배열(Array)

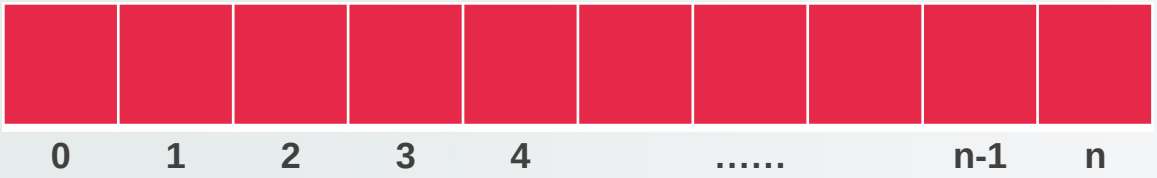
배열

배열

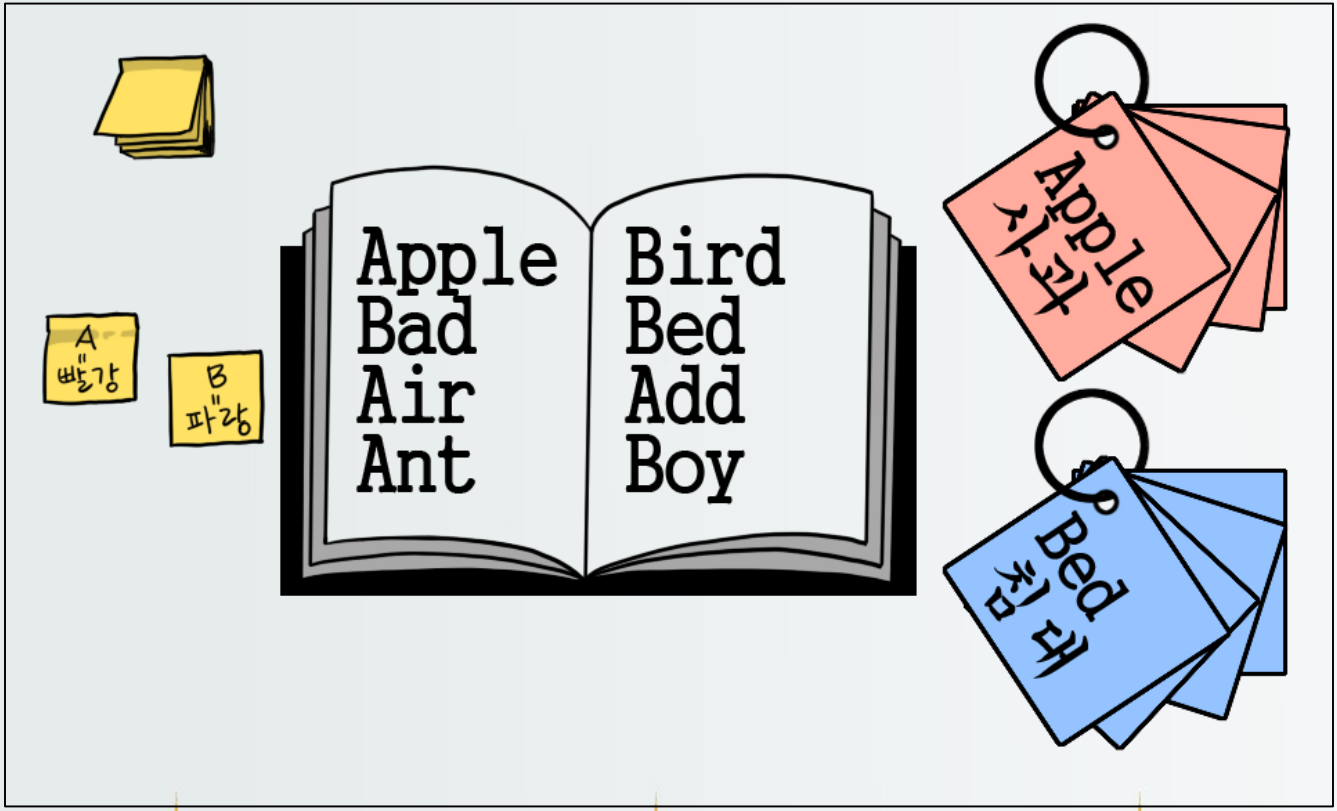
배열

→ 동일한 자료형을 가지는 변수들의 집합
→ 동일 자료형의 묶음

배열은 저장된 값마다 인덱스 번호가 설정됨(0부터 시작)



배열



배열

배열의 선언

→ 주소 값을 가지지 않은 변수 생성(Stack)

자료형 [] 변수명; → **int [] arr;**

자료형 변수명[]; → **int arr [];**

배열의 할당

→ 배열객체 생성 후 변수에 주소 값 할당(Heap)

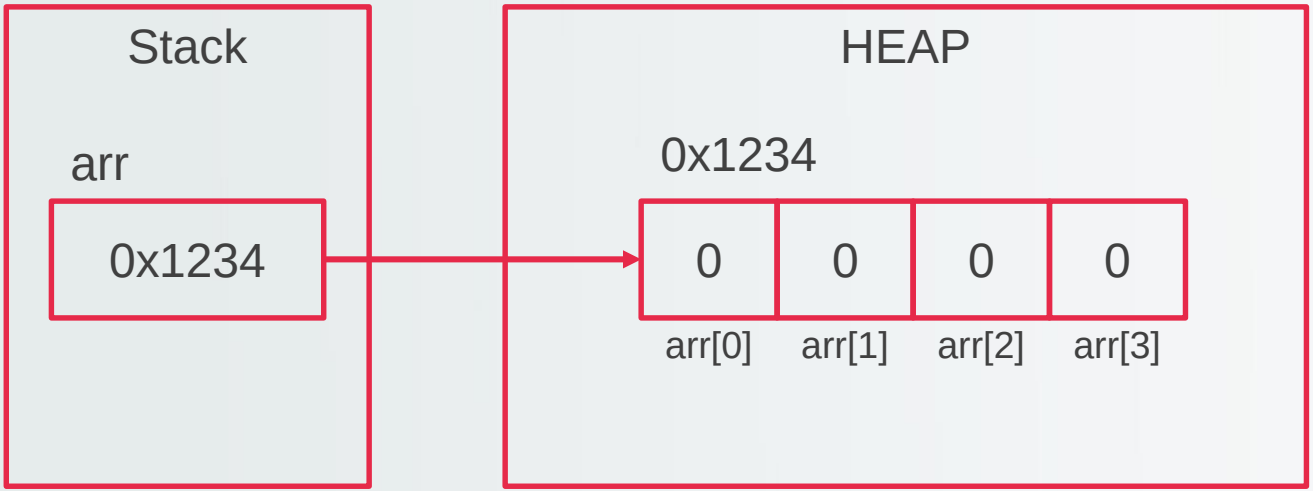
변수명 = new 자료형[저장할 데이터 개수]; → **arr = new int[5];**

→ 정수 5개 저장할 수 있는 배열

배열

배열의 저장 구조

int [] arr = new int[4]; → 정수 4개를 저장할 배열 생성



배열

배열의 초기화

→ 배열을 선언 후 할당하게 되면 각 자료형 별 기본값이 들어 감
→ 숫자형 변수 모두 0, boolean : false, char : ‘ ’
→ 선언과 동시에 값을 기록 할 수 있음

int[] arr = {1,2,3,4,5};
→ 길이가 5인 정수형 배열을 생성하면서 각 인덱스에 1,2,3,4,5 순서대로 저장

String[] arr = {“hi”, “hello”, “bye”};
→ 길이가 3인 문자열 배열을 생성하면서 각 인덱스에 “hi”, “hello”, “bye”를 순서대로 저장

※ 기본형 변수 8개에 해당하는 배열은 배열 할당 시 기본값으로 초기화 되어 바로 사용해도 문제가 없지만, 참조형 변수를 사용하는 경우 반드시 값을 초기화해서 사용해야 함

배열

배열 값 기록

1. 인덱스를 이용한 직접 값 기록

```
int [] arr = new int[3];
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
```

2. for문을 이용한 값 기록

```
int [] arr = new int[3];
for(int i=0;i<arr.length;i++){
    arr[i] = i+1;
}
```

※ 배열 인덱스에 들어갈 값이 일정한 규칙으로 증가한다면 반복문의 index를 통해 기록 가능

배열

for문을 이용한 사용자 입력 데이터 기록

```
Scanner sc = new Scanner(System.in);
int [] arr = new int[5];
for(int i=0;i<arr.length;i++){
    System.out.print(“배열에 저장할 값 입력 : ”);
    arr[i] = sc.nextInt();
}
```

배열

배열 값 기록

→ 결국 배열은 []안에 인덱스를 통해 값에 접근을 한다는 것을 이용하면 변수를 이용한 처리가 가능

```
int [] arr = new int[3];
int index = 0;
arr[index++] = 1;        //arr[0] = 1; 을 수행하고 index값이 1증가
arr[index++] = 2;
arr[index++] = 3;
```

※ 배열명[인덱스번호]가 하나의 변수명이라고 생각하면 됨

배열

배열 값 출력

1. 인덱스를 이용한 직접 값 출력

```
int [] arr = new int[3];
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
System.out.println(arr[0]);
System.out.println(arr[1]);
System.out.println(arr[2]);
```

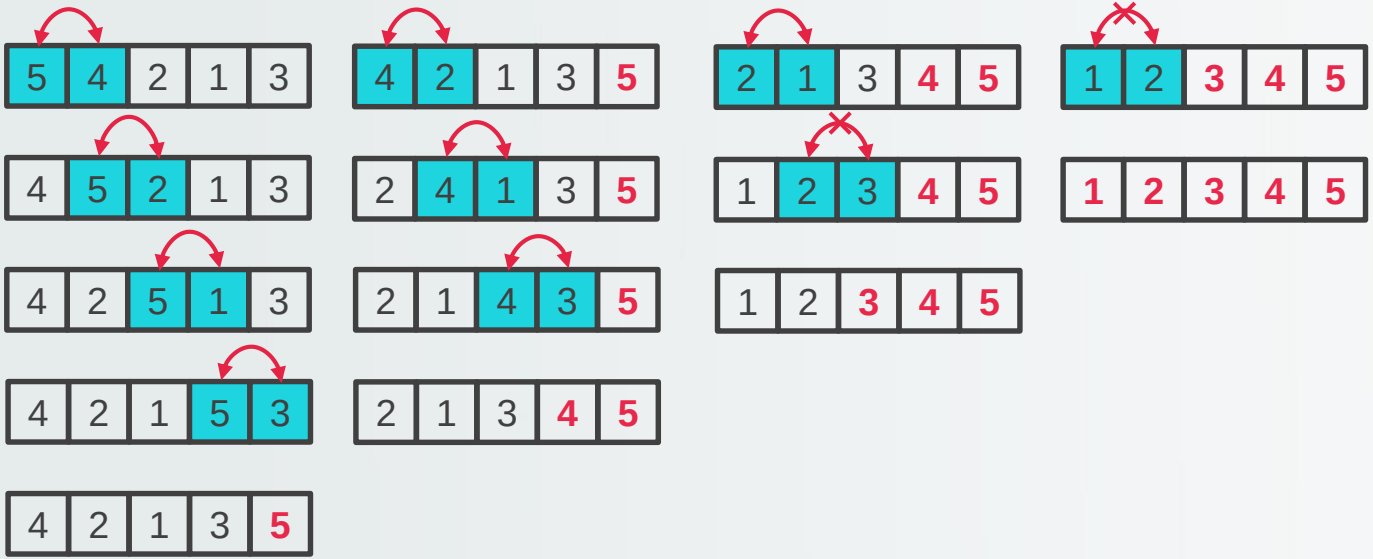
2. for문을 이용한 값 출력

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

배열 실습 1

배열

버블 정렬

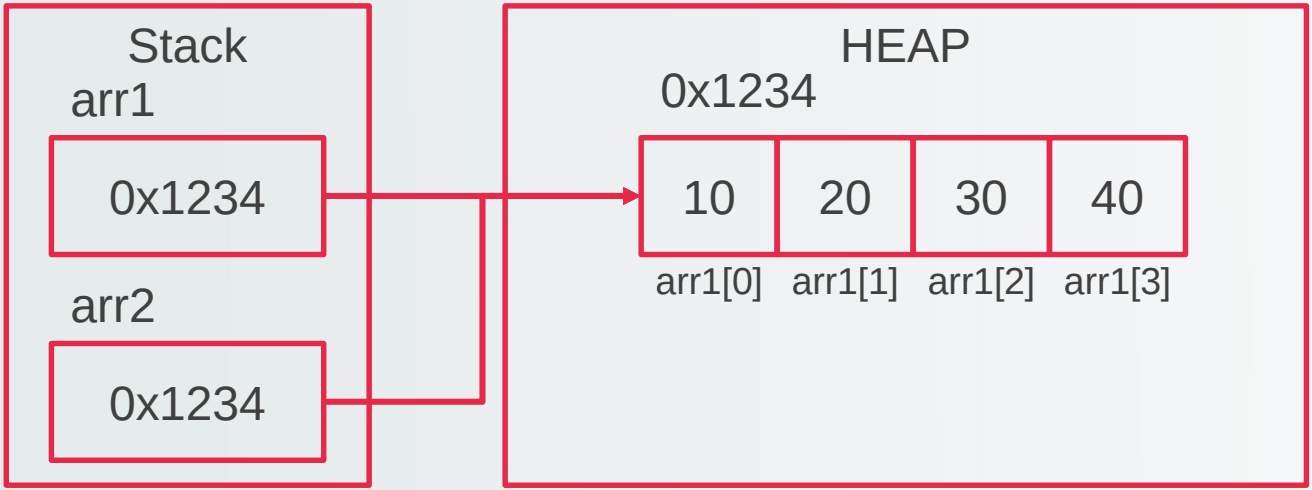

5 4 2 1 3
4 5 2 1 3
4 2 5 1 3
4 2 1 5 3
4 2 1 3 5
4 2 1 3 5
2 4 1 3 5
2 1 4 3 5
2 1 3 4 5
2 1 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

배열 복사

배열 복사

얕은 복사

→ 객체의 주소 값만 가져와 참조형 변수에 저장하고 하나의 객체를 두 변수가 참조하고 있는 것
int [] arr1 = new int[4];
int [] arr2 = arr1; → 배열은 참조형 변수이기 때문에 인덱스를 쓰지않으면 배열의 주소 값을 가져옴



배열 복사

배열 얕은 복사 예

```
int [] arr1 = {1,2,3,4,5};
int [] arr2 = arr1;
System.out.println(arr1.hashcode());          //arr1의 주소값 확인
System.out.println(arr2.hashcode());          //arr2의 주소값 확인
for(int i=0;i<arr1.length;i++){
    System.out.print (arr1[i]+“ ”); //arr1의 배열 내부의 값 확인
}
System.out.println();
for(int i=0;i<arr2.length;i++){
    System.out.print (arr2[i]+“ ”); //arr2의 배열 내부의 값 확인
}
```

배열 복사

배열 얕은 복사 예

```
int [] arr1 = {1,2,3,4,5};
int [] arr2 = arr1;
System.out.println(arr1.hashcode());          //arr1의 주소값 확인
System.out.println(arr2.hashcode());          //arr2의 주소값 확인
arr1[2] = 100;
arr2[4] = 200;
for(int i=0;i<arr1.length;i++){
    System.out.print (arr1[i]+“ ”); //arr1의 배열 내부의 값 확인
}
System.out.println();
for(int i=0;i<arr2.length;i++){
    System.out.print (arr2[i]+“ ”); //arr2의 배열 내부의 값 확인
}
```

arr1과 arr2로 각각 배열 내부의 데이터 값 변경 후 확인

배열 복사

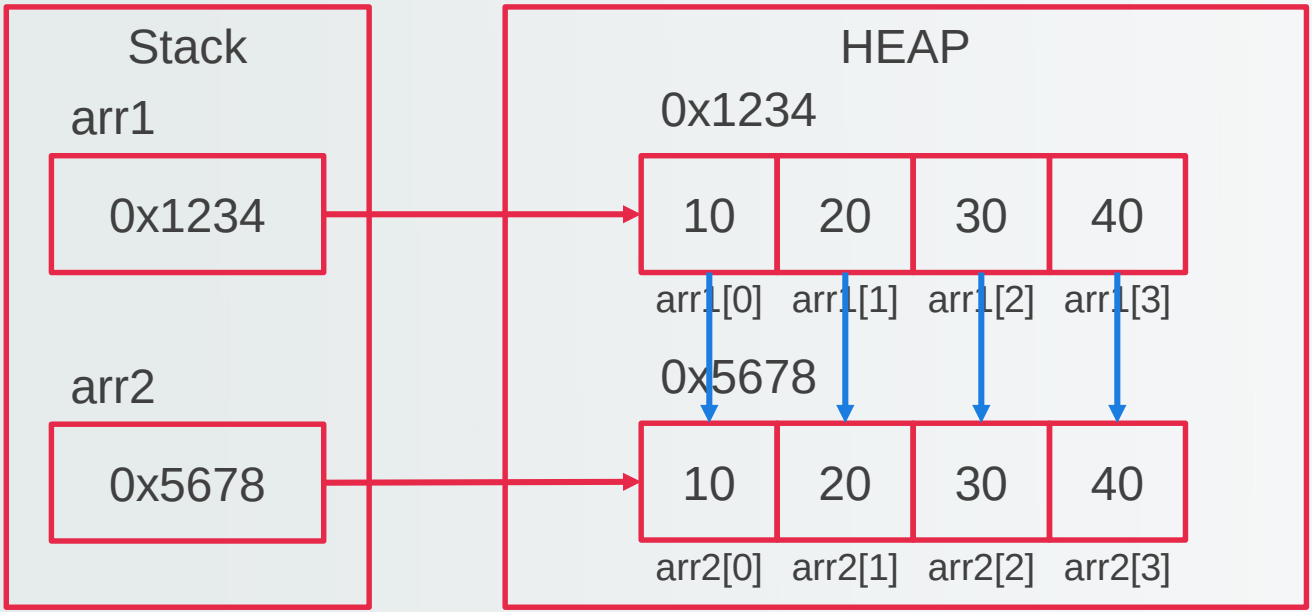
깊은 복사

→ 다른 객체를 생성하여 새로운 객체에 데이터 내용을 복사하는 것
깊은 복사 하는 방법

1. for문을 통한 1:1 값 대입
2. System.arraycopy() 메소드를 이용하는 방법
3. Clone() 메소드를 이용하는 방법

배열 복사

깊은 복사


Stack
arr1
0x1234
arr2
0x5678
HEAP
0x1234
10
20
30
40
arr1[0]
arr1[1]
arr1[2]
arr1[3]
0x5678
10
20
30
40
arr2[0]
arr2[1]
arr2[2]
arr2[3]

배열 복사

배열 깊은 복사 1. for문을 이용한 방법

```
int [] arr1 = {1,2,3,4,5};
int [] arr2 = new int[5];
for(int i=0;i<arr1.length;i++){
    arr2[i] = arr1[i];      //arr2번에 arr1번의 데이터를 그대로 입력
}
```

※ 얕은 복사와 같은 방법으로 주소 값, 실제 데이터, 데이터 변화 확인

배열 복사

배열 깊은 복사 2. arraycopy()메소드를 이용한 복사

```
int [] arr1 = {1,2,3,4,5};
int [] arr2 = new int[5];
System.arraycopy(arr1,0,arr2,0,arr1.length);
```

5가지 값을 입력하여 복사

1. 복사 할 원본 배열 이름
2. 원본 배열 중 복사를 시작할 인덱스 번호
3. 새 데이터가 들어가 배열 이름
4. 복사된 데이터가 들어갈 시작 인덱스 번호
5. 총 복사 할 배열 길이 길이

→ arr1배열의 0번인덱스부터 5개를 복사하여 arr2번 배열의 0번인덱스 부터 넣겠다는 의미

※ 얕은 복사와 같은 방법으로 주소 값, 실제 데이터, 데이터 변화 확인

배열 복사

배열 깊은 복사 3. clone()메소드를 이용한 복사

```
int [] arr1 = {1,2,3,4,5};
int [] arr2 = new int[5];
arr2 = arr1.clone();
```

※ 얕은 복사와 같은 방법으로 주소 값, 실제 데이터, 데이터 변화 확인