Week 1. Key Concepts Often Overlooked

– Reviewing What I Learned at School

황현석 / 25. 04. 05

Table of Contents

Data Type - Basic Primitive Type

종류 \ 크기	1 byte	2 byte	4 byte	8 byte
논리형	boolean			
문자형		char		
정수형	byte	short	**int**	long
실수형			float	**double**

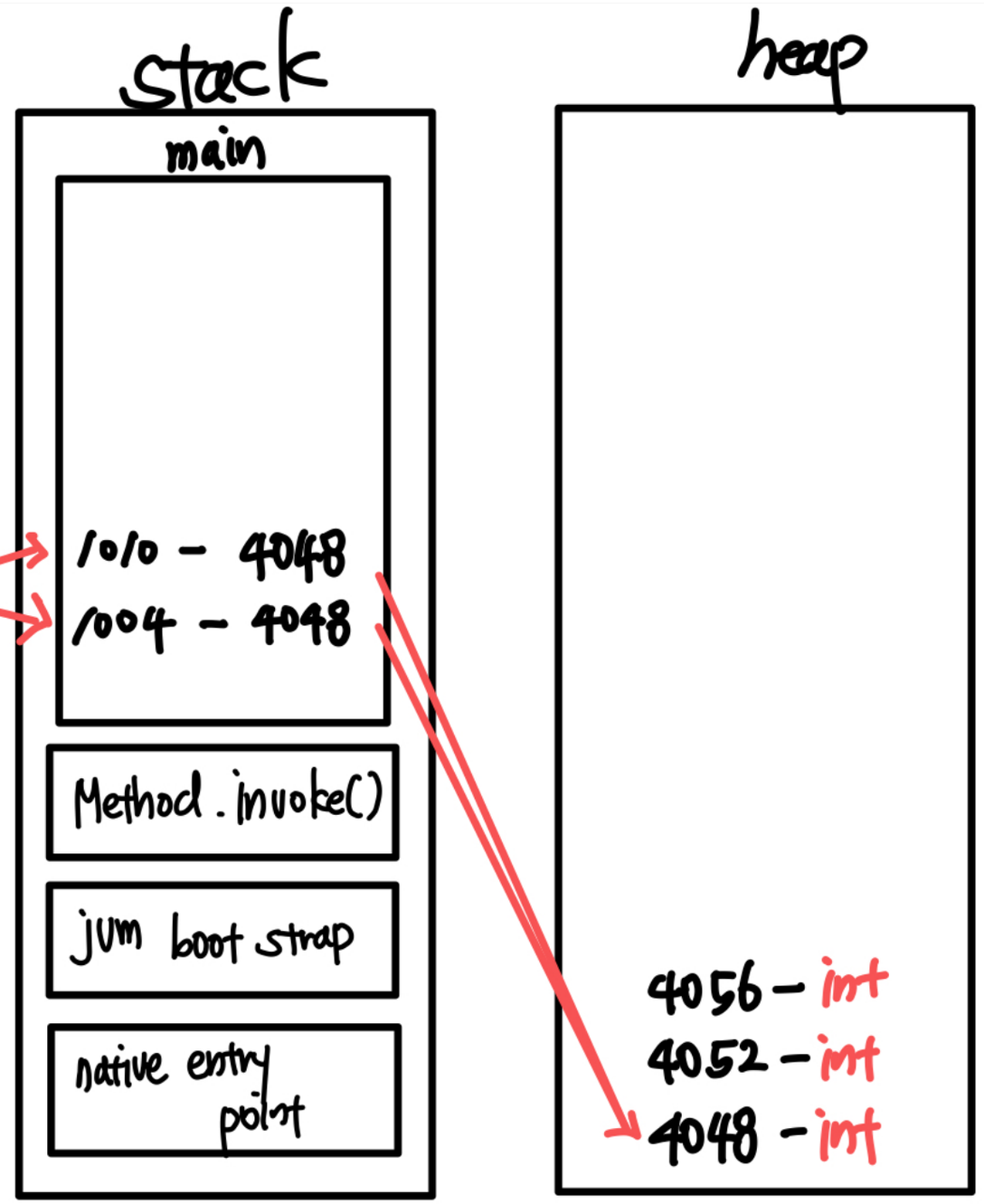
▲ 표2-3 기본형의 종류와 크기

Data Type - Primitive Type vs reference Type

Primitive Type (기본형)	값을 저장하고 있는 메모리 주소를 저장하고 있다.	실제 값으로써, 연산하고 사용할 수 있다.	byte, char, int, long
Reference Type (참조형)	주소 값을 저장하고 있는 메모리 주소를 저장하고 있다.	객체들을 다루고 보관하는데에,사용된다.	Scanner, String, Array, Class etc)..

Reference Type

```
int a[] = new int[3];

int b[] = a;
```



Reference Type

```java
public class Main {
    public static void main(String[] args) throws IOException {
        int a[] = new int[3];
        int b[] = a;

        System.out.printf("a : %s\n", Arrays.toString(a));
        System.out.printf("b : %s\n", Arrays.toString(b));

        a[0] = 10;

        System.out.printf("a : %s\n", Arrays.toString(a));
        System.out.printf("b : %s\n", Arrays.toString(b));
    }
}
```

```
/Users/hyeonseok/Library/Java/JavaVi
a : [0, 0, 0]
b : [0, 0, 0]
a : [10, 0, 0]
b : [10, 0, 0]

Process finished with exit code 0
```

Data Type - String..?

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String a = "abc";
        String b = "def";

        System.out.println(a+b);
    }
}
```

```
/Users/hyeonseok/Library/Java/JavaVirtua
abcdef

Process finished with exit code 0
```

완전, Primitive Type 처럼 작동한다….?

Data Type - String..?

String 은 Class 입니다.

https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/lang/String.java

잘보면, 내부 값은 final 값입니다.

```
144  public final class String
145          implements java.io.Serializable, Comparable<String>, CharSequence,
146                     Constable, ConstantDesc {
147
148      /**
149       * The value is used for character storage.
150       *
151       * @implNote This field is trusted by the VM, and is a subject to
152       * constant folding if String instance is constant. Overwriting this
153       * field after construction will cause problems.
154       *
155       * Additionally, it is marked with {@link Stable} to trust the contents
156       * of the array. No other facility in JDK provides this functionality (yet).
157       * {@link Stable} is safe here, because value is never null.
158       */
159      @Stable
160      private final byte[] value;
161
162      /**
163       * The identifier of the encoding used to encode the bytes in
164       * {@code value}. The supported values in this implementation are
165       *
166       * LATIN1
167       * UTF16
168       *
169       * @implNote This field is trusted by the VM, and is a subject to
170       * constant folding if String instance is constant. Overwriting this
171       * field after construction will cause problems.
172       */
173      private final byte coder;
174
175      /** Cache the hash code for the string */
176      private int hash; // Default to 0
177
178      /**
179       * Cache if the hash has been calculated as actually being zero, enabling
180       * us to avoid recalculating this.
181       */
182      private boolean hashIsZero; // Default to false;
```

Data Type - Why the +Operator Works in class…?

그냥 단지, Java Compiler가 String에 대해서 특별대우를 해줌.

```java
public static void main(String[] args) throws I
    String s = "a" + "b" + "c";
}
```

```java
String s = new StringBuilder()
        .append("a")
        .append("b")
        .append("c")
        .toString();

// s == "abc"
```

컴파일러가 알아서 다음과 같이 바꿔 준다.

Data Type - What is StringBuilder()?

StringBuilder()는 효율적인, String 구축을 위해 사용하는 클래스이다.

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String result = "";

        for (int i=0;i<1000;i++) {
            result += i;
        }
    }
}
```

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String result;
        StringBuilder sb = new StringBuilder();

        for (int i=0;i<1000;i++) {
            sb.append(i);
        }

        result=sb.toString();
    }
}
```

Data Type - What is Constant…?

Constant, 컴파일 전에 결정되어 있는 값, 리터널이라고도 한다.

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String g = "a" + "b" + "c";
        String name = "황현석";
        int a = 5, b = 6;
        int c = 7;
        int s = 1;
    }
}
```

정수들은 변수에 직접 저장 된다.

String 변수들은, 메모리에서 만들어 진 곳을 가르킨다.

Data Type - How Strings Are Stored at Compile Time

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String a = "abc";
        String b = "abc";

        System.out.println(a == b);
    }
}
```

String Literal은 메모리의 Heap 영역의 String Constant Pool 영역에 String 객체로써 저장된다.

리터널을 변수에 할당 할 때, 이미 String Constant Pool에 동일한 리터널이 있으면, 그 객체를 할당시킨다.

reference type 간의 == 연산자는 주소를 비교한다.

```
/Users/hyeonseok/Library/Java/JavaVirtual
true

Process finished with exit code 0
```

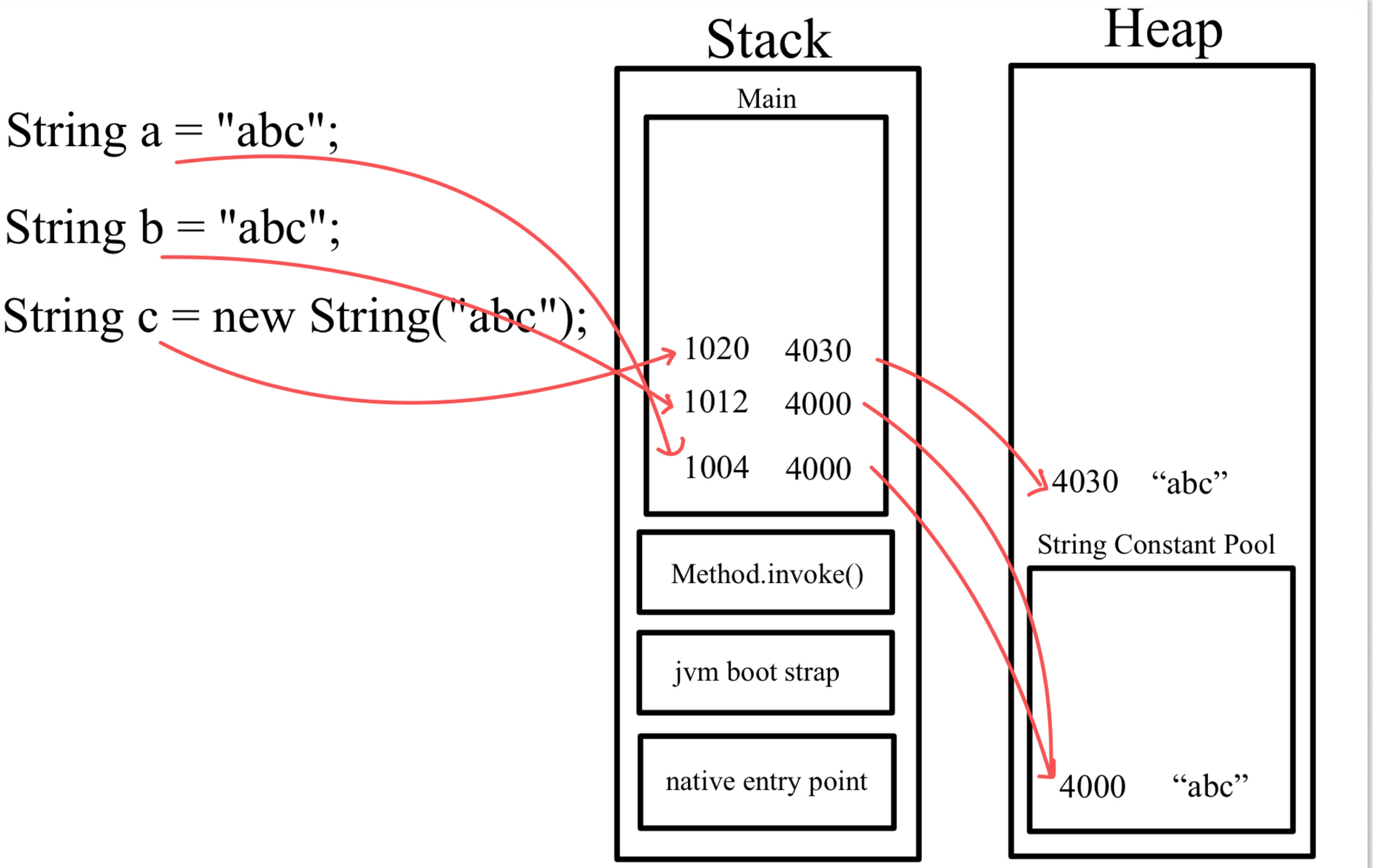
Data Type - How Strings Are Stored at Compile Time

```java
public class Main {
    public static void main(String[] args) throws IOException {
        String a = "abc";
        String b = "abc";
        String c = new String( original: "abc");

        System.out.printf("a == b -> %b\n", a == b);
        System.out.printf("a == c -> %b\n", a == c);
    }
}
```

```
/Users/hyeonseok/Library/Java/JavaVirtualMach
a == b -> true
a == c -> false

Process finished with exit code 0
```

a와 b는 Heap 영역의 String Constant Pool에서 만들어진 객체를 할당 받았고, c는 할당받은 객체를 이용하여, 새로운 객체를 만들었다.

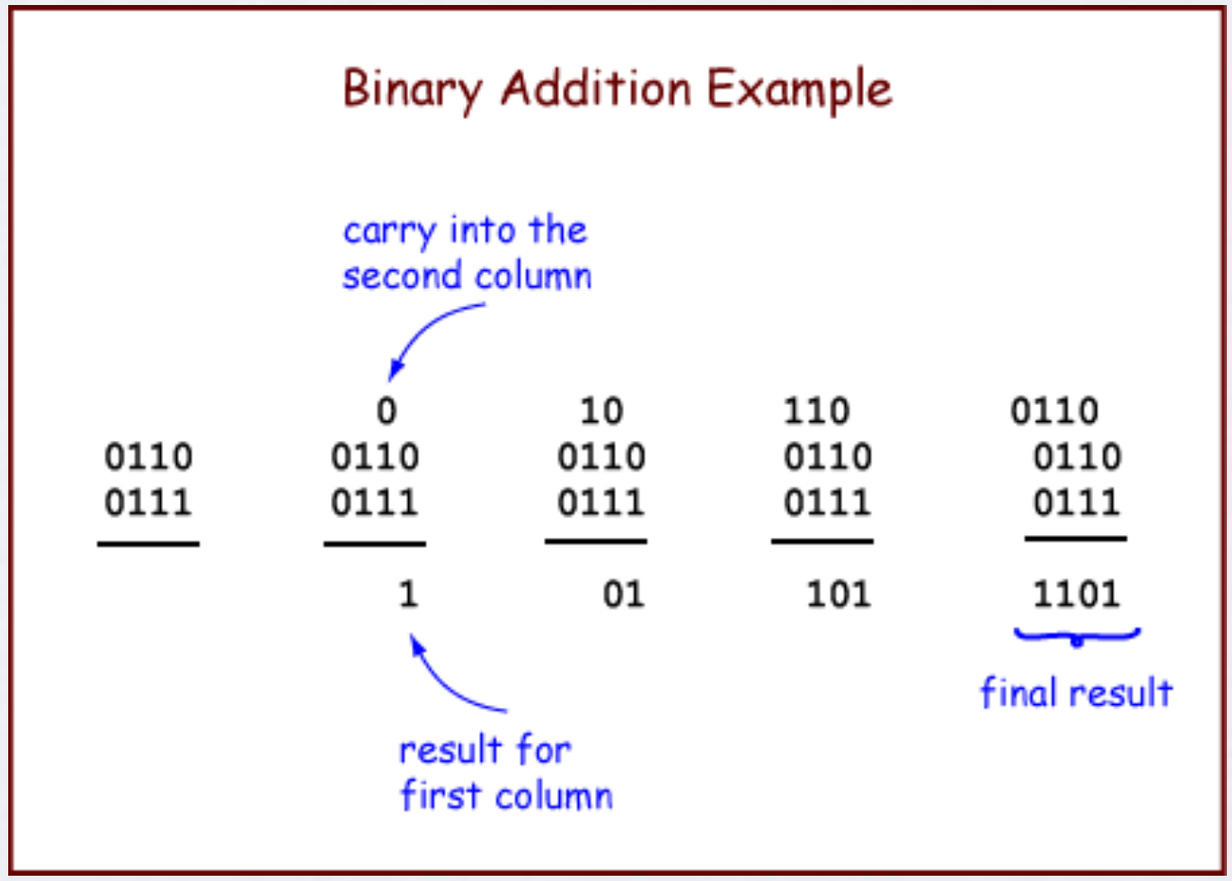
Data Type - How Strings Are Stored at Compile Time



2's Complement

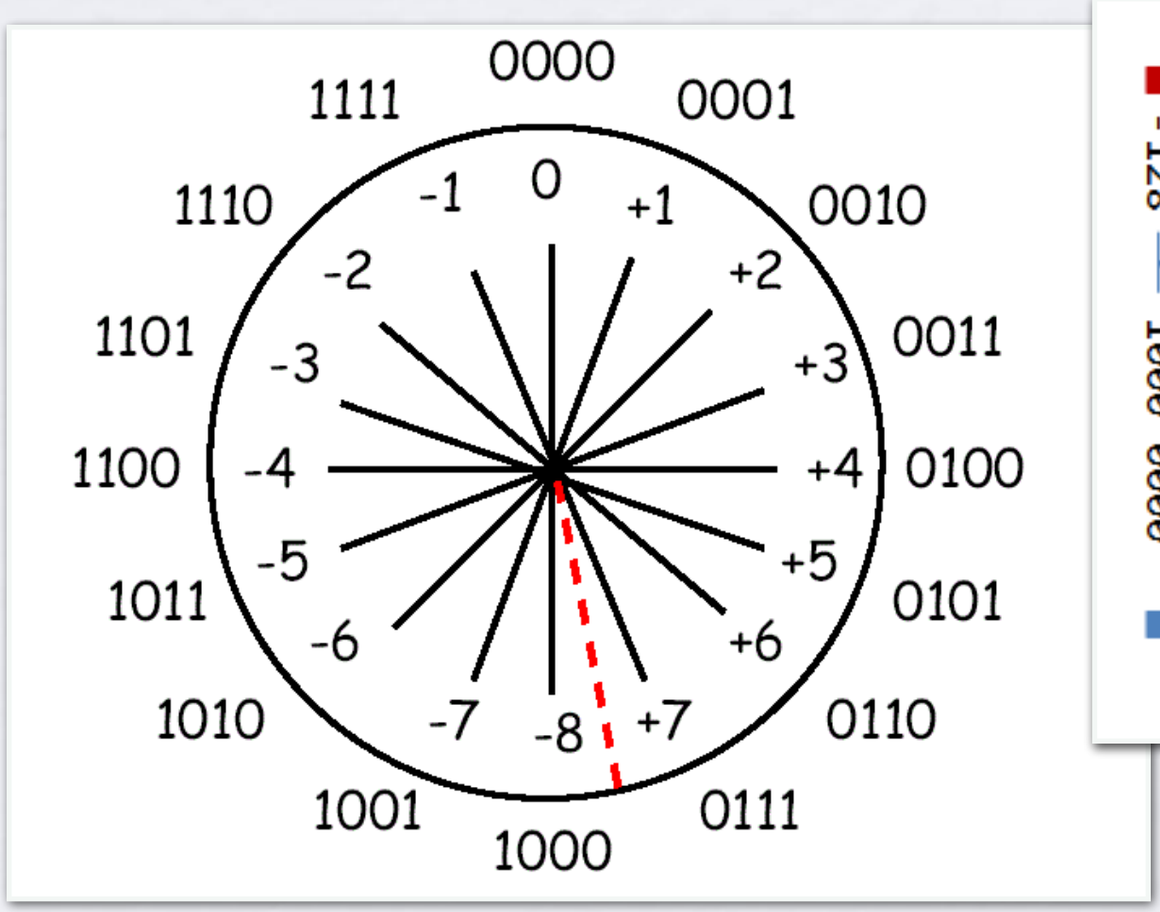
- Integer Representation in Computers

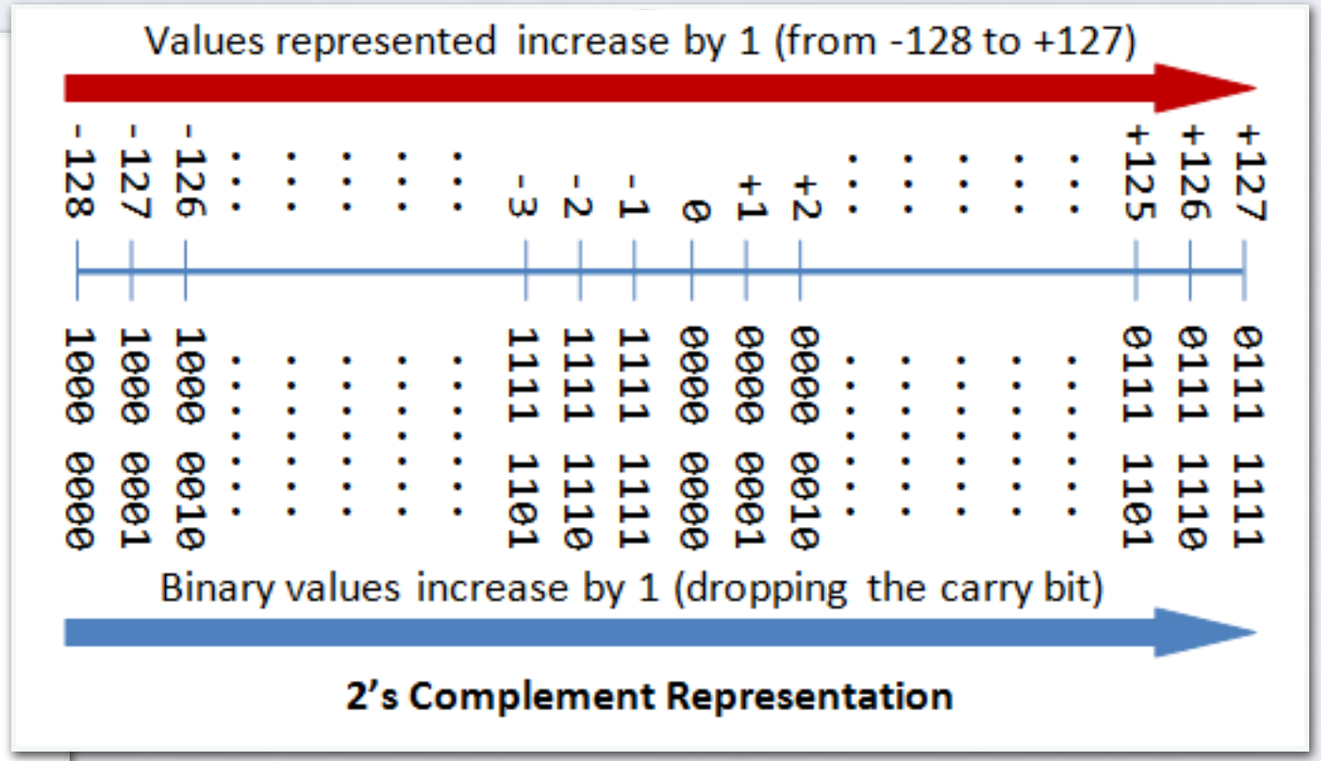
10진법	2진법	8진법	16진법
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A



2's Complement

- Integer Representation in Computers





Variable Naming...

camelCase (낙타 표기법)

```java
public class Main {
    public static void main(String[] args) throws IOException {
        int studentAge;
        String myName;
        boolean isLoggedIn, isReady, isSorted;
    }
}
```

Pascal Case (파스칼 표기법)

```java
class Node {  no usages
    int node;  no usages
}

class DataStructure <T> {  no usages
    T data;  no usages
}

public class Main {
```

Variable Naming...

snake_case (뱀 표기법)

```
public static void main(String[]
    int student_age;
    String my_name;
    boolean is_logged_in;
    boolean is_ready;
    boolean is_valid;

}
```

UPPER_SNAKE_CASE (대문자 스네이크)

```
public static final int MAX_SIZE = 10;  no usa
public static final double PI = 3.141592;  no
```

```
public static void main(String[] args)
    double pi = Math.PI;
    double e = Math.E;

}
```

```
public static void main(String[] args) throws
    int MAX = Integer.MAX_VALUE;
    int MIN = Integer.MIN_VALUE;

    long MIN_2 = Long.MIN_VALUE;
    long MAX_2 = Long.MAX_VALUE;

    double MIN_3 = Double.MIN_VALUE;
    double MAX_3 = Double.MAX_VALUE;
}
```

Ternary operator

```java
public class Main {
    public static void main(String[] args) throws IOException {
        int score = new Random().nextInt( bound: 100);

        int bin = (64 < score) ? 1 : 0;
```

(조건식) ? (참일 때) : (거짓 일 때)

단순 알고리즘 구현에서는 사용하는 경우가 없지만,
실제 프로젝트 구현시에는 많이 사용한다.

Ternary operator

```java
public class Main {
    public static void main(String[] args) {
        int score = new Random().nextInt( bound: 100);

        if (score < 60) {
            System.out.println("PASS");
        } else {
            System.out.println("FAIL");
        }
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        int score = new Random().nextInt( bound: 100);

        String result = score < 60 ? "PASS" : "FAIL";
        System.out.println(result);
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        int score = new Random().nextInt( bound: 100);
        System.out.println(score < 60 ? "PASS" : "FAIL");
    }
}
```

Ternary operator

```java
public class Main {
    public static void main(String[] args) {
        int score = new Random().nextInt( bound: 100);

        int result;

        if (score < 25)  {
            result = 1;
        } else if (score < 50) {
            result = 2;
        } else if (score < 75) {
            result = 3;
        } else {
            result = 4;
        }
    }
}
```

Practice Time..

Q. 삼항 연산자 만을 사용하여, result가 위의 코드가 실행 됐을 때와 같은 값을 가지게 하시오.

tip. 삼항 연산자 의 결과 값으로 삼항 연산자를 넣을 수 있다.

Ternary operator

```
public class Main {
    public static void main(String[] args) {
        int score = new Random().nextInt( bound: 100);
        int result = score < 25 ? (1) : (score < 50 ? (2) : (score < 75 ? (3) : (4)));
    }

}
```

```
//reponsivility
body:
    width < 600
        ? Column(
          children: [
            Chart(expenses: _registeredExpenses),
            Expanded(child: mainContent),
          ],
        ) // Column
        : Row(
          children: [
            Expanded(child: Chart(expenses: _registeredExpenses)),
            Expanded(child: mainContent),
          ],
        ), // Row
```

Bit Operator

Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

신입생때는 비트 연산자를 많이 다룰 기회가 사실 없다.

멘토멘티 때라도, 많은 응용법을 알아 두면 좋을 것 같다.

Bit Operator

실제 개발 환경에서, 상태 관리를 유용하게 하기 위해서, 각 자리 별 비트의 상태를 통해, 다양한 상황이나 상태를 관리하기도 한다.

K번째, 비트의 상태를 잘 알아야 하는데, 어떻게 각 자리의 비트에 쉽게 접근 할 수 있을까?

기본적으로 K번째, 비트를 컨트롤 하기 위해서, (1 << K)의 수를 만든다.

해당 수를 사용해 목표 달성에 필요한 연산을 한다.

```
int G = 0b01101101010111111110100101101010;
    1 = 0b00000000000000000000000000000001;
1 << 3 = 0b00000000000000000000000000001000;
```

Bit Operator

```
int G = 0b01101101010111111110100101101010;
```

Q. 32비트 정수형 변수, G의 오른쪽에서 3번째 (0-based) 비트를 1로 바꾸시오.

Q. 32비트 정수형 변수, G의 비트를 오른쪽으로 2번 옮기시오.

Q. 32비트 정수형 변수, G를 바이너리 형태로 출력하시오.
Integer.toBinaryString(int) 메서드가 존재한다.

Q. 32비트 정수형 변수, G의 7번째 (0-based) 비트 가 켜져 있으면, Yes 그렇지 않으면. No를 출력 하시오.

Bit Operator

```
int G = 0b01101101010111111110100101101010;
```

Q. 32비트 정수형 변수, G의 26번째 비트가 켜져있으면 끄고, 꺼져 있으면 키시오.

Q. 32비트 정수형 변수, G의 비트를 6번째 비트를 제외하고 모두 0으로 만드시오.

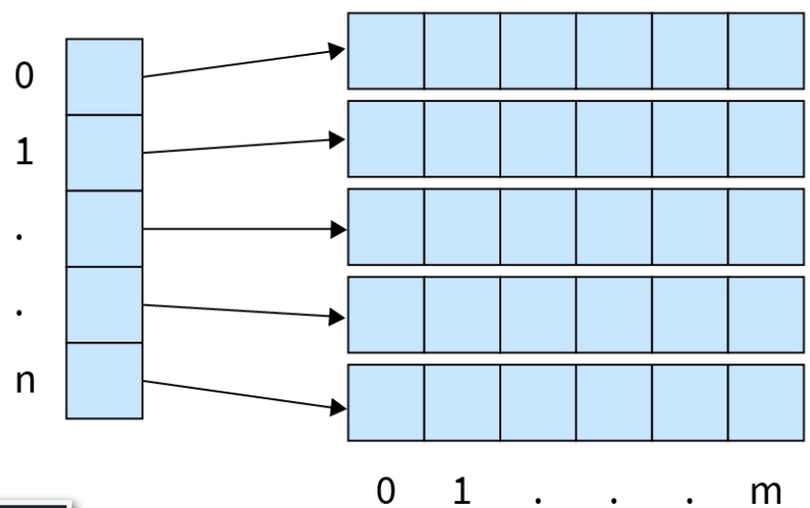
Q. 32비트 정수형 변수, G의 모든 비트를 반전 시키시오.

Q. 32비트 정수형 변수, G가 양수일 때, 짝수 인지, 홀수 인지 비트 연산자를 응용하여, 확인 하시오.

Bit Operator

```java
int G = 0b01101101010111111110100101101010;

//Q1.
int Q1 = G | (1 << 3);
//Q2.
int Q2 = G >> 2;

//Q3.
System.out.println(Integer.toBinaryString(G));

//Q4.
String result = (G & (1 << 7)) > 0 ? "Yes" : "No";
System.out.println(result);

//Q5.
int Q5 = G ^ (1 << 26);

//Q6.
int Q6 = G & (1 << 6);

//Q7.
int Q7 = ~G;

String Q8 = (G & (1<<0)) > 0 ? "odd" : "even";
```

Array....



Total Elements (n * m)

```java
public class Main {
    public static void main(String[] args) {
        int N = 10, M = 20;

        int [][]arr = new int[N][M];

        for (int i=0;i<N;i++) {
            for (int j=0;j<M;j++) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Array....

```java
public class Main {
    public static void main(String[] args) {
        int N = 10, M = 20;

        int [][]arr = new int[N][M];

        for (int i=0;i<N;i++) {
            for (int j=0;j<M;j++) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

[ ]가 무엇일까…? 2차원배열일 때, 한개만 쓴다면 그것은 뭘 의미 하는 걸까…?

Array Access Operator [ ]

[ ] 는 사실 배열 접근 연산자다.

배열에 사용하는 연산자 이며, 배열 안의 원소의 값을 설정하거나 가져오는 연산자이다.

```
int array[] = new int[4];
(array)[2] = 1;
```

그럼.. 2차원 배열 arr이 있을 때, arr[0]은 뭘까..?

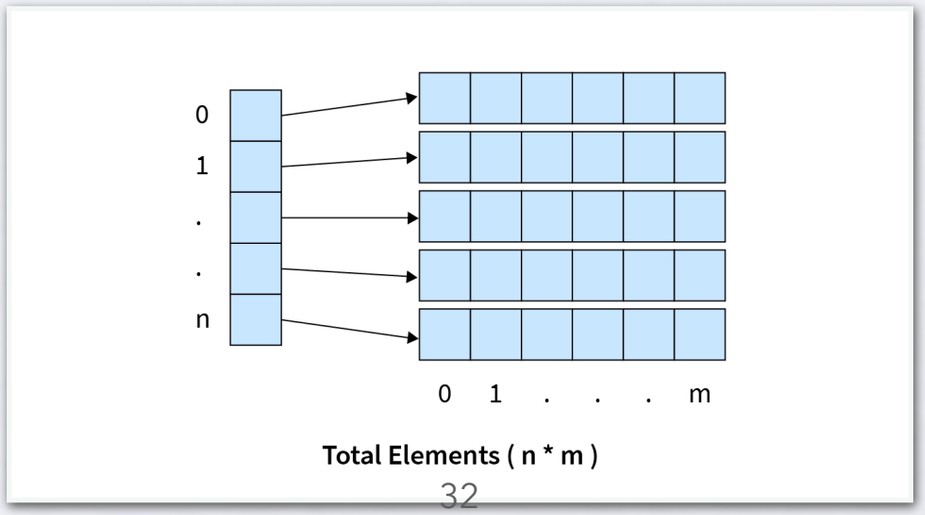
```
public class Main {
    public static void main(String[] args) {
        int N = 10, M = 20;
        int [][]arr = new int[N][M];

        //what is arr[0]...?
```

Array Access Operator [ ]

```java
public class Main {
    public static void main(String[] args) {
        int N = 10, M = 20;
        int [][]arr = new int[N][M];

        System.out.println(arr[0].getClass().getTypeName());
    }
}
```

```
/Users/hyeonseok/Library/Java/JavaV
int[]

Process finished with exit code 0
```

놀랍게도…? arr[0] 은 int 배열이다…

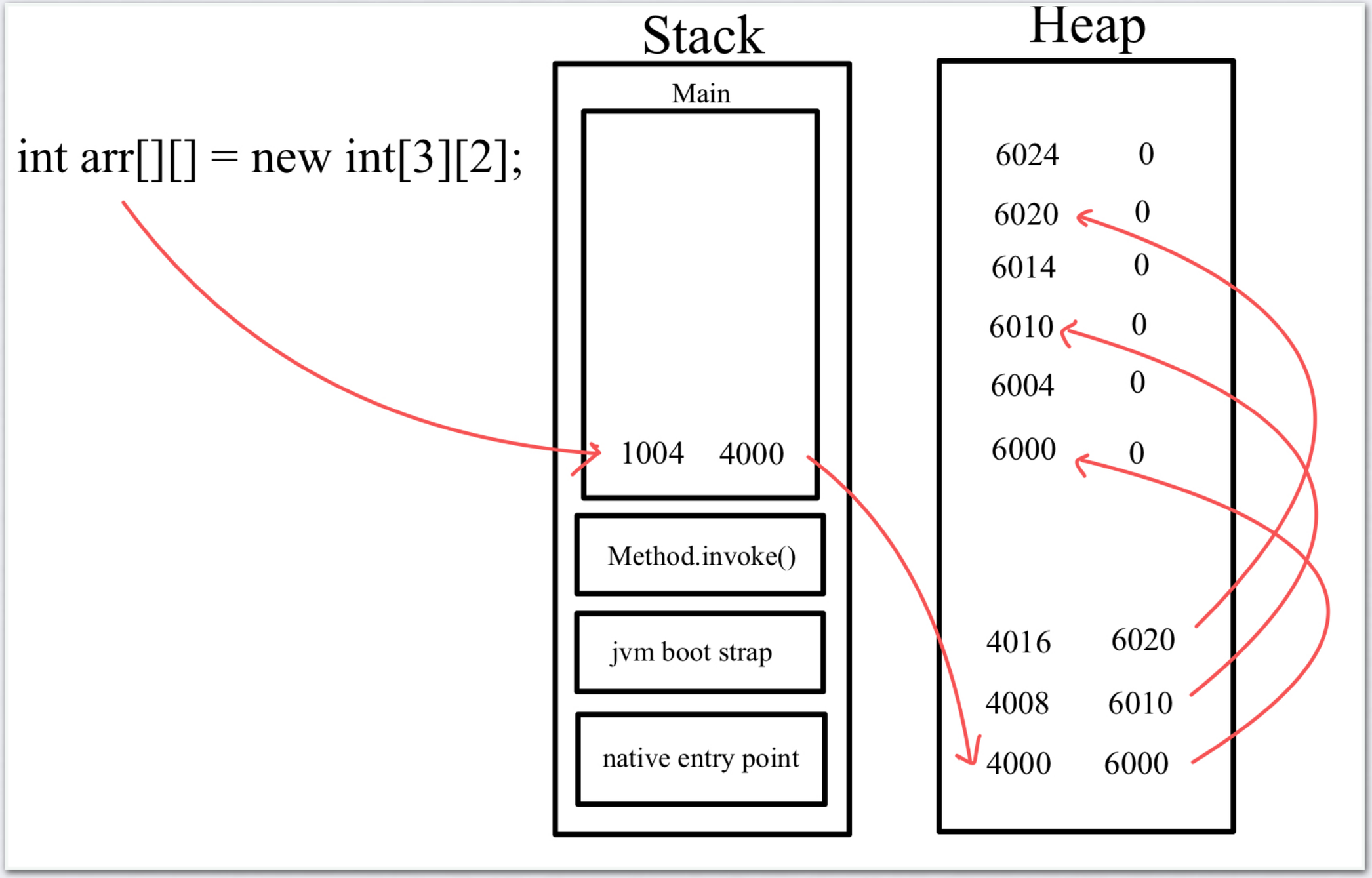


Array Access Operator [ ]

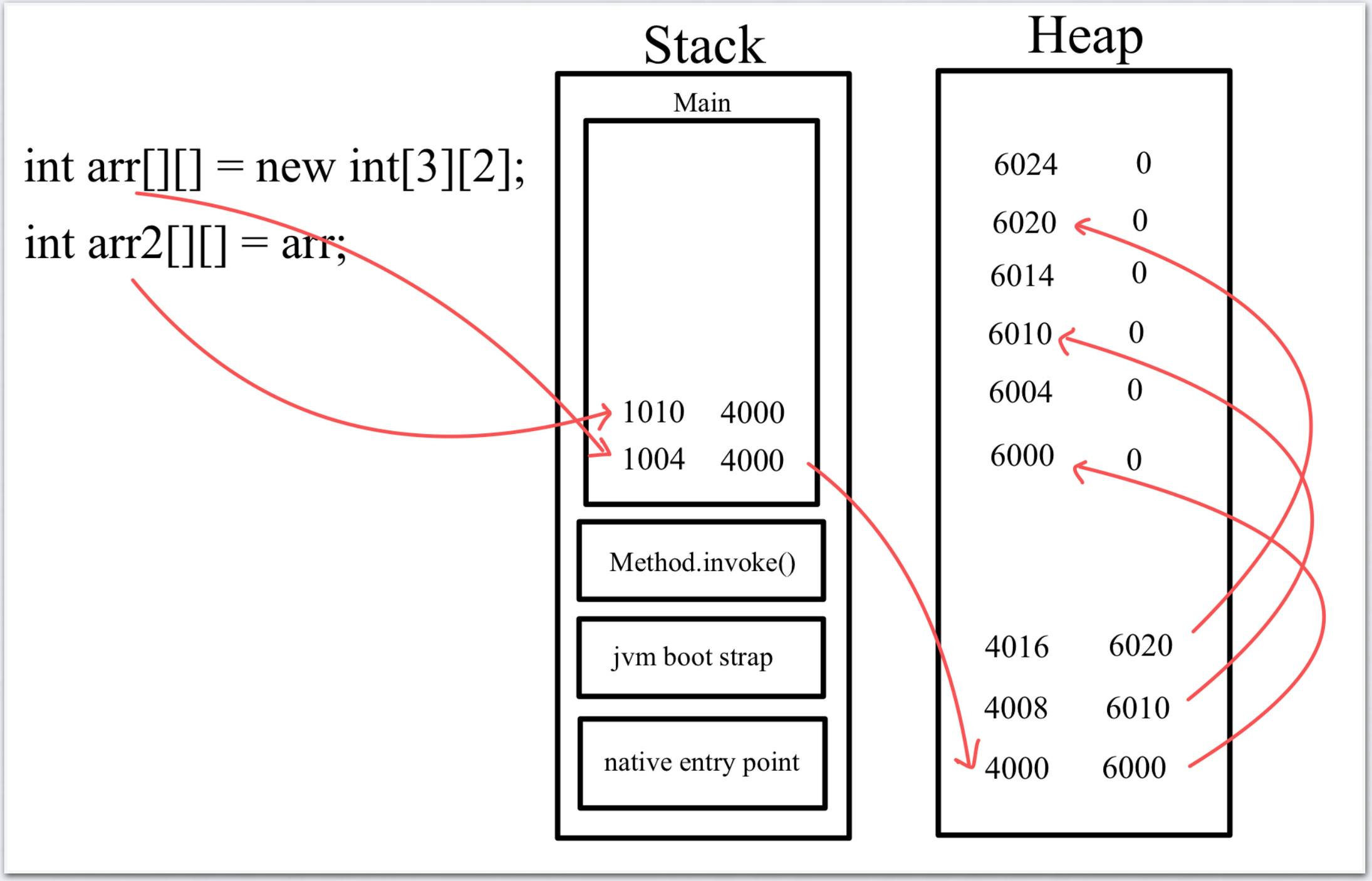
```java
public class Main {
    public static void main(String[] args) {
        int N = 10, M = 20;
        int [][]arr = new int[N][M];

        (arr[3])[5] = 1;
    }
}
```

말 그대로 접근을 도와주는 “연산자” 이기 때문에,
다음과 같이 사용해도 문제 없다.

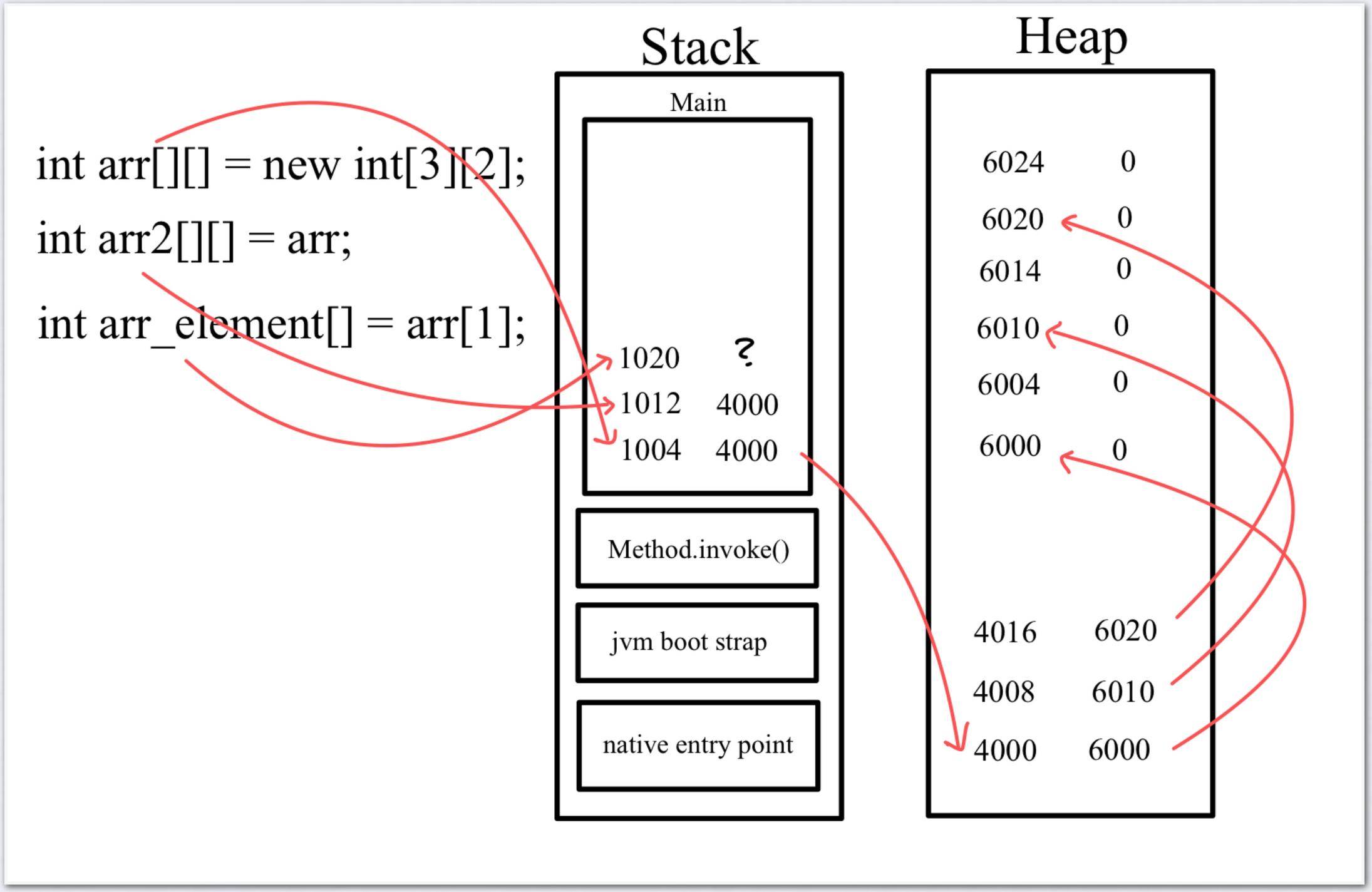
Array Access Operator [ ]



Array Access Operator [ ]



Array Access Operator [ ]



Array Access Operator [ ]

```java
public static void print (int [][]arr) {  no usages
    for (int i = 0; i < arr.length; i++) {

        for (int j = 0; j < arr[i].length; j++) {
            System.out.print(arr[i][j] + " ");
        }

        System.out.println();
    }

}
```

다음과 같이, 배열의 크기가 모르는 상황에서 동적으로 크기를 알아내어 전체를 순회 시킬 수 있다.

Array Access Operator [ ]

```java
public class Main {
    public static void main(String[] args) throws IOException {
        int [][]arr = new int[3][5];
        arr[1] = new int[100];

        for (int i=0;i<3;i++) {
            System.out.println(arr[i].length);
        }
    }
}
```

arr[ ]도 배열이다. arr[ ]은, int 배열의 주소를 원소로 가지는 배열이다.

arr[k] = new int[N]; 을 통해, k번째 배열을 새롭게 만들어 할당 가능.

```
/Users/hyeonseok/Library/Java/JavaVirtua
5
100
5
```

Array Access Operator [ ]

```java
public class Main {
    public static void main(String[] args) throws IOException {
        int [][]arr = new int[3][5];
        arr[1] = new int[100];

        for (int i=0;i<3;i++) {
            System.out.println(arr[i].length);
        }
    }
}
```

arr[ ]도 배열이다. arr[ ]은, int 배열의 주소를 원소로 가지는 배열이다.

arr[k] = new int[N]; 을 통해, k번째 배열을 새롭게 만들어 할당 가능.

```
/Users/hyeonseok/Library/Java/JavaVirtua
5
100
5
```

Problem Solving...

2750번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 질문 게시판

수 정렬하기

시간 제한	메모리 제한	제출	정답
1 초	128 MB	235427	135561

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

https://www.acmicpc.net/problem/2750

Sort함수를 사용하지 말고, 직접 Bubble Sort를 구현하여 풀자

We’ve completed Week 1