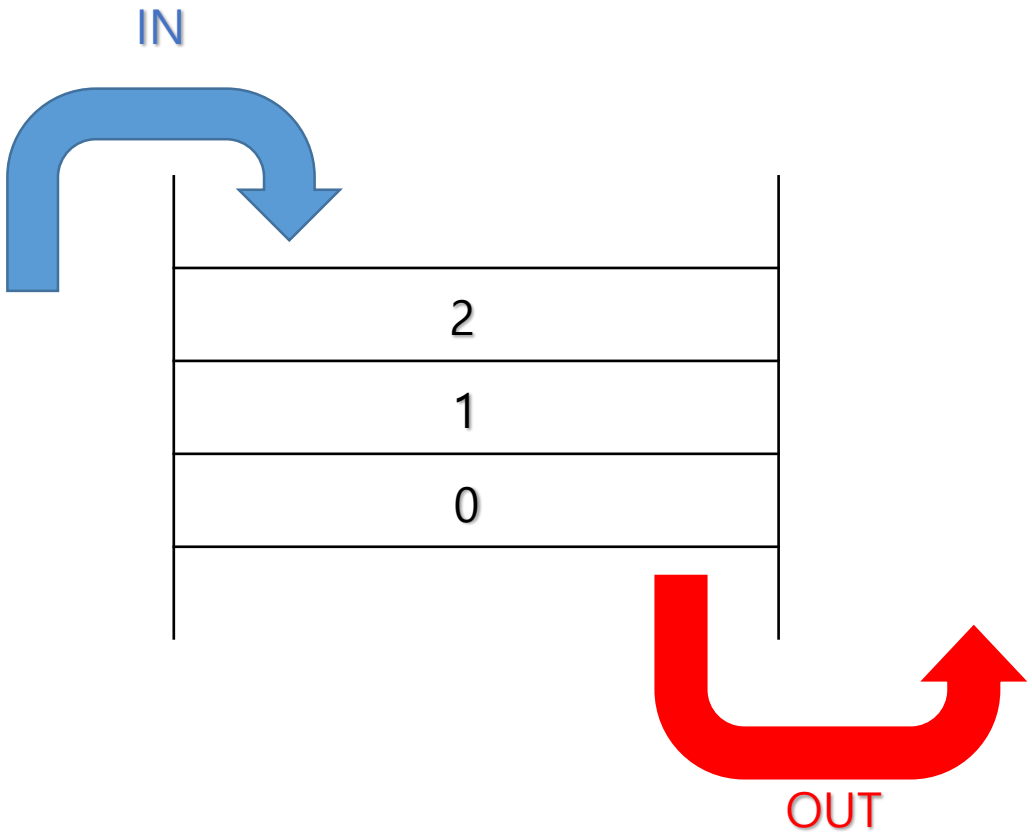
1. 데이터 0을 꺼내기 위해서는 데이터 1, 2를 모두 꺼내야 함
2. 데이터 1을 꺼내기 위해서는 데이터 2를 꺼내야 함



스택 vs 큐

- **큐(Queue)**

1. 데이터 0을 바로 꺼낼 수 있음
2. 데이터 1을 꺼내기 위해서는 데이터 0을 먼저 꺼내야 함



호출 스택 - 1

- **호출 스택(Call Stack)**

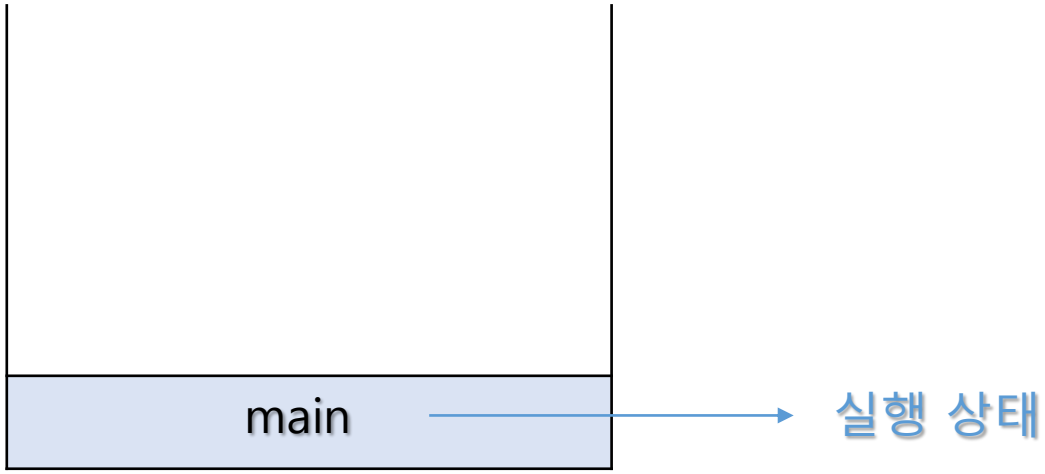
  메소드 호출과 관련된 정보를 저장하는 메모리 영역으로 후입선출(LIFO) 구조를 가짐

호출 스택 - 2

- 설명

main 메소드가 호출되어 스택에 올라가있는 상태

```
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
```



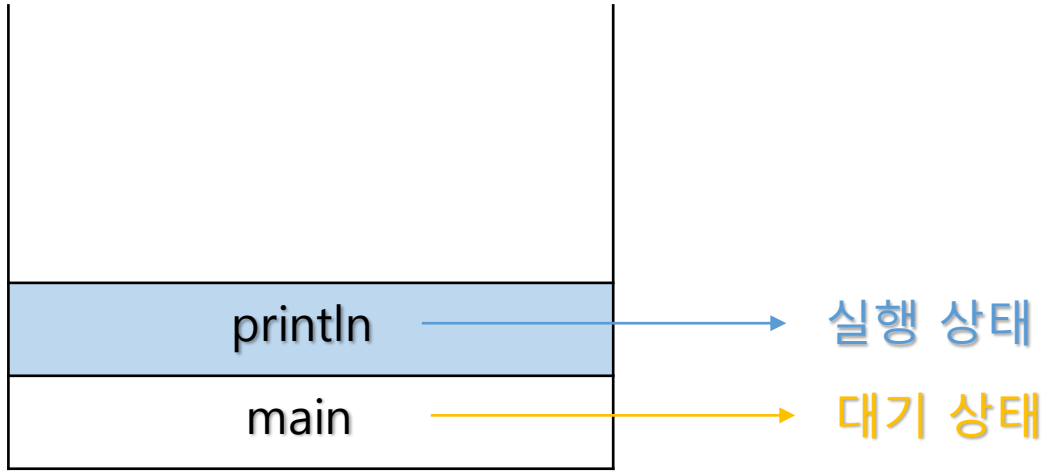
호출 스택 - 3

- 설명

main 메소드가 println 메소드를 호출함

1. main 메소드는 대기 상태로 변경
2. println 메소드는 실행 상태로 변경

```
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
```

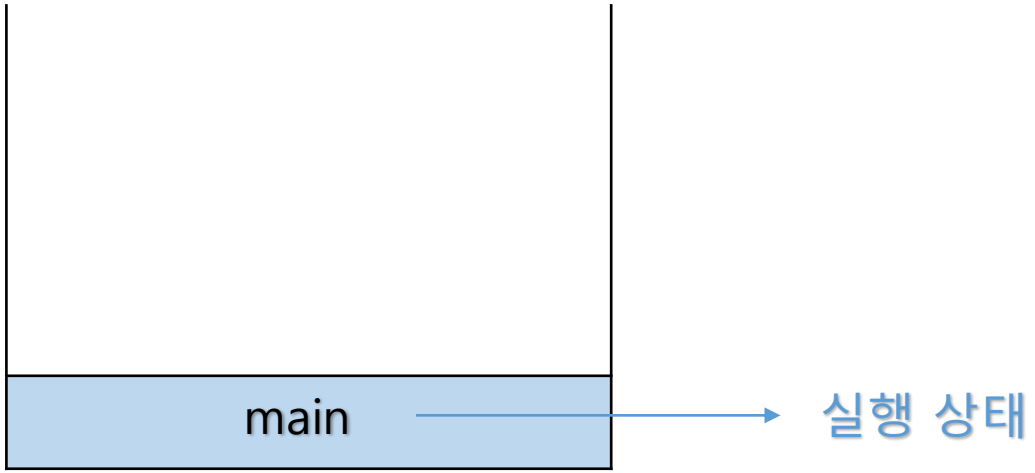


호출 스택 - 4

- 설명

println 메소드가 종료되며 main 메소드가 다시 실행 상태로 변경

```
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
```



호출 스택 - 5

- 설명

main 메소드가 종료되며 스택이 완전히 비워지며 메모리 영역에서 소멸함

기본형 매개변수

기본형 vs 참조형

- ## 기본형 vs 참조형

어떠한 값을 복사하여 매개변수에 넣느냐에 따라 값을 수정할 수 있느냐, 없느냐가 결정됨
* 교재 5.객체 19슬라이드에 언급 되었었음 (메소드 매개변수 주의사항)

이를 이해하기 위해서는 변수가 메모리를 사용하는 흐름과 호출스택에 대해 이해해야함

- ## 기본형 매개변수

메소드 매개변수에 전달을 할 때 값을 전달함

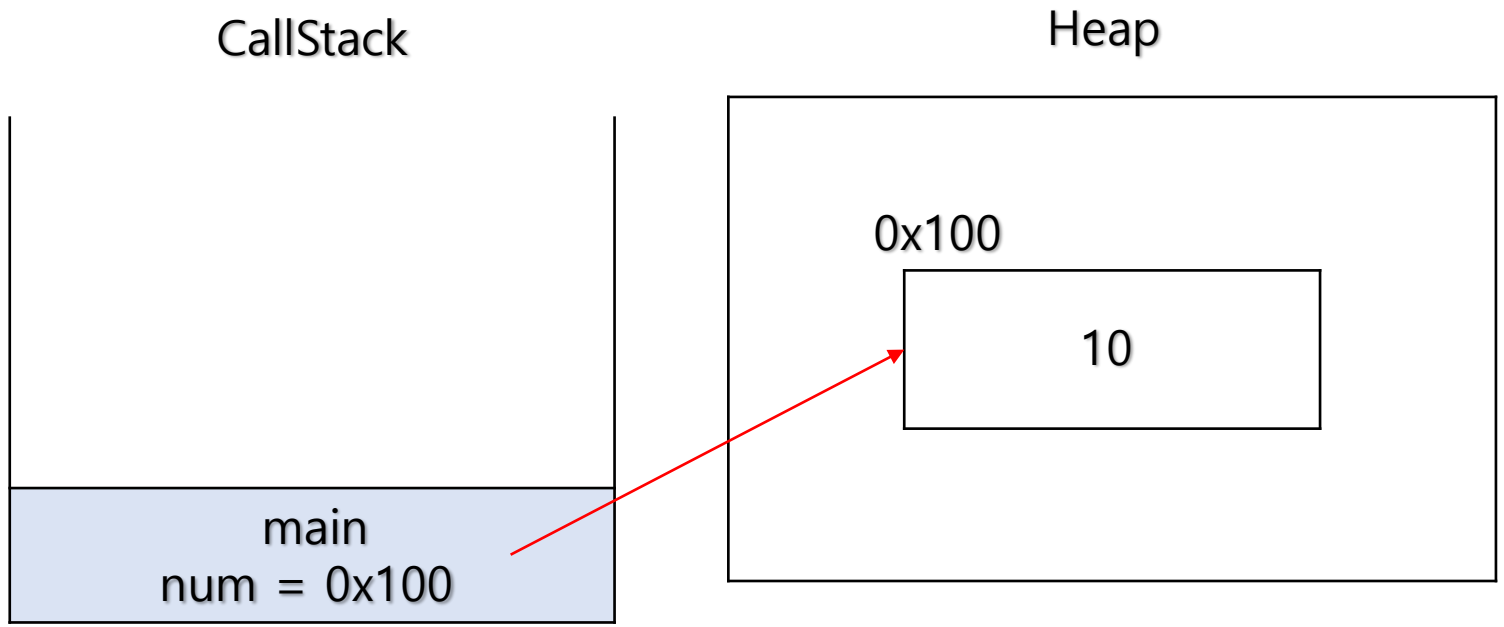
기본형 매개변수

- 설명

main 메소드가 호출되어 스택에 올라가있는 상태
main 메소드는 num 변수의 위치를 참조

* CallStack cs = new CallStack(); → CallStack 타입(참조형)의 객체를 생성

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



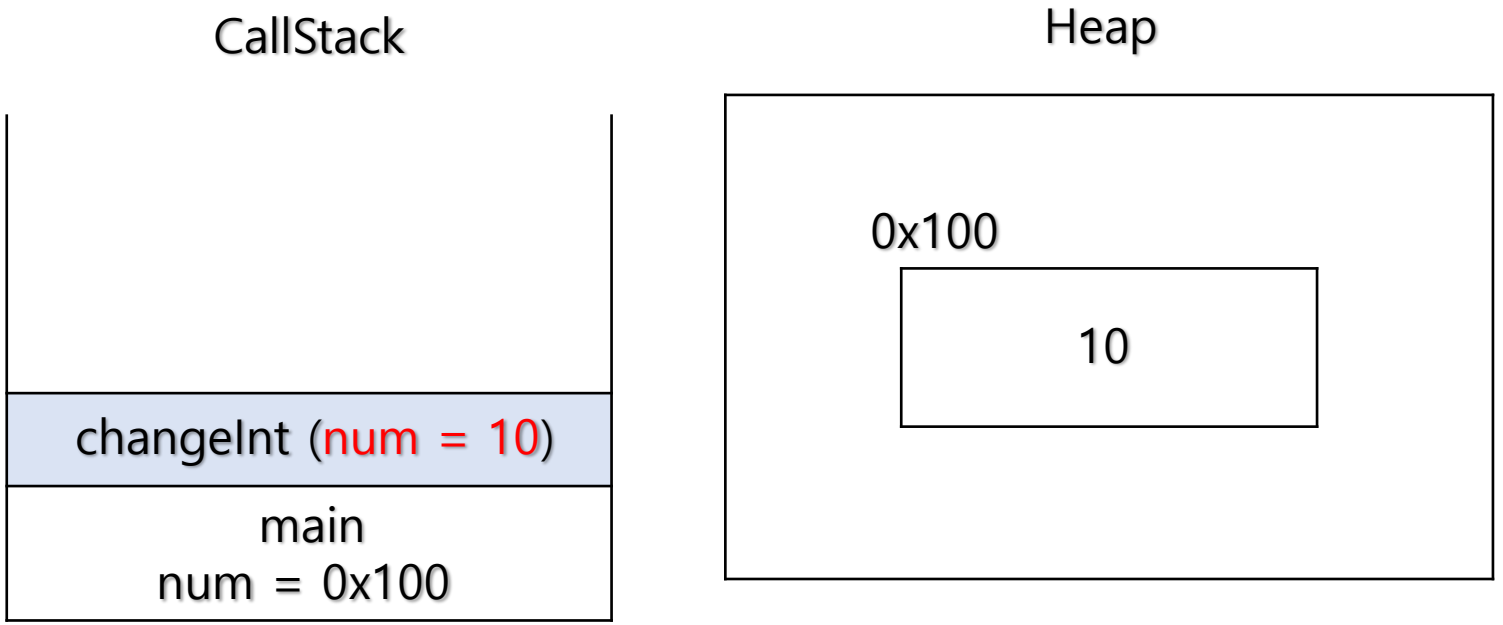
기본형 매개변수

- 설명

changeInt 메소드가 실행되며 값을 전달 (매개변수의 타입이 기본형 int이기 때문)

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

10

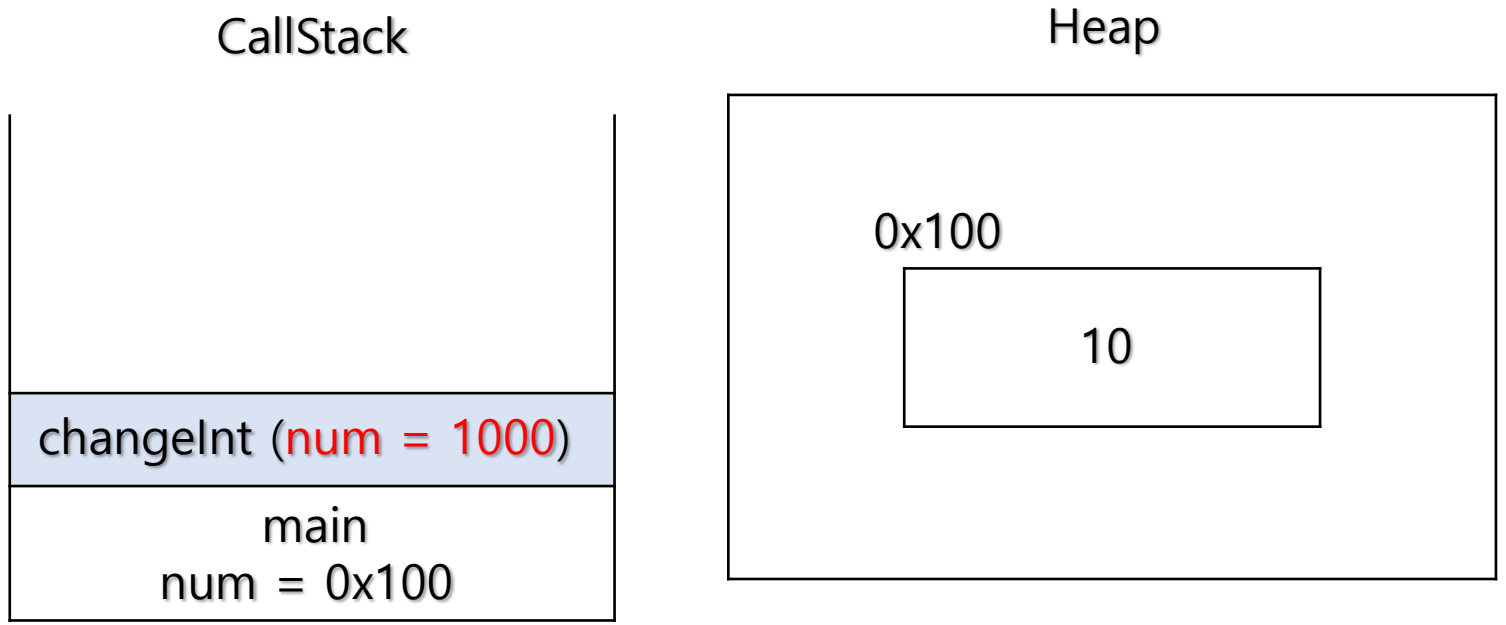


기본형 매개변수

- 설명

전달받은 값을 1000으로 변경

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

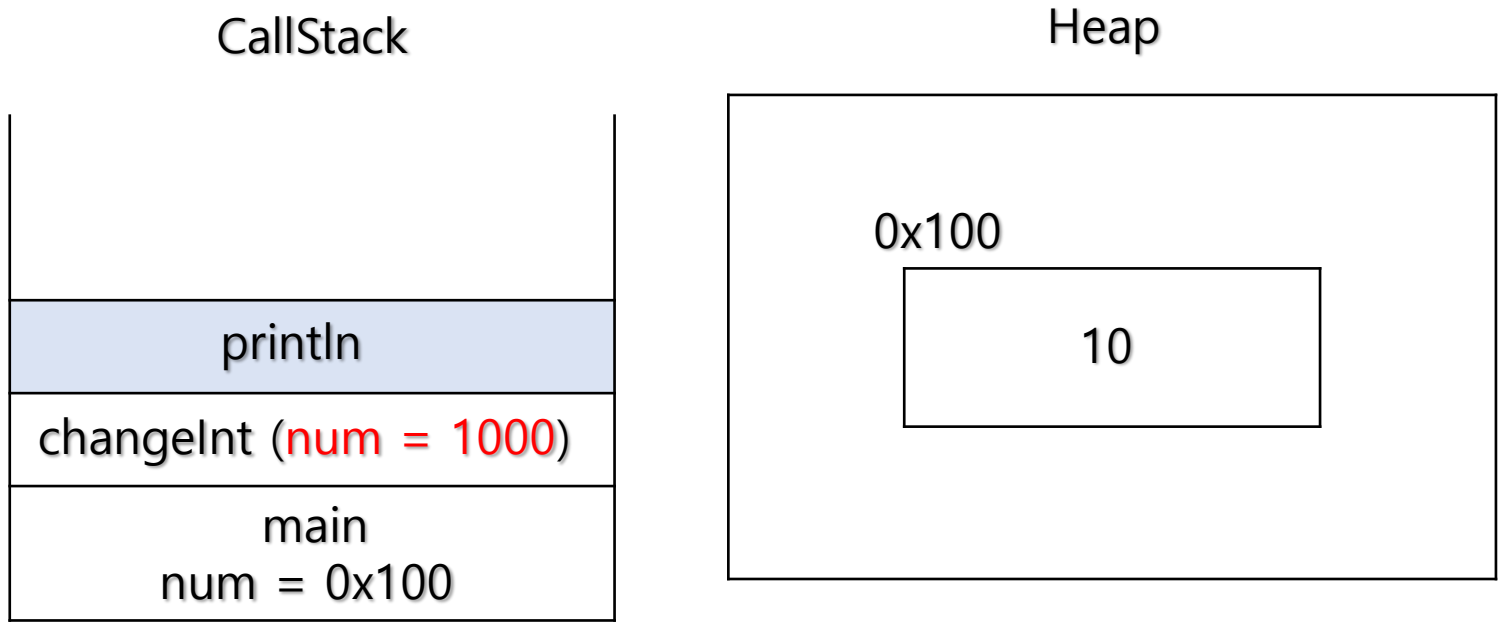


기본형 매개변수

- 설명

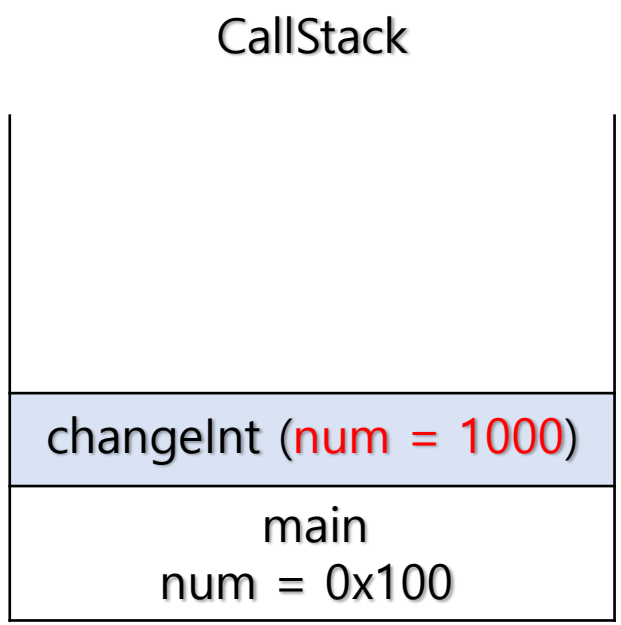
println 메소드가 실행되며 num의 값(1000)을 출력

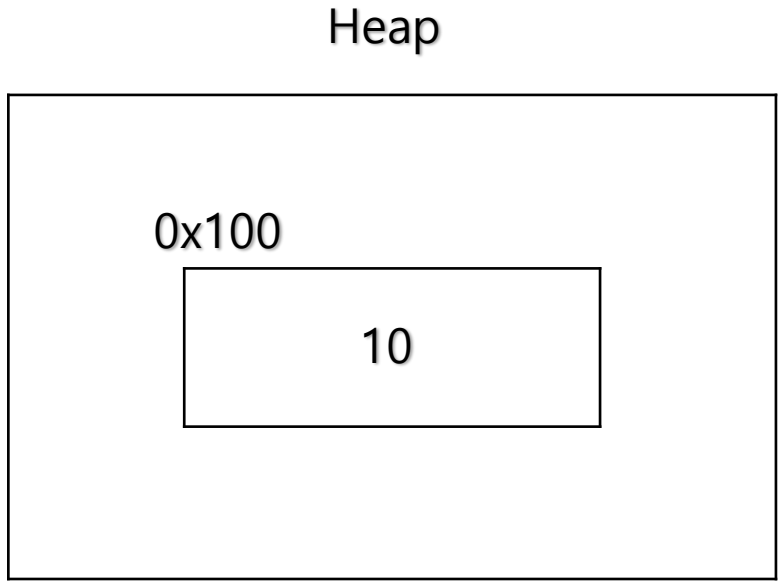
```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



기본형 매개변수

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



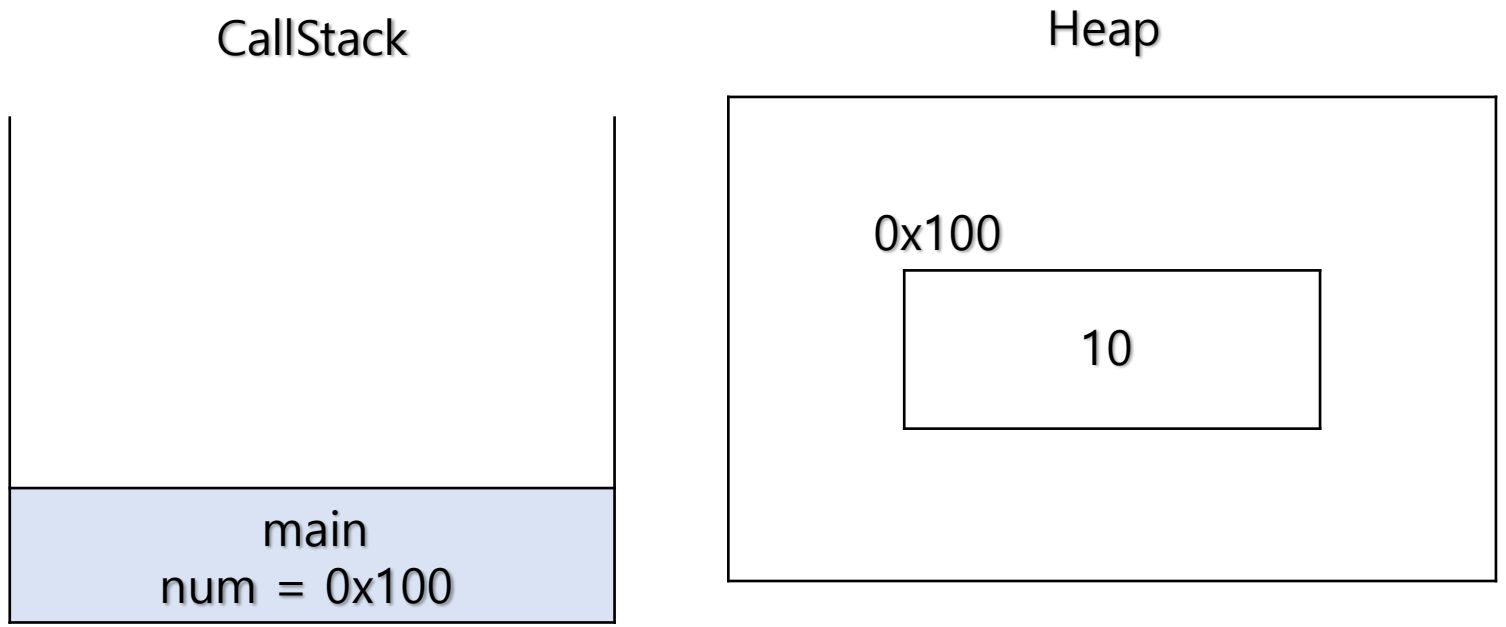


기본형 매개변수

- 설명

changeInt 메소드가 종료되며 main 메소드로 돌아옴

```
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

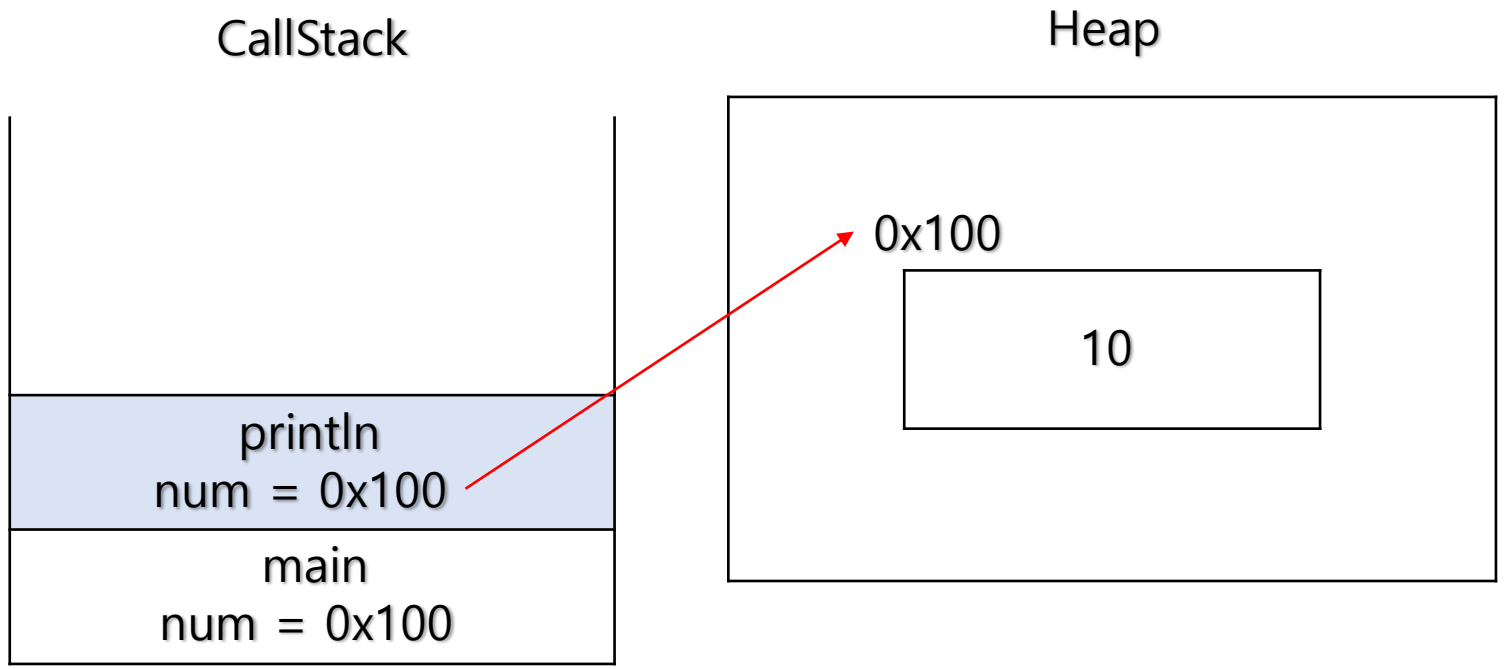


기본형 매개변수

- 설명

main 메소드에 있는 num의 주소를 참조하여 확인 하는데, num의 값은 10이기 때문에 println에서 값 10을 출력

```
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

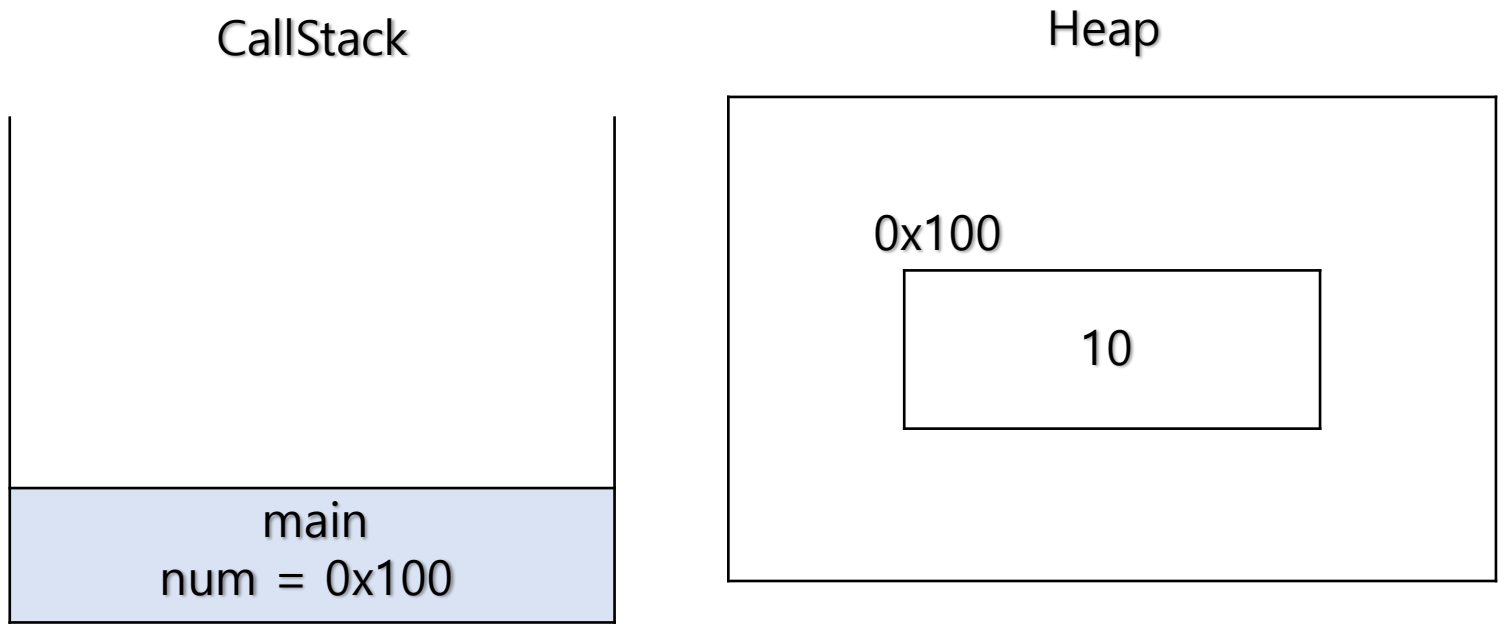


기본형 매개변수

- 설명

println 메소드 호출이 종료되어 사라지면서 main 메소드가 실행 상태로 변경

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

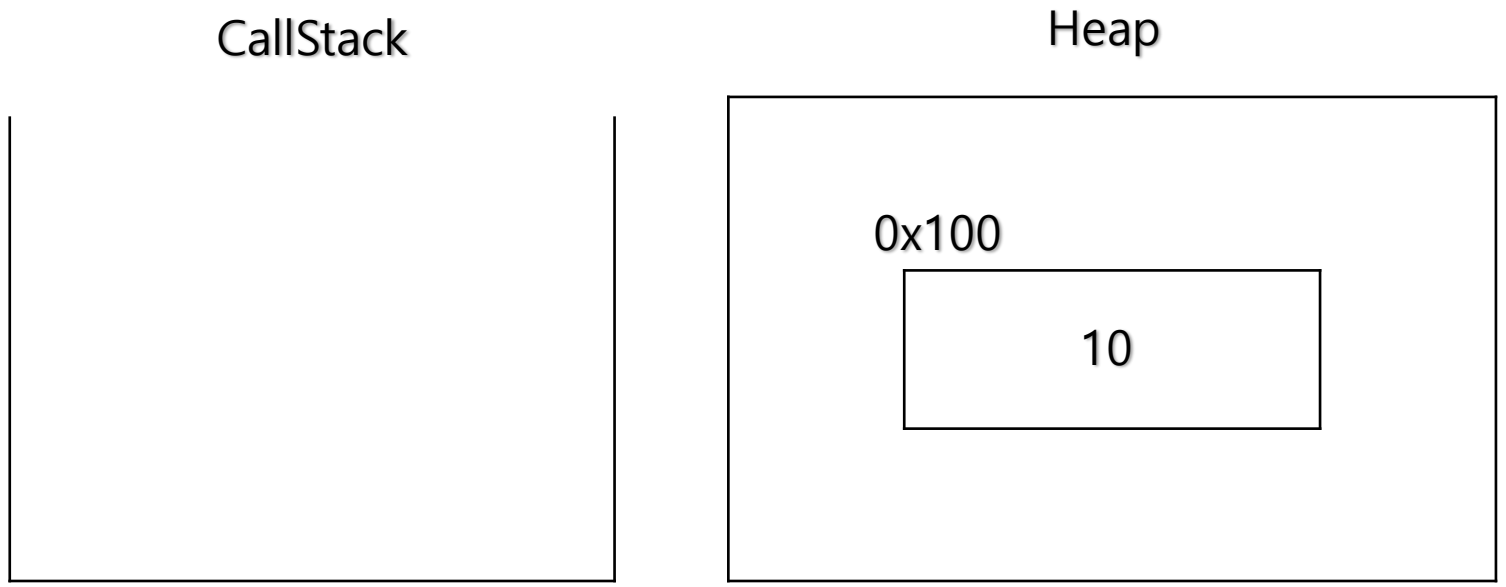


기본형 매개변수

- 설명

main 메소드가 끝나면서 Call Stack이 비워짐

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



참조형 매개변수

참조형 매개변수

- **참조형 매개변수**

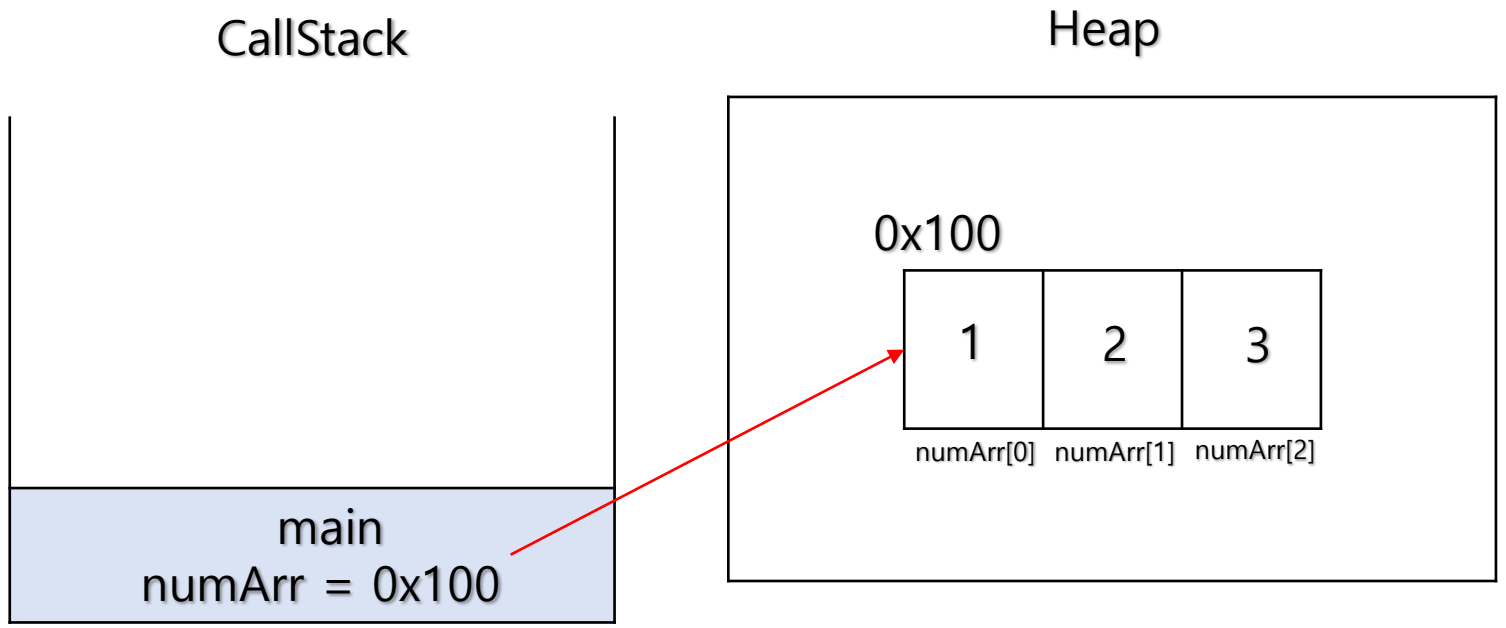
  메소드 매개변수에 전달을 할 때 주소를 전달함

참조형 매개변수

- 설명

main 메소드가 호출되어 스택에 올라가있는 상태
main 메소드는 numArr 변수의 위치를 참조

```java
public class CallStack {
    int[] numArr = {1, 2, 3};

    public static void main(String[] args) {
        CallStack cs = new CallStack();

        System.out.println("메소드 호출 전 : " + Arrays.toString(cs.numArr));

        cs.changeArr(cs.numArr);
        System.out.println("메소드 호출 후 : " + Arrays.toString(cs.numArr));
    }

    public void changeArr(int[] numArr) {
        numArr[1] = 1000;
        System.out.println("메소드 호출 : " + Arrays.toString(numArr));
    }
}
```



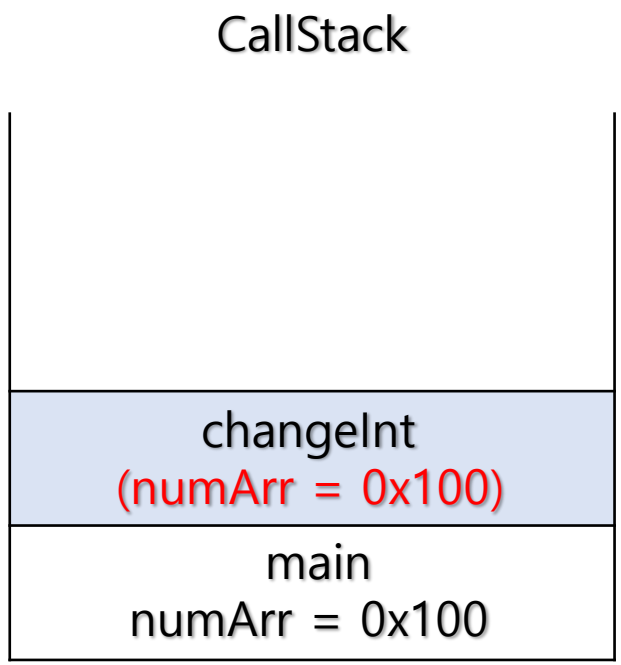
참조형 매개변수

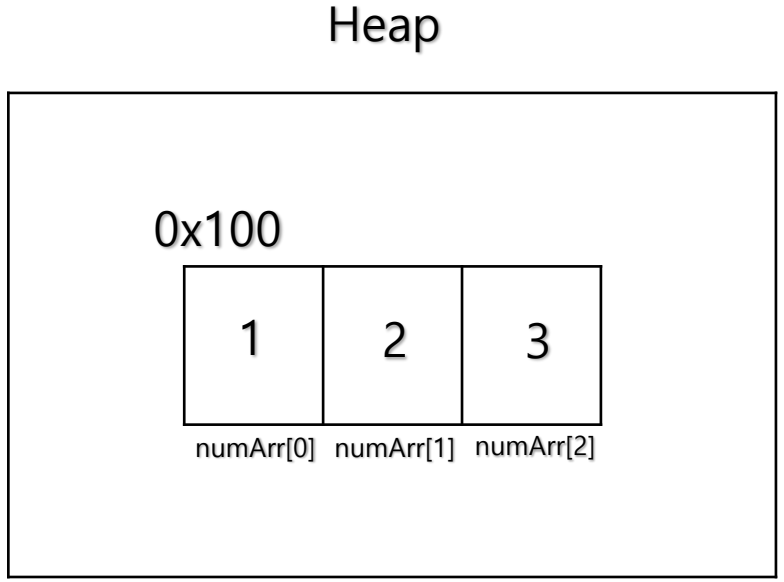
- 설명

changeInt 메소드가 실행되며 값을 전달 (매개변수의 타입이 참조형이기 때문)

```java
public class CallStack {
    int[] numArr = {1, 2, 3};

    public static void main(String[] args) {
        CallStack cs = new CallStack();

        System.out.println("메소드 호출 전 : " + Arrays.toString(cs.numArr));

        cs.changeArr(cs.numArr);
        System.out.println("메소드 호출 후 : " + Arrays.toString(cs.numArr));
    }

    public void changeArr(int[] numArr) {
        numArr[1] = 1000;
        System.out.println("메소드 호출 : " + Arrays.toString(numArr));
    }
}
```

0x100



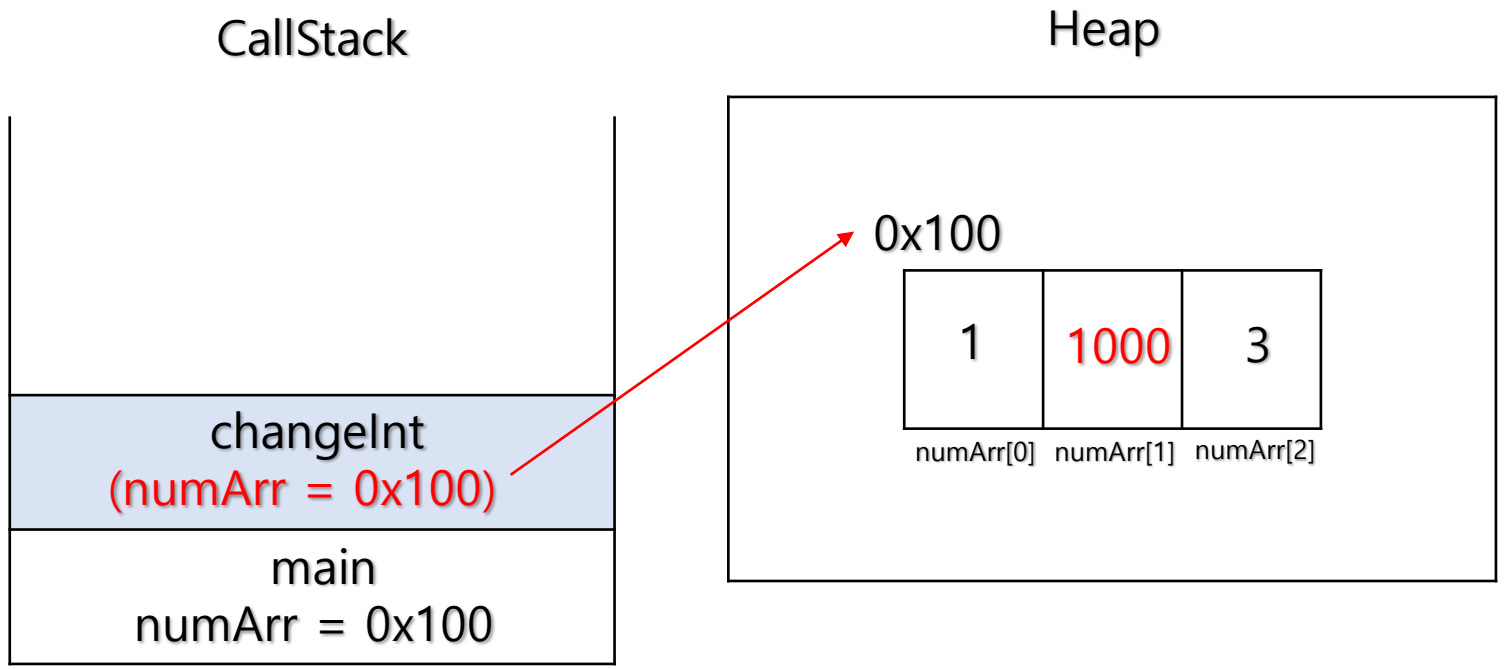


참조형 매개변수

- 설명

전달받은 값을 1000으로 변경

```java
public class CallStack {
    int[] numArr = {1, 2, 3};

    public static void main(String[] args) {
        CallStack cs = new CallStack();

        System.out.println("메소드 호출 전 : " + Arrays.toString(cs.numArr));

        cs.changeArr(cs.numArr);
        System.out.println("메소드 호출 후 : " + Arrays.toString(cs.numArr));
    }

    public void changeArr(int[] numArr) {
        numArr[1] = 1000;
        System.out.println("메소드 호출 : " + Arrays.toString(numArr));
    }
}
```

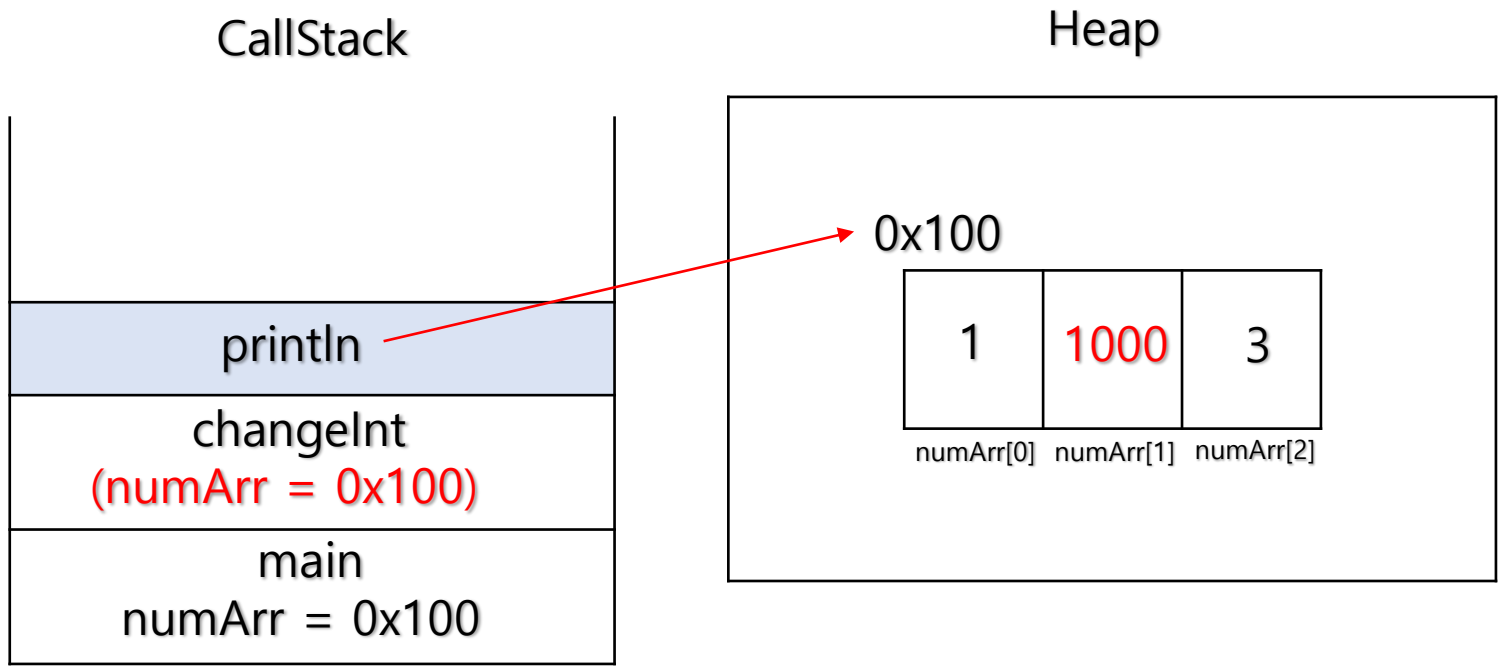


참조형 매개변수

- 설명

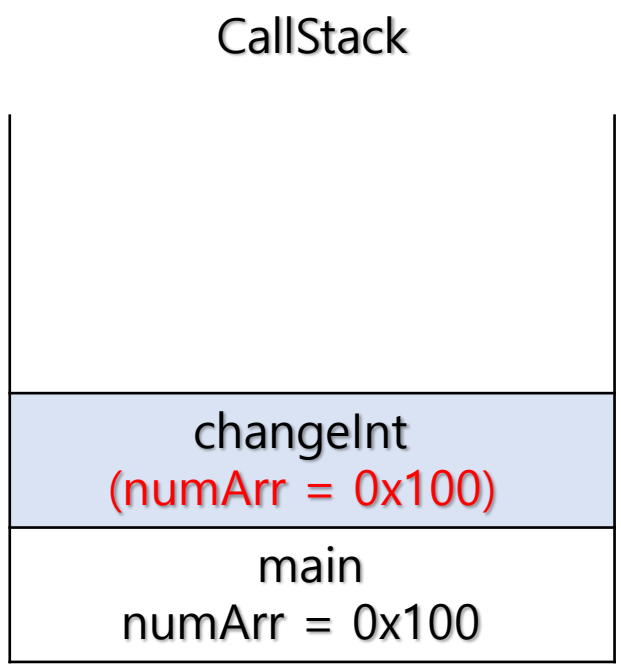
println 메소드가 실행되며 num의 값(1000)을 출력

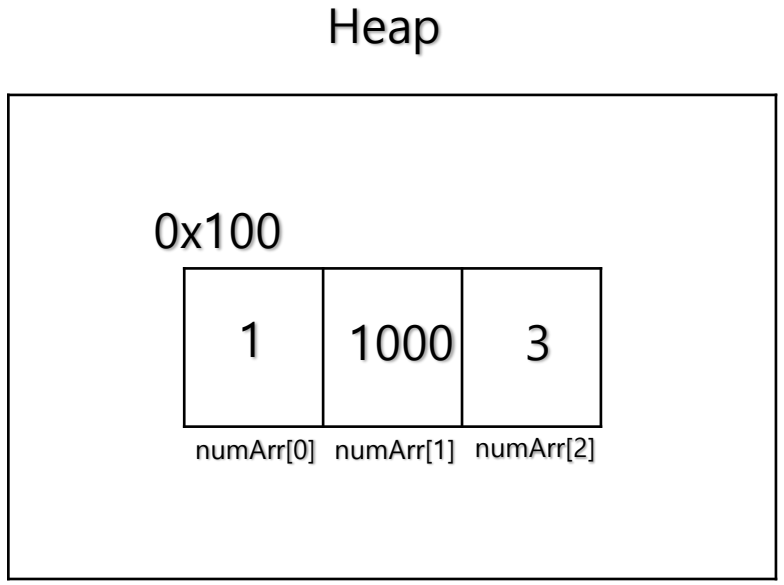
```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



참조형 매개변수

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```



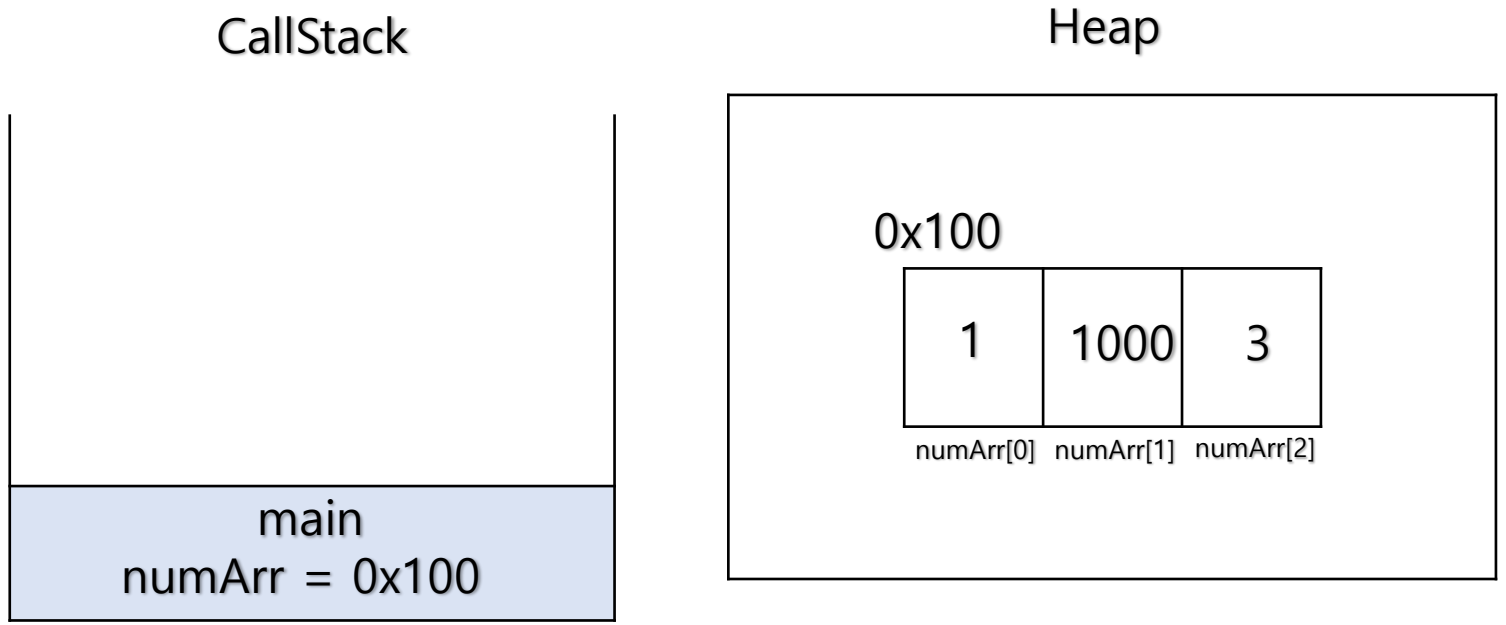


참조형 매개변수

- 설명

changeInt 메소드가 종료되며 main 메소드로 돌아옴

```
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

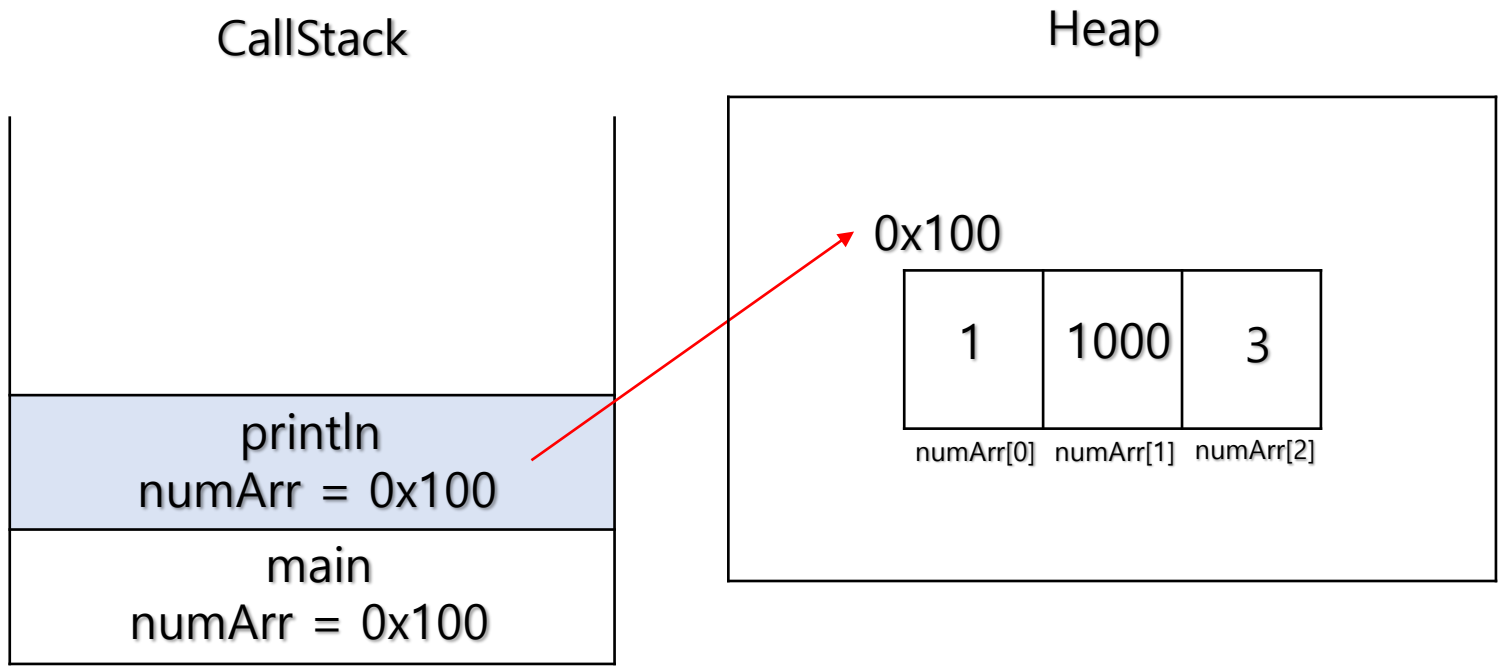


참조형 매개변수

- 설명

main 메소드에 있는 num의 주소를 참조하여 확인 하는데,
numArr의 주소에 값이 1000이 들어있으므로 1000을 출력

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```

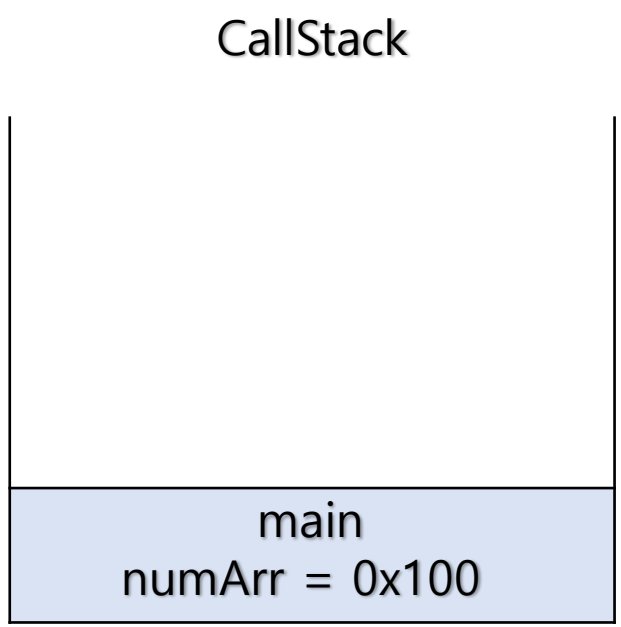


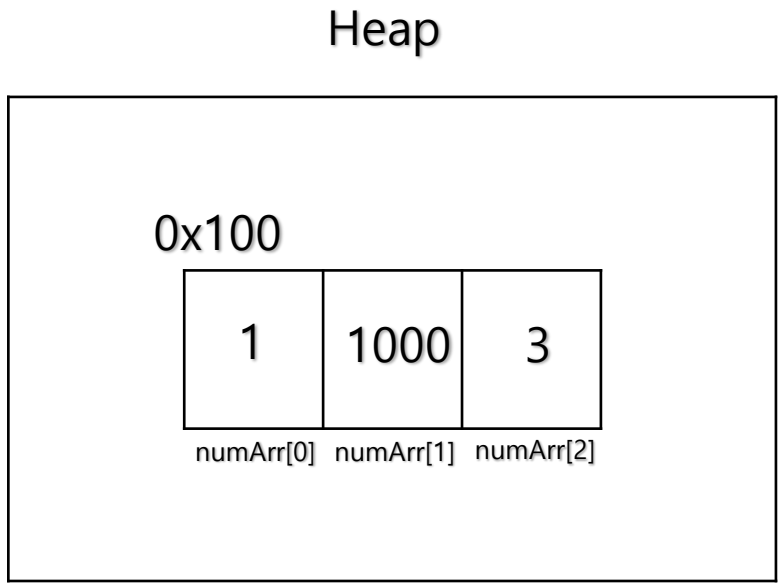
참조형 매개변수

- 설명

println 메소드 호출이 종료되어 사라지면서 main 메소드가 실행 상태로 변경

```java
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```





참조형 매개변수

- 설명

main 메소드가 끝나면서 Call Stack이 비워짐

```
public class CallStack {
    int num;

    public static void main(String[] args) {
        CallStack cs = new CallStack();
        cs.num = 10;
        System.out.println("메소드 호출 전 : " + cs.num);

        cs.changeInt(cs.num);

        System.out.println("메소드 호출 후 : " + cs.num);
    }

    public void changeInt(int num) {
        num = 1000;
        System.out.println("메소드 호출 : "+num);
    }
}
```