배열을 쓰면 변수 하나에 값을 여러 개 담을 수 있습니다. 자바의 배열을 만들고 사용하는 방법을 간단하게 살펴봅시다.

배열 생성

배열을 만드는 방법은 몇 가지 있습니다.

(1) 선언과 동시에 빈 배열 생성

```
int[] intArray = new int[5]; // 크기 5의 빈 배열
```

(2) 선언 후, 배열 생성

```
int[] intArray;
intArray = new int[5]; // 크기 5의 빈 배열
```

위 두개는 사실 거의 똑같다고 볼 수 있죠?

(3) 리터럴로 생성

```
int[] intArray = {1, 2, 3, 4, 5};
```

이렇게 하면 5개의 원소가 있으니까 **intArray** 는 자동으로 크기 **5** 의 배열이 됩니다.

그런데 이 방식은 변수를 정의할 때만 할 수 있습니다. 밑에 코드처럼 두 줄에 나누어서 하면 오류가 나옵니다.

```
int[] intArray;
intArray = {1, 2, 3, 4, 5}; // 오류
```

배열 사용

이제 배열을 사용하는 방법을 볼까요? 먼저 값을 대입하는 방법을 보겠습니다.

```
intArray[0] = 1;
intArray[1] = 2;
intArray[2] = 3;
intArray[3] = 4;
intArray[4] = 5;
```

이렇게 하면 **0** 번 인덱스에는 정수 **1** 을 넣어주고, **1** 번 인덱스에는 정수 **2** 를 넣어주고. 이런식으로 **intArray** 배열을 채우게 됩니다. 여기서 중요한 점: 인덱스는 **0** 부터 시작합니다! 5칸 짜리 배열이라면 인덱스가 **0** 부터 **4** 까지이겠죠?

아래와 같이 범위에서 벗어나면

```
intArray[5] = 6;
```

런타임에 이런 에러가 나옵니다:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

그러니 배열을 사용할 때는 인덱스를 조심히 다루어야겠죠?

런타임?

런타임이란, 코드 작성 시점이 아니라 실제 실행될 때를 뜻합니다. **intArray[5] = 6;** 을 적었을 때는 오류가 없습니다. 문법적으로 오류가 없기 때문입니다. 그런데 실제 실행을 해서 접근하려고 하면 문제가 생기는 거죠. 강의 후반부에 예외처리를 배울 때 '컴파일 시점', '런타임 시점' 등에 대한 이야기를 더 깊게 나눠 봅시다!

변수명 뒤에 인덱스가 들어간다는 것 말고는 일반적인 변수와 사용법이 같습니다.

값을 대입할 때는 아까처럼:

```
intArray[0] = 1;
intArray[1] = 2;
```

값을 읽을 때는:

```
System.out.println(intArray[0] + intArray[1]); // 1 + 2
```

```
3
```

이렇게 사용할 수 있겠죠?

앨리어싱 (Aliasing)

이렇게 배열을 통째로 넘길 수도 있는데요.

```
int[] arr1 = {1, 2, 3, 4, 5};
int[] arr2 = arr1;
```

급 퀴즈! 아래 코드의 출력값은 뭘까요?

```
arr1[0] = 100;
System.out.println(arr2[0]);
```

```
100
```

100 입니다! **arr1** 을 **arr2** 에 지정해줬을 때, 두 변수는 같은 주소를 가리키게 됩니다. 사실 **arr1** 과 **arr2** 는 아예 똑같은 값인 거죠. **arr2** 를 **arr1** 의 'Alias(가명)'이라고 할 수 있습니다.

그럼 만약 **arr1** 을 **arr2** 에 새롭게 복사하고 싶으면 어떻게 해야할까요?

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

arr1[0] = 100;
System.out.println(arr1[0]);
System.out.println(arr2[0]);
```

```
100
1
```

이렇게 **clone** 라는 메소드를 사용하면 됩니다. 메소드의 사용은 나중에 천천히 배울테니 문법적인 어려움이 있어도 걱정하지 마시고, '이런게 있구나' 정도만 확인하시면 됩니다.

어쨌든 이렇게 하면 배열이 복사가 된 것이라, **arr1** 과 **arr2** 는 서로 다른 배열입니다. 그래서 **arr1[0]** 을 수정해도 **arr2[0]** 에는 영향을 미치지 않고, **1** 이 출력됩니다!

예제 1

100개 짜리 배열을 만들고 첫 번째 칸은 **1** , 두 번째 칸은 **2** , ..., 마지막 칸은 **100** 으로 채워봅시다.

```
int[] intArray = new int[100];
for (int i = 0; i < 100; i++) {
    intArray[i] = i + 1;
}
```

어렵지 않죠? 여기서 코드를 더 직관적으로 바꾸려면 **100** 대신 **intArray.length** 를 쓰면 됩니다. **length** 를 쓰면 우리가 배열의 크기를 몰라도 사용할 수 있고, 코드를 읽는 입장에서도 의미를 쉽게 해석할 수 있어서 좋습니다.

```
int[] intArray = new int[100];
for (int i = 0; i < intArray.length; i++) {
    intArray[i] = i + 1;
}
```

위에서 만든 배열에 값이 제대로 들어갔는지 확인해 봅시다.

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

```
1
2
3
4
...
99
100
```

배열이 아닌 일반 변수를 읽는 것과 거의 똑같죠?

for-each

자바에는 'for-each'라는 문법이 있습니다. 파이썬에도 있었죠? 자바에서 for-each를 쓰는 방법입니다:

```
for (int i : intArray) {
    System.out.println(i);
}
```

이렇게 쓰면, 처음에 수행 부분으로 들어갈 때 **i** 는 **intArray** 의 **0** 번 인덱스의 값(원소)을 갖게 되고, 그 다음 들어갈 때는 **1** 번 인덱스의 값(원소)을 갖게 되고... 이런 식으로 배열의 마지막 값(원소)까지 갖게 됩니다. 직접 확인해보세요!

```
for (double i : intArray) {
    System.out.println(i);
}
```

'형 변환' 강의에서 보셨듯이, **double** 에 **int** 값을 넣으면 자동으로 형 변환이 되기 때문에 위의 식처럼 써도 문제 없겠죠?

예제 2

문자열을 담는 **fruitsArray** 을 만든 후, 원소를 저장하고, **for - each** 문을 활용하여 원소들을 출력해 주었습니다.

```
String[] fruitsArray = new String[5];

fruitsArray[0] = "딸기";
fruitsArray[1] = "당근";
fruitsArray[2] = "수박";
fruitsArray[3] = "참외";
fruitsArray[4] = "메론";

for (String fruit : fruitsArray) {
    System.out.println(fruit);
}
```

딸기
당근
수박
참외
메론

☑ 수업을 완료하셨으면 체크해주세요.

수강생 Q&A 보기 (/questions?assignment_id=268&sort_by=popular)

질문하기 (/questions/new?assignment_id=268&op1=%EA%B0%9D%EC%B2%B4+%EC%A7

이전 강의 배열 (/assignments/250)

다음 강의 배열 연습 (/assignments/275)