2025-08-19 회고록

<2025.08.19> Summary & 회고록

전날 복습

1. 자료형 (복습)

- 기본 타입 & 레퍼런스 타입(참조형)
- 기본형에는 값, 레퍼런스형에는 주소 값(위치)이 들어감
- 기본형 8개에 대응하는 레퍼런스 타입 존재 (ex. int ↔ Integer)

1-1. 기본형

타입	크기(byte)	범위
byte	1	-128 ~ 127
short	2	-32,768 ~ 32,767
int	4	-2^31 ~ 2^31-1
long	8	-2^63 ~ 2^63-1
float	4	±1.4E-45 ~ ±3.4E38
double	8	±4.9E-324 ~ ±1.7E308
boolean	1bit	true(1), false(0)
char	2	0 ~ 65535

자주 쓰는 참조형 = String

- String은 객체 ->
  객체의 크기는 가변적이므로 주소만 가짐

---

2. 연산자 (복습)

- 산술연산자: `+`, `-`, `*`, `/`, `%`
- 비교연산자: `==`, `!=`, `<`, `>`, `<=`, `>=`
- 논리연산자: `&&`, `||`, `!`, `^`
- 대입연산자: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
- 증감연산자: `++`, `--` (전위 : 먼저 증가 후 사용, 후위 : 먼저 사용 후 증가)

---

3. 타입 변환 (형 변환) (복습)

* 묵시적 형 변환

- 작은 자료형 -> 큰 자료형은 자동으로 변환

- 정수형 -> 실수형 변환 시 소수 유지
- 산술 승격 : 연산 후 대입 시에도 변환

우선순위:
`byte → short → int → long → float → double → char` == 자료형의 크기 순

* 명시적 형 변환

- 강제로 자료형을 변환해야 할 때 사용

```
int a = 1;
byte b = (int) a; // 명시적 형변환 필요!
```

오늘 배운 내용

반복문의 종류와 구조

1. 반복문

- 반복적으로 일을 처리하기 위한 구문

while 문

기본 구조

```
변수 초기화;
while (조건식) {
    // 조건이 참일 때 실행되는 코드
    변수의 증감식;
}
```

while문과 break

- `break`를 만나면 반복을 즉시 종료
- 보통 if문과 함께 사용

while문과 continue

- `continue`를 만나면 **다음 반복**으로 넘어감 (통과)

---

do / while문 (오늘 배운 내용)

- 반복문 중 하나
- while문과 유사하지만 **무조건 한 번은 실행**됨

기본 구조

```
변수 초기화;
do {
    // 반복 실행할 코드, 무조건 한 번은 실행됨
```

```
    변수의 증감식;
} while (조건식);
```

---

for문 기본 개념

- for문은 반복문(iteration statements) 중 하나
- while문은 변수 선언, 탈출 조건식, 증감식이 3줄로 나뉘지만, for문은 *한 줄에 모두 표현*

```
for (변수의 초기화; 탈출조건식; 증감식) {
    // 탈출 조건식이 참일 경우 실행되는 부분
}
```

중첩반복문

```
public class NestedLoopExample {
    public static void main(String[] args) {
        // 외부 반복
        for (int i = 1; i < 3; i++) {
            // 내부 반복
            for (int j = 1; j < 3; j++) {
                System.out.println(i + "-" + j);
            }
            System.out.println(); // 줄바꿈
        }
    }
}
```

반복문과 label

- label은 중첩 반복문에서 특정 반복문을 제어하고 싶을 때 사용

```
public class LabelExam1 {
    public static void main(String[] args){
        outter:
        for(int i = 0; i < 3; i++){
            for(int k = 0; k < 3; k++){
                if(i == 0 && k == 2)
                    break outter; // 바깥쪽 반복문까지 종료
                System.out.println(i + "," + k);
            }
        }
    }
}
```

문자열과 반복문 활용

- 문자열과 숫자, boolean을 더하면 문자열로 자동 변환

```
public class StringExam1 {
    public static void main(String[] args){
        String str1 = "hello" + 1;
        String str2 = "hello" + true;
        String str3 = "hello" + 50.4;
```

```
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
    }
}
```

배열의 개념과 특징

1. 배열이란?

- **동일한 타입**의 여러 값을 하나의 변수로 관리
- **연속된 메모리 공간**에 순차적으로 저장
- **인덱스(index)**를 통해 각 요소에 접근 (0부터 시작)
- 한 번 생성된 배열의 **크기는 변경 불가**

1-2. 배열 타입별 기본값

타입	기본값	예시
정수형 (byte, short, int, long)	0	`int[] arr = new int[3];` → `{0, 0, 0}`
실수형 (float, double)	0.0	`double[] arr = new double[2];` → `{0.0, 0.0}`
문자형 (char)	`\u0000`	`char[] arr = new char[2];` → `{'\\0', '\\0'}`
논리형 (boolean)	false	`boolean[] arr = new boolean[2];` → `{false, false}`
참조형 (String, Object 등)	null	`String[] arr = new String[2];` → `{null, null}`

1-3. 생성과 초기화

배열 선언 형태

형태	예시
타입[] 변수명	`int[] arr;`
타입 변수명[]	`int arr[];`

배열 생성 방법

구분	문법	예시
선언과 생성 분리	`타입[] 변수명;` `변수명 = new 타입[크기];`	`int[] arr;` `arr = new int[3];`
선언과 생성을 동시에	`타입[] 변수명 = new 타입[크기];`	`int[] arr = new int[3];`

배열 초기화 방법

구분	문법	예시	결과
기본값 초기화	`new 타입[크기]`	`int[] arr = new int[3];`	`{0, 0, 0}`
명시적 초기화	`{값1, 값2, ...}`	`int[] arr = {1, 2, 3};`	`{1, 2, 3}`
new와 함께 초기화	`new 타입[] {값1, 값2, ...}`	`int[] arr = new int[] {1, 2, 3};`	`{1, 2, 3}`

반복문 초기화	`for` 문 이용	`int[] arr = new int[3];`
		`for(int i=0; i<3; i++) {`
		`arr[i] = i+1;`
		`}`
			`{1, 2, 3}`

배열 복사

개념

- 배열은 한 번 생성되면 크기를 변경할 수 없음 → 새로운 배열을 만들어 복사해야 함
- 단순히 대입(`=`)을 하면 **참조(주소)만 복사**됨 → 원본과 복사본이 같은 배열을 가리키게 됨
- 원하는 것은 보통 **값을 복사(깊은 복사)**하는 것임

---

2. 배열 복사의 종류

(1) 얕은 복사

- 배열 변수의 **참조값(주소)만 복사**
- 두 변수가 **같은 배열 객체**를 가리킴
- 하나를 수정하면 다른 배열에도 영향을 줌

```
int[] arr1 = {1, 2, 3};
int[] arr2 = arr1;   // 얕은 복사
arr2[0] = 100;

System.out.println(arr1[0]); // 100 (같이 변경됨)
```

(2) 깊은 복사

- 배열의 **값 자체를 새로운 배열에 복사**하는 방식
- 원본 배열과 복사된 배열은 **서로 독립적**임
- 하나를 변경해도 다른 배열에는 영향을 주지 않음

for문 이용

반복문을 사용해 원본 배열의 값을 하나씩 새로운 배열에 저장

```
int[] arr1 = {1, 2, 3};
int[] arr2 = new int[arr1.length];

for (int i = 0; i < arr1.length; i++) {
```

```
    arr2[i] = arr1[i];
}
```

개인회고

최형규

오늘도 조금 버겁긴 했지만 후에 어제처럼 복습하며 내 것으로 하나씩 만들어 가야겠다.

먼저 공부하기전

1. 개념 이해(기능과 개념을 설명가능한가?
2. 코드 사용이해(작성된 코드 읽어낼 수 있는가?)
3. 코드 사용연습(문제 보고 직접 작성한가?)

나온 순서로 이해도에 따라 필요한 것을 보충해나가야겠다는 생각이 들었다.
이번 주 기본적인 것을 잘 마무리하면 앞으로 더 많이 도움이 될 것 같다.

남혜린

뱁새가 황새 쫓듯 가랑이가 찢어지는 하루였습니다.

이해가 잘 안 되다 보니 타이핑하거나 직접 수기로 작성하면서 따라가는데, 그 사이에 진도는 훌쩍 지나가 버려 동공지진이 왔습니다.

회고 시간에 팀원분들의 다양한 설명을 들었지만, 결국 반복 학습이 답이라는 걸 느꼈습니다. 주말을 잘 활용해 이번 주 진도를 꼭 따라가 보겠습니다.

오늘 배운 내용 중에서는 중첩 반복문과 배열을 특히 집중적으로 복습해야 할 것 같습니다.

백종현

오늘은 반복문과 배열을 배웠습니다.
배열을 공부하면서 제가 처음 접했을 때 어려움을 겪었던 부분이 떠올라, 팀원분들께 조금이나마 도움이 되고자 설명을 드렸습니다. 설명을 하면서 저 스스로도 개념을 다시 복습하는 느낌이 들어 더욱 유익한 시간이었습니다.

또한 오늘 회고 시간에 나눈 대화가 참 좋았습니다. 전공자분들의 경험과 비전공자분들의 질문이 오가며 서로에게 큰 동기부여가 되었고, 제 열정에도 다시 한 번 불이 지펴졌습니다. 그리고 코드너리, 데브이벤트와 같은 정보 공유 플랫폼도 함께 소개하며 서로 성장할 수 있는 좋은 기회가 되었습니다.

앞으로 다 함께 프로젝트를 진행할 날이 무척 기대됩니다!
모두들 화이팅하시고, 롱런을 위해 휴식도 잘 챙기시길 바랍니다!

김도은

오늘 수업에서는 문자열이 단순한 값이 아니라 주소값을 가진 참조형이라는 사실을 알게 되었고, 그래서 배열의 length 속성과는 달리 문자열은 length() 메서드로 길이를 구한다는 점을 이해할 수 있었다.
또한 업캐스팅과 다운캐스팅을 통해 객체지향에서 다형성이 구현되는 원리를 배우면서, 상위 클래스와

하위 클래스 사이의 관계를 조금 더 명확히 grasp할 수 있었다.
마지막으로 로컬 변수의 생명주기와 call by value, call by reference 개념을 공부하며, 메모리 관리 방식과 얕은 복사·깊은 복사의 차이에 대해 깊이 생각해보는 계기가 되었다.

윤수정

오늘은 자기점검을 할 수 있는 날이었다.
개념들은 알지만, 사고력이 많이 부족해진 나를 발견하게 되었고 더욱더 시간을 내어 문제를 많이 풀어보고 내것으로 만들어야겠다는 필요성을 절실히 느끼는 시간이었다.
특히 중첩반복문을 중점으로 개인 실습을 해나가며 앞으로의 과정들을 대비해야 할 것 같다.
이론도 개념도 물론 중요하지만 그 후 우선순위로 해야 할 것은 결국 나의 꾸준하고 성실한 개인공부라고 생각한다.
기본적인 것부터 내 것으로 만드는 연습을 해야겠다.

진솔빈

아... 어제가 불완전연소인줄 알았더니 그게 시작이었다
오늘은 ... 코드를 더 많이 다뤄봐야겠다는 생각 뿐이다
ㅇㅅㅇ

오늘의 할 일:
일단, 너무 스트레스 받아서 운동 좀 1시간
교안 코드 이해해가며 타이핑 해보기
개념정리했던거 정리 -> 옵시디언 정리

백준 문제들 최대한 풀어보겠지만, 풀어보니 개념정리가 우선인거같다. 이게 급선무니 빠르게 풀고, 그래도 답이라도 보고 눈코딩이라도. 맥락을 키워나가자.

무조건 복습 복습 복습 !!

** 1. 백준은 2차원 배열까지 풀 정도가 되고, 솔브드 ac도 그 다음 풀어보자.

1. 이야기를 들어보니 코드를 가능한 가독성있고, 깔끔하게 짜서 인터프리터 성격의 자바 특성에 맞게 정렬을 잘 하는게 중요하다고 느꼈다. 클린코드 관련 유튭강의도 몇개 보기

이환진

어제까지는 따라갈만하다고 생각했다.

근데 오늘은 반복문까지 어떻게든 줄잡다가 중첩 반복문과 배열까지 손코딩하면 진도놓쳐있고 편하기 듣기만하면 코딩쳐보면 이해가될텐데가 생각이난다 그래서 정말 따라가기 힘들었고 수업기억하는내용은 20프로조차 안되는것같다.

회고시간에는 팀원들이 모르는것을 ppt로 자세히 잘설명해주니 좀 알맹이가 겨우 벗겨져서 이해되니 좋았다. 회고시간이 있으니 나에게 도움이 많이되서 정말 다행이라고 생각이든다. 나도 많이 공부해서 팀원들에게 도움줄수있으면 좋겠다.

그래서 오늘은 새벽2시까지 김영한 3시간 강의 듣고 오늘 수업 들었던 교안 손코딩 따라하면서 복습하고 자야겠다.