Java chapter 3

반복문(while)
반복문(for)
반복문에서의 break와 continue
무한루프(Infinity Roof)
배열(Array)

3-1. 반복문(while문)

반복문은 특정 코드를 반복적으로 실행할 때 사용한다. 자바의 반복문은 while, do while, for문이 있다. do while문은 while문으로 대체가 가능하기 때문에 잘 사용하지는 않는다. 먼저 while문부터 알아보겠다.

이번에 배우는 반복문과 다음에 배우게 되는 배열이 프로그래밍 학습의 첫 고비가 될 것이다. 처음엔 어렵지만 계속 보면 자연스레 익숙해진다!

while문 문법

```
whlie(반복 조건){ ①
    실행내용; ②
    실행내용; ③
    ...
}
```

해석순서

- ①의 조건이 참인지 검사
- 참이라면 while문 안의 실행문(②, ③, …)을 순차적으로 진행
- ①의 조건이 참인지 검사
- 참이라면 while문 안의 실행문(②, ③, …)을 순차적으로 진행
- ①의 조건이 참인지 검사
- 참이라면 while문 안의 실행문(②, ③, …)을 순차적으로 진행
- ①의 조건이 참인지 검사
- 거짓이라면 while을 종료함

while이 성립되기 위해서는 반복의 시작점, 반복 조건, 반복 조건을 파기할 수 있는 코드가 반드시 작성되어야 한다.

아래 코드는 while문의 기본 예시이다.

```
public static void main(String[] args) {
    int num = 1;

    while(num < 5) {
        System.out.println(num);
        num = num + 1;
    }

}
```

반복문이 성립되기 위해서는…

반복의 시작지점

반복 조건(조건이 참일 경우 반복 실행, 조건이 거짓이면 while문 종료)

반복 조건을 파기할 수 있는 코드

3-2. 반복문(for문)

for문은 while문보다 훨씬 많이 사용된다. for문은 while문에서 중요한 3가지(시작점, 반복 조건, 반복 조건을 파기할 수 있는 코드)를 한 곳에 몰아 작성하여 반복 상태를 좀 더 직관적으로 파악하기 위해 만들어졌다. 아래 코드는 while문과 for문의 문법을 비교한 것이다.

while문

```
int i = 0;
while(i < 5){
  System.out.println("A");
  i++;
}
```

for문

```
for(int i = 0 ; i < 5 ; i++){
  System.out.println("A");
}
```

반복의 시작지점

반복 조건(조건이 참일 경우 반복 실행, 조건이 거짓이면 while문 종료)

반복 조건을 파기할 수 있는 코드

for문의 문법은 그리 복잡하지 않다. 단, 실행 순서를 대충 알고 있으면 코드 해석이 어려워진다. 아래 해석 순서는 반드시 알고 있어야 한다.

for문의 기본 문법

```
          ①          ②              ④
for(반복 시작 초기화 ; 반복조건 ; 반복을 중단시키는 코드){
   실행문;
   실행문;  ③
   ...
}
```

실행 순서 : 1 2 3 4 2 3 4 2 3 4 2 3 4 2

위와 같이 1번은 최초 실행 시에만 해석하며, 마지막 해석은 2번이다. 2번을 해석했을 때 조건이 거짓이면 반복을 중단하고 다음 코드를 실행한다.

3-3. break와 continue

반복문(while, for)문에서는 break와 continue 키워드를 사용할 수 있다.

break : 반복문을 벗어남(반복문 종료, 주로 특정 조건을 만족할 때 강제로 반복을 끝낼 때 사용)

continue : 해당 회차만 중지하고 다음 회차를 실행함(continue가 실행되면 반복문의 끝으로 이동하여 계속 실행함)

break 사용 예시

```
for(int i = 0 ; i < 5 ; i++){
  if(i == 3){
    break;
  }
  System.out.println(i);
}
```

```
0
1
2
```

continue사용 예시

```
for(int i = 0 ; i < 5 ; i++){
  if(i == 3){
    continue;
  }
  System.out.println(i);
}
```

```
0
1
2
4
```

3-4. 무한루프

무한루프

무한루프란 반복문을 의도적으로 끝나지 않게 무한히 반복시키는 문법을 말한다.

이러한 무한루프는 반복 횟수가 정해지지 않았을 때 의도적으로 사용한다. 그렇다고 해서 정말 무한히 반복되는 프로그램이 끝나지 않을 것이다.

그렇기 때문에 무한루프는 특정 조건이 만족되었을 때 주로 break문을 통해 반복을 종료시킨다.

무한루프를 사용하는 경우의 예시

키보드로 정수를 입력받되, 짝수를 입력할 때까지 반복적으로 입력을 받을 때.

점수를 키보드로 입력 받는데 0 ~ 100점 사이의 값이 아니면 다시 입력 받고 싶을 때.

while문과 for문에서 무한루프를 사용하는 문법

```
whlie(true){
    실행내용;
}
```

```
for(;;){
    실행내용;
}
```

위와 같이 for문, while문 모두에서 무한루프를 사용할 수 있지만, 주로 while문을 이용한 무한루프가 많이 사용된다.

3-5. 배열(Array)

배열(Array)

- 자료형이 같은 다수의 변수를 효율적으로 관리하기 위한 자료형
- 효율적인 관리란 데이터을 읽고 쓰기 용이하다는 의미
- 우리가 아는 모든 자료형은 모두 배열로 사용이 가능

배열의 사용 목적

int형 변수 10개를 선언하고 사용한다고 해보자. 변수 10개를 타이핑 한다는 것도 문제이고, 10개의 변수에 각각 다른 변수명을 주는 것도 쉬운 일이 아니다. 또한, 10개의 변수명을 기억하는 것도 쉬운 일이 아니다. 더욱이 100개의 변수를 선언했다고 해도, 100개의 변수에 들어간 수를 모두 +1 시킨다고 한다면 어떻게 해야 할까..

배열의 선언과 생성

- 변수를 사용하기 위해선 변수의 선언 및 변수의 초기화 작업이 필요했다.
- 배열 사용을 위해서는 배열의 선언 및 배열의 생성 작업이 필요하다.
- 변수의 선언 및 초기화를 동시에 진행할 수 있는 것처럼 배열의 선언과 생성도 동시에 진행 할 수 있다.

배열 선언과 생성

배열 선언 문법 : 자료형[] 배열명;

```
ex> int[] arr1;                  //정수를 여러개 저장할 수 있는 배열 arr1 선언
    arr1 = new int[3];           //정수를 3개 저장할 수 있는 배열 arr1 생성
    String[] arr2;               //문자열을 여러개 저장할 수 있는 배열 arr2 선언
    arr2 = new String[5];        //문자열을 다섯개 저장할 수 있는 배열 arr2 생성
    double[] arr3                //실수를 여러개 저장할 수 있는 배열 arr3 선언
    arr3 = new double[10];       //실수를 10개 저장할 수 있는 배열 arr3 생성
    boolean[] arr4               //참 또는 거짓을 여러개 저장할 수 있는 배열 arr4 선언
    arr4 = new boolean[3];       //boolean 값 세 개를 저장할 수 있는 배열 arr4 생성
    int[] arr5 = new int[3];     //정수를 세개 저장할 수 있는 배열 arr5 선언 및 생성
    String[] arr6 = new String[5] //문자열 다섯개를 저장할 수 있는 배열 arr6 선언 및 생성
```

위와 같이 배열을 선언만 하면 null값을 갖는다(출력문으로 확인 가능). 자바에서 null은 데이터가 정의되지 않았다는 의미를 나타낸다.

정수형 배열을 생성하면 모든 공간은 0으로 초기화, 실수형 배열을 생성하면 모든 공간이 0.0으로 초기화, 문자열 배열을 생성하면 모든 공간이 null로 초기화된다.

배열 생성 시 선언한 공간의 갯수는 변경 불가능이다.

배열의 또 다른 초기화 방법

```
int[] arr = new int[5];
```

위와 같이 배열을 선언하고 생성하면 정수 5개를 저장할 공간이 만들어지고, 각각 0으로 초기화가 된다. 그렇다면 생성과 동시에 0이 아닌 다른 수로 초기화를 하기 위해선 어떻게 해야 할까.

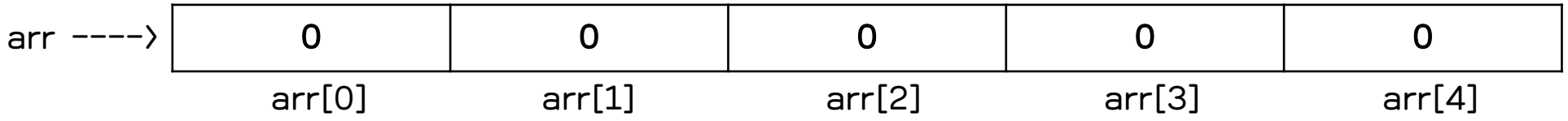
```
int[] arr1 = {1, 2, 3};
String[] arr2 = {“aa”, “bb”, “cc”};
double[] arr3 = {1.1, 4.5, 9.7};
```

이렇게 사용하면 원하는 대로 초기화를 할 수 있으며, 자바가 알아서 필요한 만큼 공간도 생성도 해준다.

배열 생성의 형태

```
int[] arr = new int[5];
```

위 코드에 의해 생성된 배열 arr은 다음과 같이 표현한다.



배열 생성을 통해 만들어진 각각의 공간을 '배열의 요소 ' 라고 부른다.

배열 요소은 0부터 시작한다. (arr[0] -> 배열의 0번째 요소, arr[2] -> 배열의 두번째 요소)

배열의 값 읽기

배열의 값을 읽을 때는 주의가 필요하다. 기본자료형 변수에 저장된 데이터를 읽는 문법과 다르다.

배열 데이터를 읽을 때는 배열 이름이 아닌 배열의 요소를 읽어야한다.

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

배열의 값 쓰기

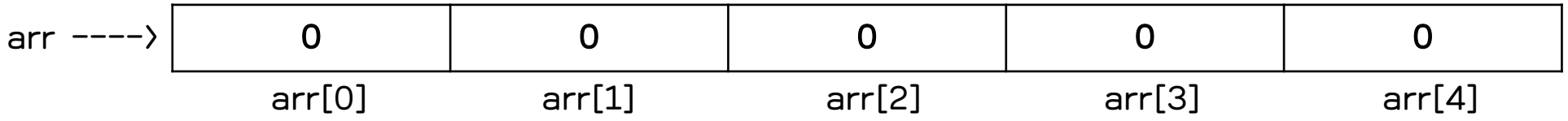
배열의 값을 수정할때에도 배열이름으로 접근하는 것이 아닌, 각 요소로 접근하여 값을 수정해야 한다.

```
int[] arr = new int[3];
arr = 3; x
arr[1] = 3; o
```

배열에 저장된 모든 값 읽기

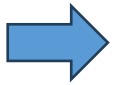
int[] arr = new int[5];

위 코드에 의해 생성된 배열 arr은 다음과 같이 표현하였다.

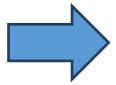


위에서 생성된 배열 arr의 모든 데이터를 출력하는 코드는 아래와 같다.

```
System.out.println(arr[0]);
System.out.println(arr[1]);
System.out.println(arr[2]);
System.out.println(arr[3]);
System.out.println(arr[4]);
```



```
for(int i = 0 ; i < 5 ; i++){
   System.out.println(arr[i]);
}
```



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

배열.length는 생성된 배열의 크기 혹은 길이 값이다.

배열과 기본자료형은 컴퓨터 내부적으로 해석될때 큰 차이점이 있다. 자세한 내용은 클래스를 배울 때 설명하고 이번 시간에는 한 번 짚고 넘어가도록 하겠다.

아래 코드의 실행 결과는?

```
public static void main(String[] args){
    int a = 0;
    int b = a;
    a = 3;
    System.out.println(a);
    System.out.println(b);
}
```

아래 코드의 실행 결과는?

```
public static void main(String[] args){
    int[] a = new int[1];
    int[] b = a;
    a[0] = 3;
    System.out.println(a[0]);
    System.out.println(b[0]);
}
```

기본자료형과 배열은 내부적으로 저장되는 구조가 다르다!