제어문

프로그램 수행 흐름을 바꾸는 역할을 하는 제어문 중 하나로 **조건에 따라 다른 문장이 수행되도록 하는 역할** 수행.
쉽게 설명하면 순차적으로 진행해야 하는 코드를 조건을 주어서 임의로 다른 코드가 먼저 실행되게끔 할 수 있다.
조건문에 종류는 두 가지가 있다.

if문

```
if(조건식1) {
        수행될 문장;
} else if(조건식2) {
        수행될 문장;
} else if(조건식3) {
        수행될 문장;
} else {
        수행될 문장;
}
```

switch문

```
switch(조건식) {
case 값1:
        수행될 문장;
        break;
case 값2:
        수행될 문장;
        break;
default:
        수행될 문장;
}
```

조건문의 종류

✓ if

```
if(조건식) {
        ⓐ
}
```

조건식의 결과 값이 true면 ⓐ 안의 내용 실행

false면 실행하지 않음

✓ if문 예시

```
if(num > 0) {
        System.out.println("양수입니다.");
}
```

조건문의 종류

✓ if~else if~else

```
if(조건식1) {
        ⓐ
} else if(조건식2){
        ⓑ
} else {
        ⓒ
}
```

조건식1의 결과 값이 true면

ⓐ 안의 내용 실행

조건식2의 결과 값이 true면

ⓑ 안의 내용 실행

모두 false면

ⓒ 안의 내용 실행

* if는 true, false와 상관 없이 조건절 실행,
if~else if~else는 조건문이 true면 이후 조건은 실행하지 않음

✓ if~else if~else문 예시

```
if(month == 1 || month == 2 || month == 12)
{
        season = "겨울";
} else if(month >= 3 && month <= 5) {
        season = "봄";
} else if(month >= 6 && month <= 8) {
        season = "여름";
} else if(month >= 9 && month <= 11) {
        season = "가을";
} else {
        season = "해당하는 계절이 없습니다.";
}
```

조건문의 종류

▶ switch문

조건식 하나로 많은 경우의 수를
처리할 때 사용하며 이때 조건식의
결과는 정수 또는 문자, 문자열

조건식의 결과 값과 일치하는
case문으로 이동
default문은 일치하는
case문이 없을 때 수행(= else)

자바에서 반복문이란,,,

프로그램 수행 흐름을 바꾸는 역할을 하는 제어문 중 하나로
특정 문장들을 반복해서 수행하도록 하는 역할수행.

쉽게 설명해 순차적으로 진행되야 할 코드를
반복을 주어서 도돌이표처럼 한 코드를 반복 수행할 수 있게 한다.

✓ 반복문의 종류

for문

```
for(초기식; 조건식; 증감식)
{
        수행될 문장;
}
```

while문

```
while(조건식) {
        수행될 문장;
        [증감식 or 분기문];
}
```

반복문의 종류

✓ for문 예시

```
for(int i = 1; i <= 10; i++) {
        System.out.println(i + " 출력");
}
```

✓ 실행 결과

```
1 출력
2 출력
...
9 출력
10 출력
```

반복문의 종류

```
for(초기값1; 조건식1; 증감식1) {
        수행될 문장1;
        for(초기값2; 조건식2; 증감식2) {
                수행될 문장2;
        }
        수행될 문장3;
}
```

반복문의 종류

✓ while문 예시

```
int i = 1;
while(i <= 10) {
        System.out.println(i + " 출력");
        i++;
}
```

✓ 실행 결과

```
1 출력
2 출력
...
9 출력
10 출력
```

반복문의 종류

✓ do ~ while

```
do {
        수행될 문장;
        [증감식 or 분기문];
} while(조건식);
```

반복문의 종류

✓ break문 예시

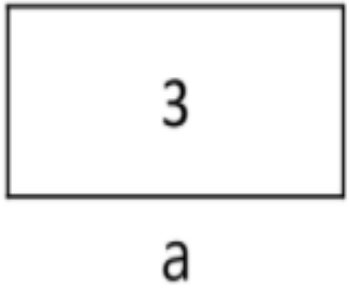
```
for(int i = 1;; i++) {
        System.out.println(i + " 출력");

        if(i >= 10) {
                break;
        }
}
```

✓ continue문 예시

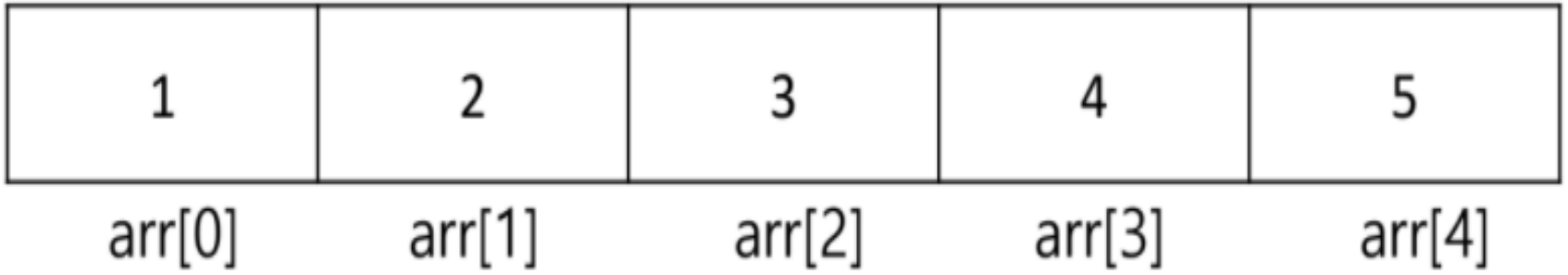
```
for(int i = 1; i <= 10; i++) {
        if(i % 2 == 0) {
                continue;
        }
        System.out.println(i + " 출력");
}
```

배열

변수



배열



배열

✓ 배열 선언

```
자료형[]   배열명 ;
자료형   배열명[ ] ;
```

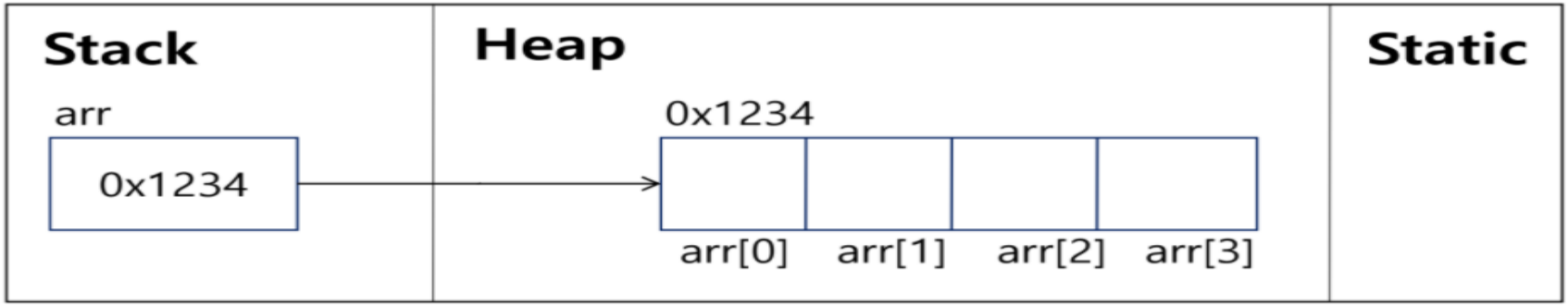
✓ 배열 할당

```
자료형[ ] 배열명 = new 자료형[배열크기];
자료형 배열명[ ] = new 자료형[배열크기] ;
```

```
ex) int[] arr = new int[3];
    int arr[] = new int[3];
```

배열

```
int[] arr = new int[4];
```



배열

✓ 인덱스를 이용한 초기화

ex)
```
arr[0] = 1;
arr[1] = 2;
```

✓ for문을 이용한 초기화

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

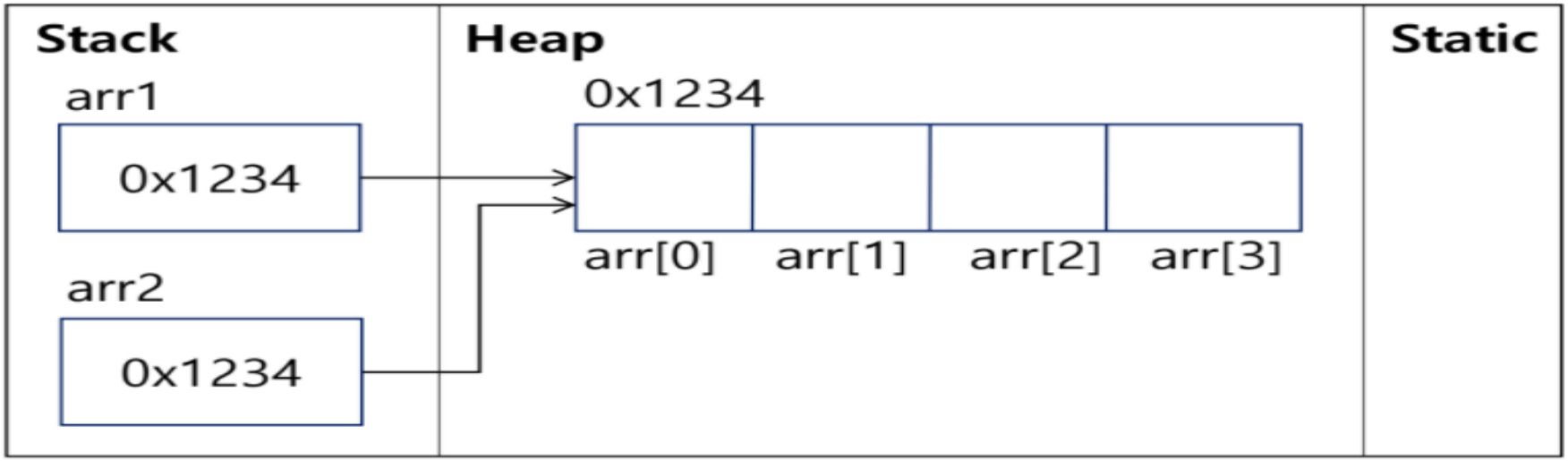
* index가 순차적으로 증가함에 따라 초기화할 리터럴 값이 규칙적이라면 반복문을 통해 배열 초기화 가능

✓ 선언과 동시에 초기화

ex)
```
int[] arr = {1, 2, 3, 4, 5};
int[] arr = new int[] {1, 2, 3, 4, 5};
String fruit[] = {"사과", "포도", "참외"};
```

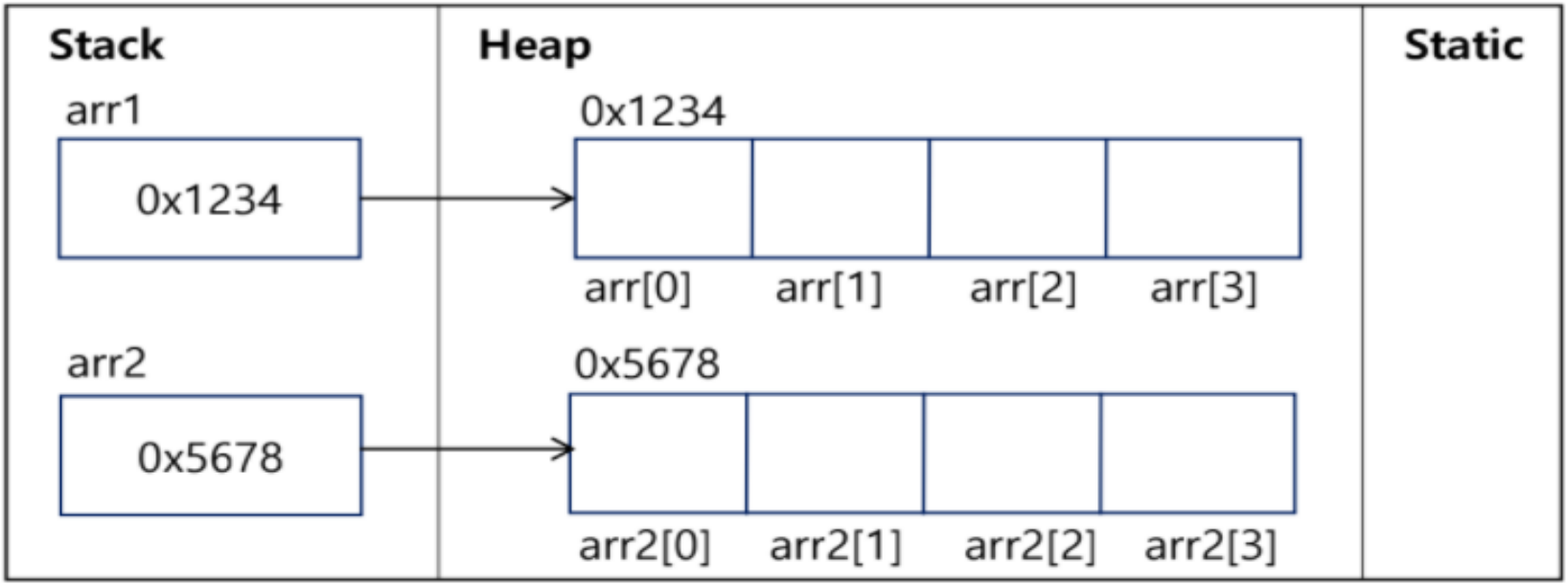
배열

```
int[] arr1 = new int[4];
int[] arr2 = arr1;
```

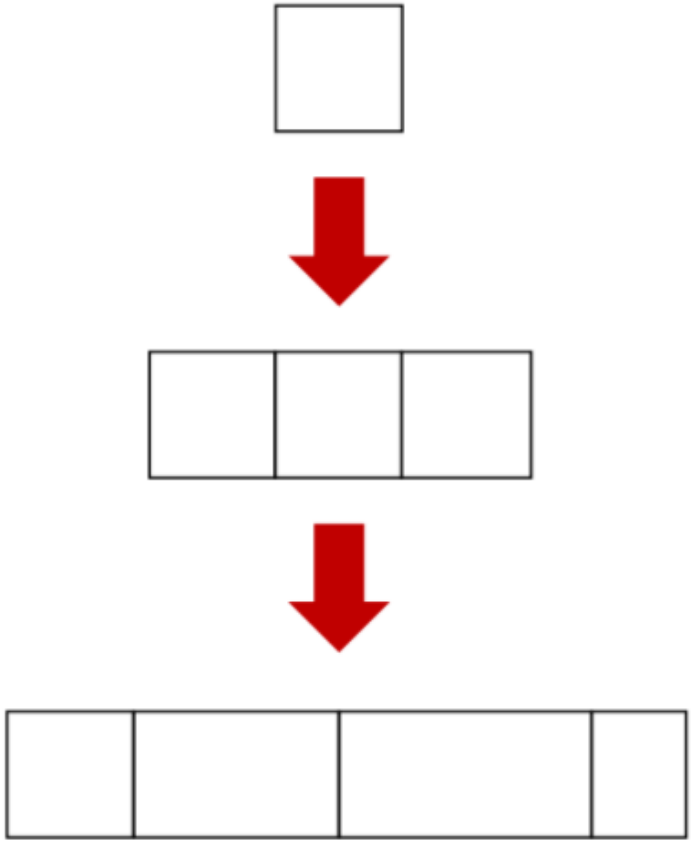


배열

```
for(int i = 0; i < arr1.length; i++) {
        arr2[i] = arr1[i];
}
System.arraycopy(arr1, 0, arr2, 0, arr1.length);
arr2 = Arrays.copyOf(arr1, arr1.length);
arr2 = arr1.clone();
```



배열



변수

1개의 자료형
1개의 데이터

배열

1개의 자료형
여러 개의 데이터

구조체

여러 개의 자료형
여러 개의 데이터

문자열

int length()	문자열의 길이를 반환
char charAt(int i)	문자열에서 i번째 문자를 반환
byte[] getBytes()	현재의 문자열을 바이트 배열로
boolean equals(Object str)	현재의 문자열과 str로 지정된 문자열이 같으면 true, 다르면 false를 반환
boolean equalsIgnoreCase(String str)	현재의 문자열과 str로 지정된 문자열이 같으면 true, 다르면 false를 반환. 단, 비교시 대소문자를 무시
int compareTo(String str)	두 개의 문자열을 비교하여 결과로 양수, 음수, 0의 값을 반환
String trim()	문자열의 앞 뒤 공백(whitespace)을 제거
static String valueOf(double num)	num을 문자열로 변환하여 반환(모든 자료형에 대해 같은 메소드 존재)
static String valueOf(char chars[])	문자 배열을 문자열로 변환하여 반환
String toLowerCase() String toUpperCase()	문자열을 모두 소문자로 변환하여 반환 문자열을 모두 대문자로 변환하여 반환
String substring(int startIndex) String substring(int startIndex, int endIndex	startIndex로부터 시작하는 부분 문자열을 반환 startIndex와 endIndex 사이의 부분 문자열을 반환
String replace(char original, char replacement)	original로 지정된 문자를 replacement로 지정된 문자로 대치