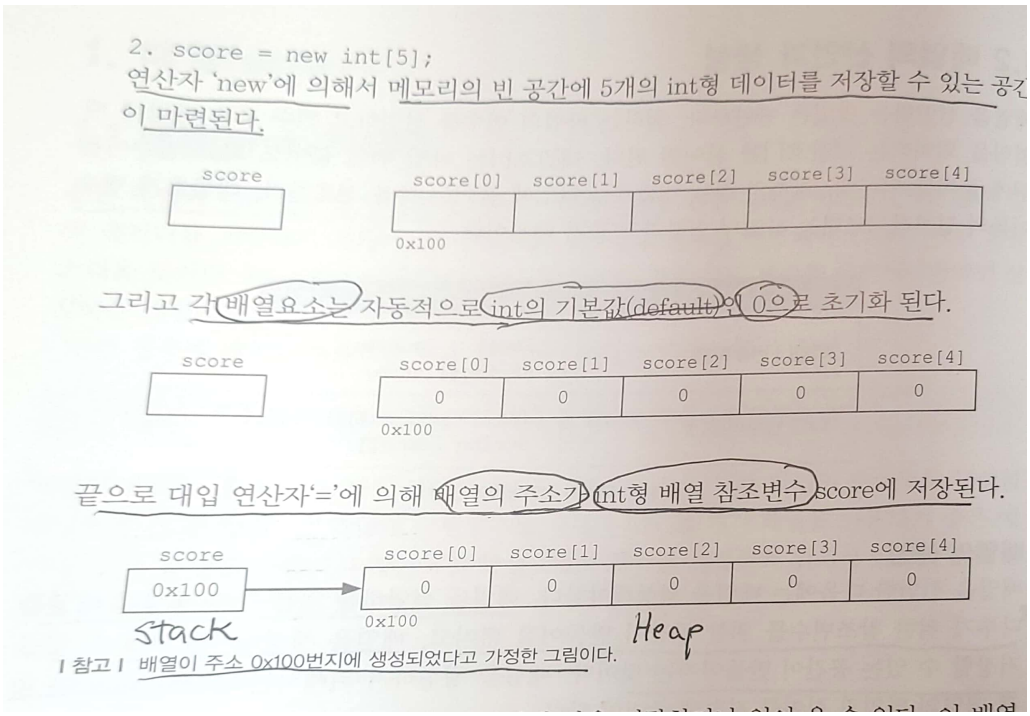
● **배열(array): 같은 타입의 여러 변수를 하나의 묶음으로 다루는 것.** 각 저장공간이 연속적으로 배치되어 있다.

▶ **배열의 선언: 타입[] 참조변수; 배열은 다루기 위한 참조변수를 위한 공간이 만들어진다.** 참조변수를 통해서 배열에 값을 저장하거나 읽어올 수 있다. 이 참조변수는 배열의 이름이다. 타입 참조변수[]; 로도 만들 수 있다. 이때 []는 **배열기호**이다.

▶ **배열의 생성: 참조변수 = new 타입[길이]; 값은 저장할 수 있는 실제 저장공간이 생성된다. 각 저장공간이 연속적으로 배치된다.**




▶ **배열의 요소(element): 생성된 배열의 각 저장공간. 배열이름[인덱스]의 형태로 배열의 요소에 접근**한다.

▶ **배열의 인덱스(index): 배열의 요소마다 붙여진 일련번호**로 각 요소를 구별하는데 사용되며, **범위는 0부터 '배열길이-1'까지**이다. **인덱스로 상수 대신 변수나 수식도 사용할 수 있기 때문에, for문의 제어변수 i를 배열의 인덱스로** 상수대신 사용하는 식으로 사용할 수 있다. 만약 인덱스에 변수를 사용하는 **수식이 포함된 경우, 수식이 먼저 계산**된다. 그래야만 배열의 몇 번째 요소인지 알 수 있기 때문이다. 유효한 범위를 벗어나는 값을 index로 사용하면 컴파일은 정상적으로 되지만 진행 시에 에러가 발생한다(ArrayIndexOutOfBoundsException).

▶ **배열의 길이: 배열의 요소의 개수, 즉 값은 저장할 수 있는 공간의 개수**다. **배열의 길이는 양의 정수(0도 포함)여야 하며 최대값은 int타입의 최대값, 약 20억이다. 길이가 0인 배열의 생성도 가능하다.** 자바에서는 JVM이 모든 배열의 길이를 별도로 관리하며, '배열이름.length'를 통해서 배열 길이에 대한 정보를 얻을 수 있다. 배열은 한번 생성하면 실행하는 동안은 길이를 변경할 수 없기 때문에, 이미 생성된 배열의 길이는 변하지 않는다. 따라서 **'배열이름.length'는 인트형 상수**다. 즉, 값을 읽을 수만 있을 뿐 변경할 수 없다. for문의 조건식에는 배열의 길이를 직접 적어주는 것보다 '배열이름.length'를 사용하는 것이 좋다. 배열의 길이를 변경하였을 경우 자동으로 이를 적용하기 때문이다.
ㄴ 배열의 길이 변경하기: 배열에 저장할 공간이 부족한 경우: 더 큰 배열을 새로 생성한 다음, 기존 배열의 내용을 새로운 배열에 복사하면 된다.

▶ **배열의 초기화: 배열은 생성과 동시에 자동적으로 자신의 타입에 해당하는 기본값으로 초기화**되므로 배열을 사용하기 전에 따로 초기화를 해주지 않아도 되지만, **원하는 값은 저장하려면 각 요소마다 값은 지정**해 줘야 한다. **저장하려는 값에 일정한 규칙이 있다면 for문**을 사용하여도 좋고, 다음의 방법으로 **배열의 선언과 생성과, 초기화를 동시에** 하여도 된다. (**다만 배열의 선언과 생성은 따로 하는 경우에는 new 타입[]을 생략할 수 없다.**) 이 경우 배열의 각 요소는 int의 기본 값인 0으로 초기화되었다가, 다시 주어진 값으로 초기화된다.

```
타입[] 참조변수 = new 타입[] {요소1, 요소2, 요소3...};
타입[] 참조변수 = {요소1, 요소2, 요소3...};

타입[] 참조변수;
참조변수 = new 타입[] {요소1, 요소2, 요소3...};
```

```
//매개변수로 int배열을 받는 메서드가 정의돼 있고, 이 메서드를 호출해야 할 경우 new 타입[]을
//생략할 수 없다.

int add(int[] arr) {...}
int result = add(new int[] {100, 90, 80, 70, 60});
```

▶ **배열의 출력**: 1) for문 사용. 2) Arrays.toString(배열이름) 메서드 사용: **배열의 모든 요소를 '[첫번째 요소, 두번째 요소...]'와 같은 형식의 문자열로 만들어서 반환한다** (import java.utils.*필요)
ㄴ **배열의 이름, 즉 참조변수를 출력하면 타입@주소(16진수 배열의 주소, 실제 주소가 아닌 내부 주소)**가 출력됨.
ㄴ 예외: **char배열은 print()나 println()으로 출력할 시에 각 요소가 구분자 없이 그대로 출력**된다 (배열의 저장된 모든 문자를 출력할 수 있다—print메서드의 기능이다).

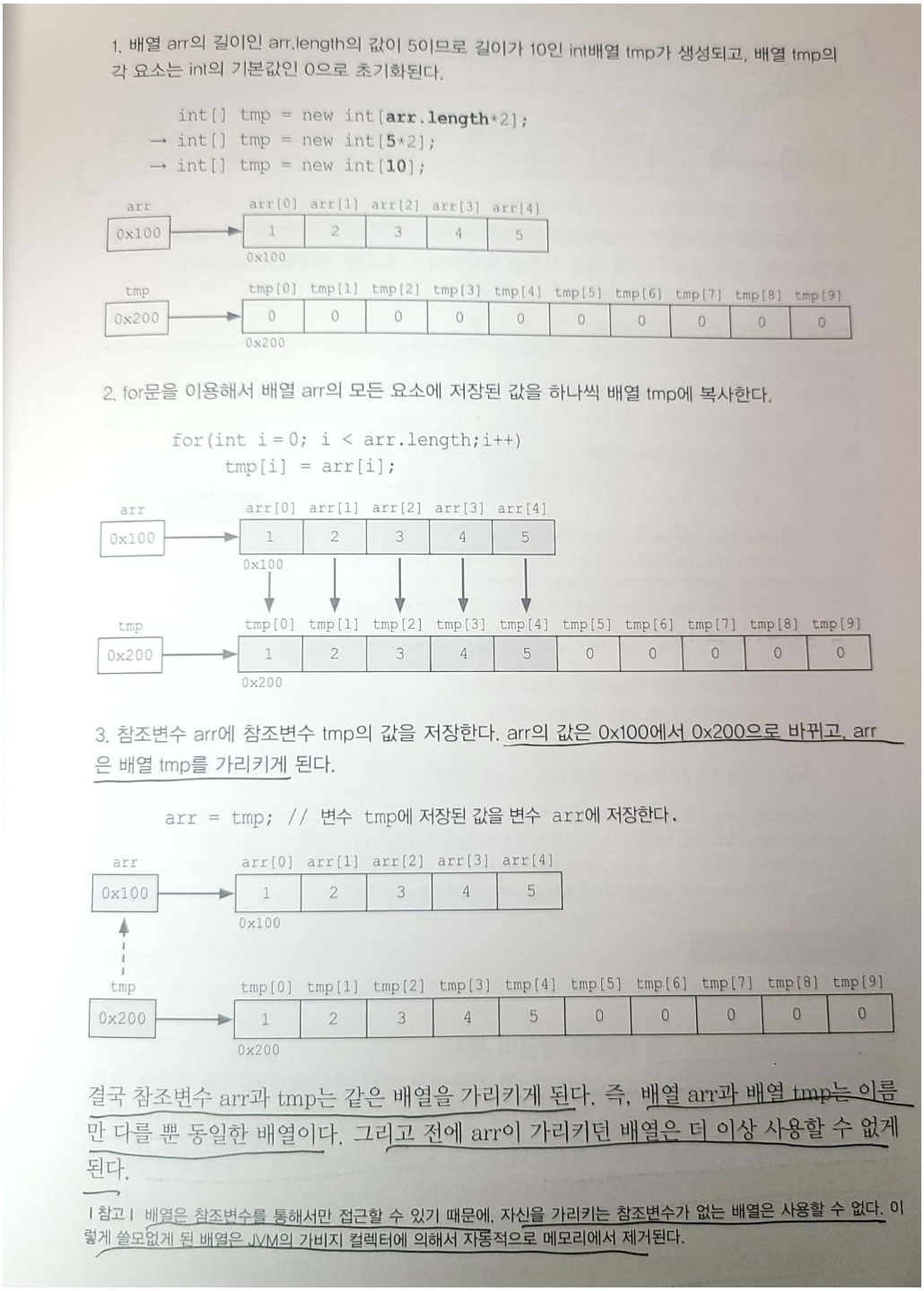
▶ 배열의 복사: 1) for문 사용 2) System.arraycopy() 사용
1) for문:

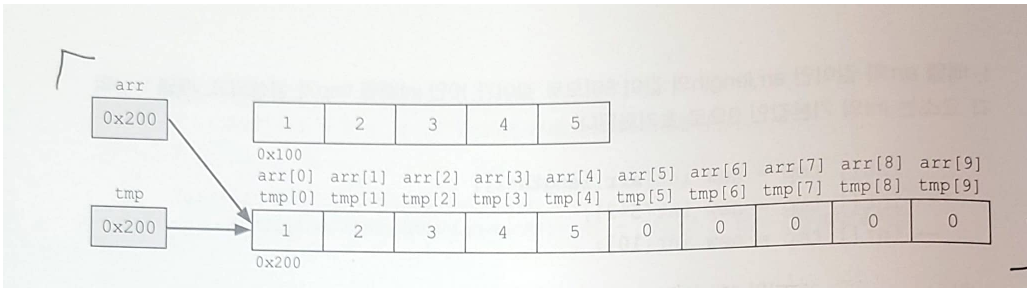
```
int[] arr = new int[5];
int[] temp = new int[arr.length*2];         // 기본 배열보다 길이가 2배인 배열 생성

for (int i = 0; i < arr.length; i++) {
    temp[i] = arr[i];    // arr배열의 요소를 차례대로 돌며 해당 index의 temp배열 요소에 저장

arr = temp;          // 참조변수 arr이 새로운 배열 temp을 가르키게 한다.
                     // 변수 temp에 저장된 값(더 큰 배열의 주소)를 변수 arr에 저장한다.
}
```

결국 참조변수 둘은 같은 배열을 가르키게 된다. 즉, 이름만 다를 뿐 둘은 동일한 배열이다. 그리고 전에 원래 참조변수가 가르키던 배열은 더 이상 사용할 수 없게 된다 (**배열은 참조변수를 통해서만 접근 가능하기 때문에, 자신은 가리키는 참조변수가 없는 배열은 사용할 수 없다. 이렇게 쓸모없게 된 배열은 JVM의 가비지 컬렉터**에 의해서 자동적으로 메모리에서 제거된다).



2) System.arraycopy(): **지정된 범위의 값들을 한 번에 통째로 복사**한다. 배열의 복사는 for문보다 System.arraycopy()를 사용하는 것이 **효율적**이다.

> System.arraycopy(배열이름1, 인덱스넘버, 배열이름2, 인덱스넘버, 몇개의 값)
>
> 배열이름1의 몇 번째 요소에서 배열2의 몇번째 요소부터 몇 개의 값을 복사할 것인지. (배열이름1은 복사하고 싶은 배열, 배열1의 어느 인덱스넘버부터 복사하고 싶은지, 배열이름2는 최종목적지 배열, 배열이름2의 몇 번째 인덱스부터 그 값을 복사하고 싶은지, 몇 개의 데이터를 배열이름1로부터 복사하고 싶은지).

▼ 예제 5-4/ch5/ArrayEx4.java

```
class ArrayEx4 {
    public static void main(String[] args) {
        char[] abc = { 'A', 'B', 'C', 'D'};
        char[] num = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
        System.out.println(abc);
        System.out.println(num);

        // 배열 abc와 num을 붙여서 하나의 배열(result)로 만든다.
        char[] result = new char[abc.length+num.length];
        System.arraycopy(abc, 0, result, 0, abc.length);
        System.arraycopy(num, 0, result, abc.length, num.length);
        System.out.println(result);

        // 배열 abc을 배열 num의 첫 번째 위치부터 배열 abc의 길이만큼 복사
        System.arraycopy(abc, 0, num, 0, abc.length);
        System.out.println(num);

        // number의 인덱스6 위치에 3개를 복사
        System.arraycopy(abc, 0, num, 6, 3);
        System.out.println(num);
    }
}
```

▼ 실행결과

```
ABCD
0123456789
ABCD0123456789
ABCD456789
ABCD45ABC9
```

Resource: 자바의 정석