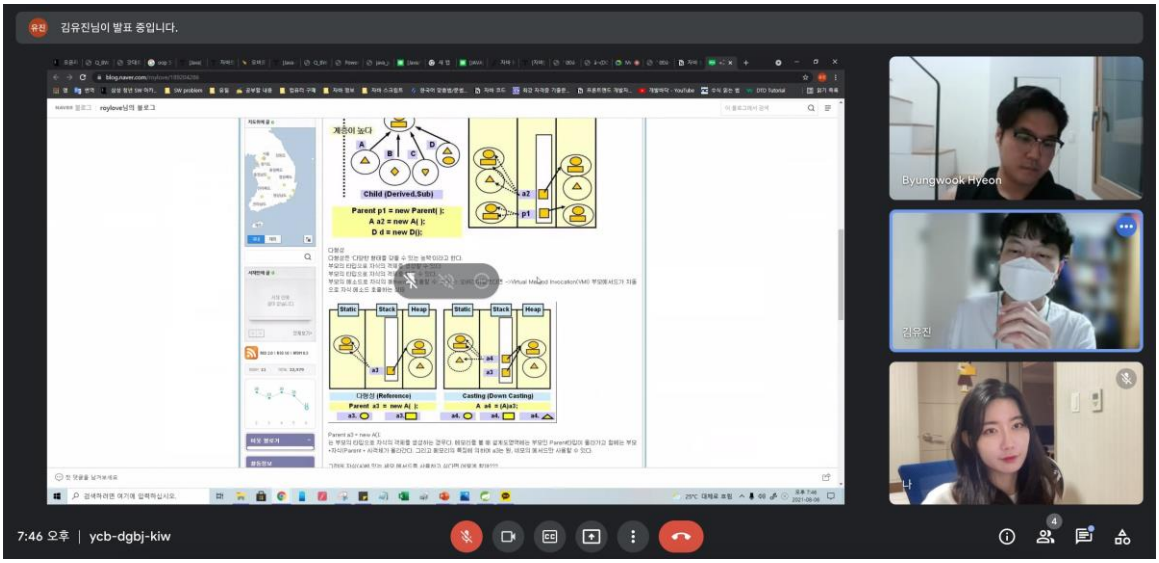
8월 1주차 스터디,5조 최종 결과

조원	김유진, 안예지, 현병욱
발표자	김유진
발표 일자	8월 7월 19시
1주차 스터디 주제	8월 9일 과목 평가 대비
5조 선정 주제	과목평가 중요도 놓은 문제 선정

발표사진

<과목 평가 대비 문제 발표>



최종 종합 결과물

Q1. OOP(Objected Oriented Programming)의 4대 원칙★.

■ **캡슐화 (Encapsulation)**

현실의 객체를 추상화 하여 class로 만들면서 멤버변수와 메서드를 하나로 묶는 과정

■ **추상화 (Abstraction)**

현실의 객체로부터 공통적인 요소나 특징을 추출하는 과정

■ **다형성 (Polymorphism)**

상속관계의 클래스 기능 확장 및 변경을 가능하게 해주는 과정

다형성의 개념을 잘 이용하는 대표적인 방법들 : 오버 로딩, 오버 라이딩

■ **상속성 (Inheritance)**

기존 상위 클래스의 뼈대를 이용하여 새로운 클래스를 만드는 과정

Q2. Primitive Type과 각 type에 대한 bit수

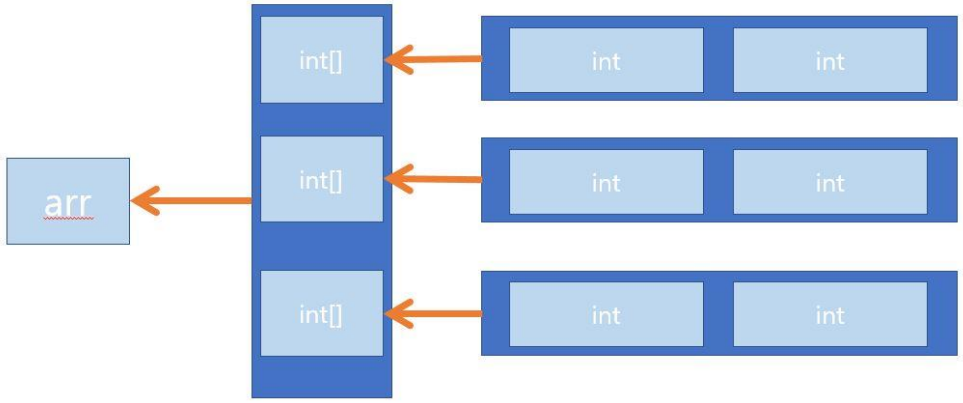
기본형 변수 + 보기 추가

Aa 이름	데이터 유형	바이트(비트)	최소값	최대값	실수 최소 분해능
boolean	논리형	? 알 수 없다.	false	true	
char	문자형	2(16)	'\u0000'	'\uFFFF'	
byte	정수형	1(8)	-128	127	
short	정수형	2(16)	-32,768	32,767	
int	정수형	4(32)	-2^15	2^15 - 1	
long	정수형	8(64)	-2^63	2^63 - 1	
float	실수형	4(32)	-3.4028E38	3.4028E38	1.4E-45
double	실수형	8(64)	-1.7976E308	1.7976E308	4.9E-324

Q3. 연산 우선 순위.

우선순위	연산자	내용
1	(), []	괄호 / 대괄호
2	!, ~, ++, --	부정 / 증감 연산자
3	*, /, %	곱셈 / 나눗셈 연산자
4	+, -	덧셈 / 뺄셈 연산자
5	<<, >>, >>>	비트단위의 쉬프트 연산자
6	<, <=, >, >=	관계 연산자
7	==, !=	
8	&	비트단위의 논리연산자
9	^	
10	|	
11	&&	논리곱 연산자
12	||	논리합 연산자
13	? :	조건 연산자
14	=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, ~=	대입 / 할당 연산자

Q4. int[][] arr = new int[3][2]는 총 몇 개의 객체 배열을 생성하는가? 4개



Q5. 배열의 특징

(1) 배열의 변수는 reference 타입, Heap 영역에 실제 메모리가 할당됨

(2) 배열 할당 시 초기화를 진행하지 않으면 배열에는 java.lang.**NullPointerException**이 **발생한다.**

(3) **배열은 동일한 자료형을** 갖는 값을 저장한다.

Q6. 생성자 this( )와 참조변수 this

생성자 this( )

생성자에서 다른 생성자를 호출할 때 사용

다른 생성자 호출 시 첫 줄에서만 사용 가능

참조변수 this

인스턴스 자신을 가리키는 참조변수

this( )와 연관이 없다.

인트턴스 메서트(생성자 포함)에서 사용 가능

지역변수(주로 매개변수)와 인트턴스 변수를 구별할 때 사용

Q7. 예외와 오버라이딩

- 오버라이딩은 조상 클래스의 메서드보다 많은 수의 예외를 선언 할 수 없다.
- 조금 더 추가 설명을 하면 예외의 상황이 더 적어야 한다.
- NumberException과 Exception은 더 상위의 그리고 더 많은 경우의 예외를 던질 수 있으므로 사용할 수 없다.

Q8. 메서드 오버라이딩 조건

- 메서드 이름이 같아야 한다
- 매개 변수의 개수, 타입, 순서가 같아야 한다.
- 리턴 타입이 같아야 한다.
- 접근 제한자는 부모 보다 범위가 넓거나 같아야 한다.
- 조상 보다 더 큰 예외를 던질 수 없다.

Q9. 객체 생성 제어와 'singleton 디자인패턴'

객체를 구별할 필요가 없는 경우 = 수정 가능한 멤버 변수가 없고 기능만 있는 경우

→ stateless 한 객체라고 한다.

<싱글톤>

```
public class Teacher{
//객체는 나만 만들 수 있어
  private static Teacher t = new Techer();


//기본 생성자의 접근제한자를  private로 외부에서 생성 불가
private Teacher(){
}

//이걸 통해서 가져다 써~
public static Teacher getTeacher(){
  return t;
}

}
```

private의 메소드를 다른 클래스의서 불러오기 위에서는 static

setter는 불필요

```
public static void main(Stirng[] args) {

Teacher t1 = Teacher.getTeacher();
Teacher t2 = Teacher.getTeacher();
}
```

Q10. 다형성

- 하나의 객체가 많은 형(타입)을 가질 수 있는 성질
- 상속관계에 있을 때 조상 클래스의 타입으로 자식 클래스 객체를 레퍼런스 할 수 있다.

```
public class PolyTest1 {

    public static void main(String[] args) {
        SpiderMan onlyOne = new SpiderMan();
        //상속관계에서 조상으로 자식을 참조하는 것 -Polymorphism
        SpiderMan sman = onlyOne;

        Person person = onlyOne;

        Object obj = onlyOne;

        // Venom venom = onlyOne;
    }
}
```

▼ 다형성으로 다른 타입의 데이터를 하나의 배열로 관리

object는 모든 클래스의 조상 어떤 타입의 객체라도 다 저장할 수 있음

Q11. super 키워드

조상 클래스 멤버 접근

super()는 자식 클래스 생성자의 맨 첫 줄에서만 호출 가능

명시적으로 this 또는 super 를 호출 하지 않는 경우 컴파일러가 super 삽입

```
void jump(){
        if(isSpider){
        spider.jump();
}else{
         System.out.println("뛰기");
        }
}
```

```
void jump(){
        if(isSpider){
                spider.jump();
}else{
                super.jump();
        }
}
```

Q12. JVM의 메모리 구조

<Class area>

- Field 정보
- Method 정보
- 타입 정보
- 상수 풀

Method Stack

- thread 별로 별도 관리
- 메서드 호출 순서대로 쌓이는 구조
- 메서드 프레임에 로컬 변수도 쌓이는 구조
- 로컬변수는 선언된 영역을 벗어나면 삭제

Heap

- 객체를 저장하기 위한 영역
- thread에 의해 공유
- 생성된 객체는 프로그래머가 삭제할 수 없고 GC만이 제어가능