CHIP 2학년 수업

프로젝트 관리 및 지켜야 할 사항들

2018년 1학기 4주차

목차

설계

설계란?

- **계획**을 세움, 또는 그 **계획**
- 건축, 토목, 기계 제작 따위에서, 그 목적에 따라 실제적인 계획을 세워 도면 따위로 명시하는 일

- 네이버 사전 참고 – '설계'

개발 환경 관련 설계

- **운영체제나 환경** -> Win10, CentOS, MacOS, AWS Cloud 등
- **개발 언어 및 툴** -> C – VS / C++ - VS Code / Java – InteliJ 등
- **사용할 소프트웨어 버전** -> APM Setup7, MySQL 8.0 등
- **사용할 프레임워크** -> Java의 Spring, Python의 Django 등

개발 환경은 왜 맞춰야 할까?

- **운영체제**마다 작동방법이 다르기 때문
- **소프트웨어 버전**마다 지원안되는 기능이 있을 수도 있음
- **프레임워크는 소스코드의 흐름**을 바꾸기 때문

구현 관련 설계

- **디자인 패턴** -> 3학년 1학기 소프트웨어 공학에서 자세히 배움
  개별적으로 공부하자면 MVC, MVP, MVVC 정도..?

- **클래스 정의**
- **함수 정의**
- **변수 정의**

-> **당장 2학기부터 중요함!**

구현 설계를 왜 해야할까?

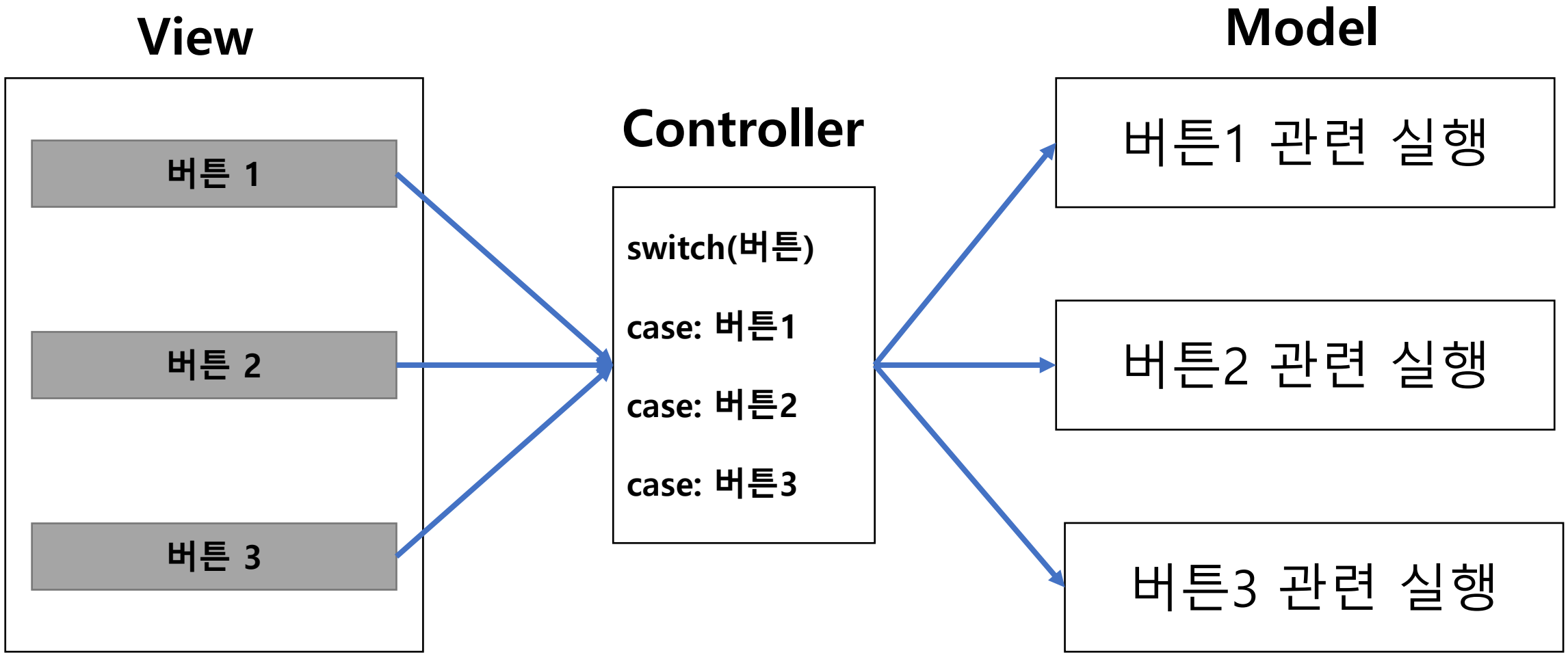
- **Code Coverage** 를 높이기 위해
- 우아한 코드(**Graceful Code**)를 위해
- 팀원들간의 **Code Conflict**를 막기 위해

디자인 패턴

- 소스 코드를 구성하는 형태 혹은 패턴

- **MVC 패턴의 예를 들어보자**
  - M – Model : 프로그램이 실질적으로 돌아가는 것을 말함
  - V – View : 프로그램이 보이는 곳을 말함 (Button, TextView 등등)
  - C - Controller : 프로그램을 어떻게 돌릴 것인가 분기하는 곳

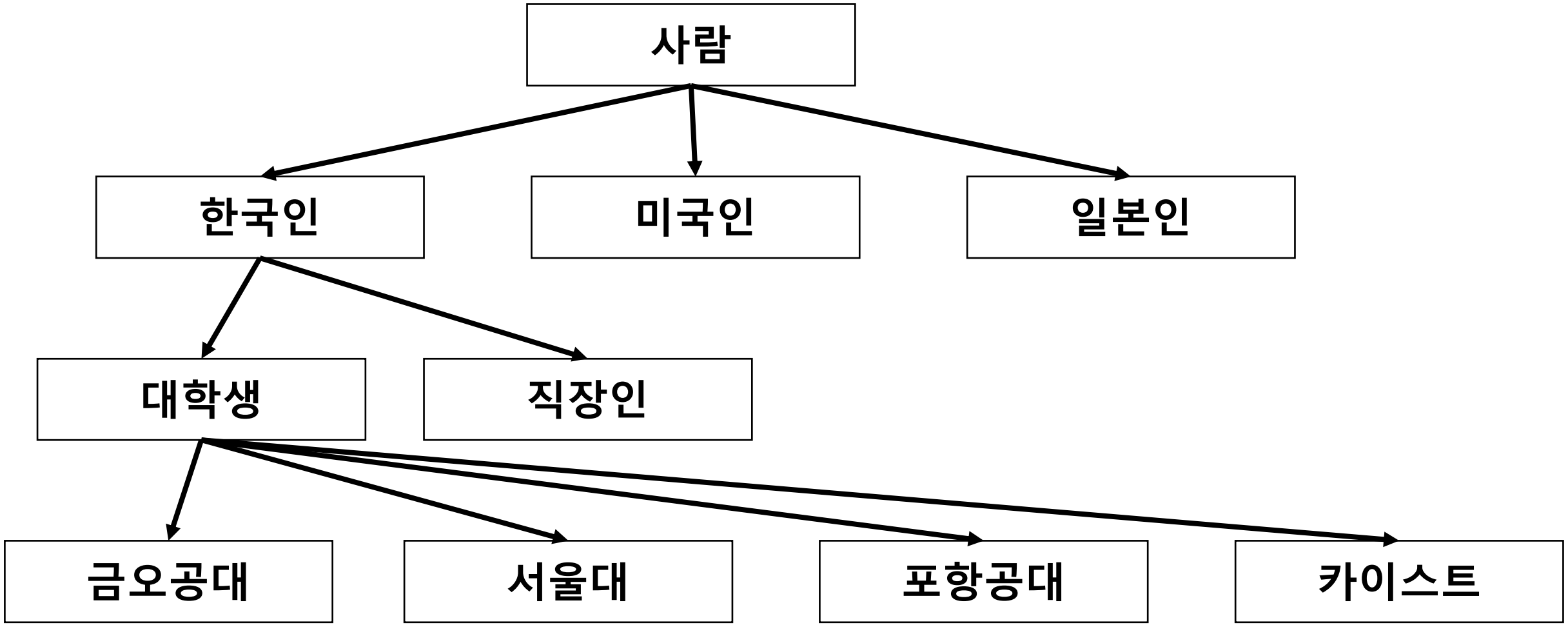
디자인 패턴 – MVC


View
버튼 1
버튼 2
버튼 3
Controller
switch(버튼)
case: 버튼1
case: 버튼2
case: 버튼3
Model
버튼1 관련 실행
버튼2 관련 실행
버튼3 관련 실행

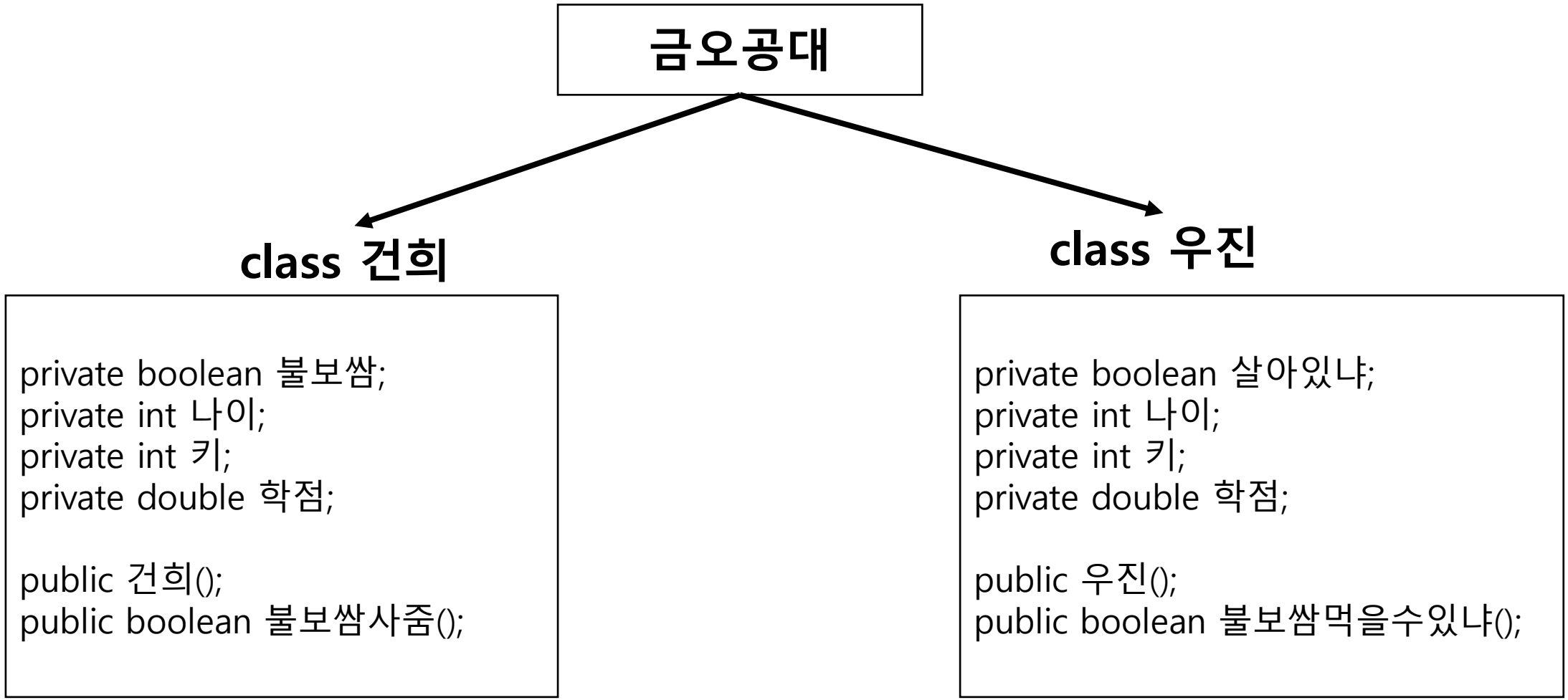
클래스 정의

- 클래스 간의 **관계 정의**
- 클래스의 **역할**을 정의
- 클래스 **내부 함수** 정의
- 클래스 **내부 변수** 정의

클래스 정의 - 예시



클래스 정의 - 예시

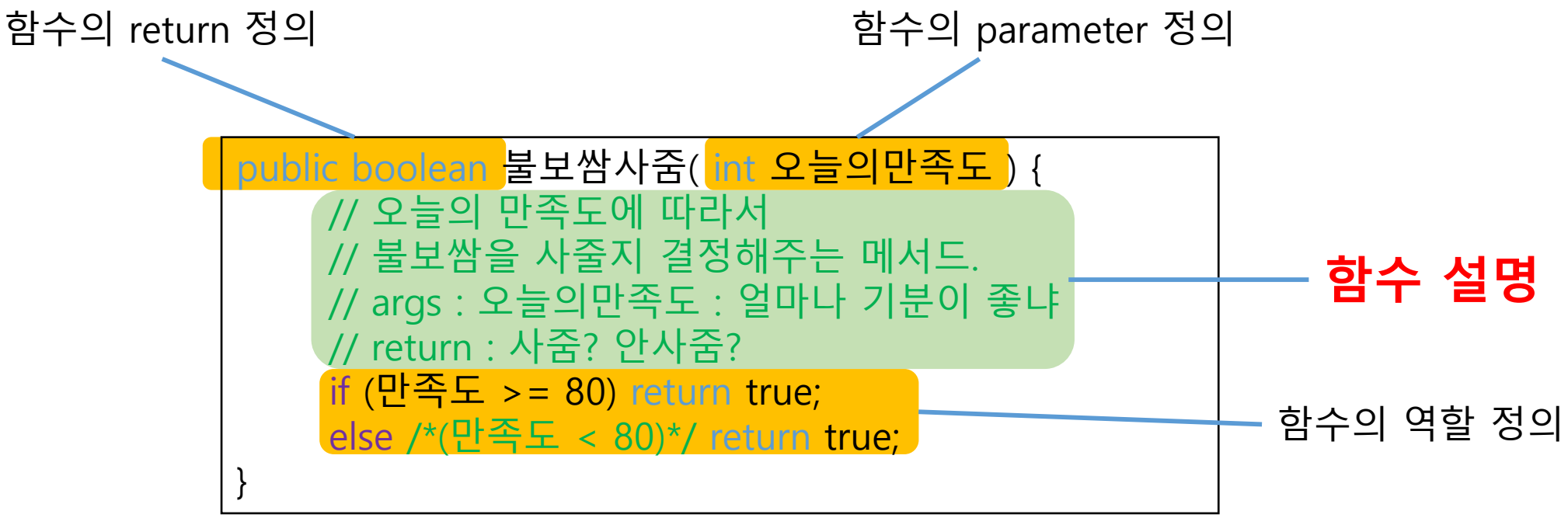

금오공대
class 건희
private boolean 불보쌈;
private int 나이;
private int 키;
private double 학점;
public 건희();
public boolean 불보쌈사줌();
class 우진
private boolean 살아있냐;
private int 나이;
private int 키;
private double 학점;
public 우진();
public boolean 불보쌈먹을수있냐();

함수 정의

- 어떤 **역할**을 하는 함수인가?
- 어떤 **Parameter**를 쓰는가?
- 어떤 값을 **Return** 하는가?

함수 정의 – 예시

- 건희 class의 불보쌈사줌 메서드 정의



변수 정의

- 어떤 것을 **가리키는** 것인가?
- 어떤 **속성**을 나타는 것인가?

각 컴포넌트별 네임 규칙

- 프로젝트를 진행함에 따라서 **무작위의 이름을 쓰는 것을 방지**
- 한 눈에 **무슨 역할을 하는지** 알기 쉽게 정할 것
- 가급적이면 상수를 쓰는 행위는 하지 말 것
  (하더라도 #define이나 final형으로 정의하고 쓰자)

네이밍 정의 방법

- 클래스명 : 첫 문자는 대문자로 써주자

- 함수명 : (동사)+(목적어) or (명사)+(동사)
  : 띄어쓰기가 필요한 경우 '_' 혹은 대문자를 쓰자

- 변수명 : (명사)_(명사) or (명사)
  : 띄어쓰기는 '_'를 쓰고, 소문자로 표기

- 상수명 : 대문자의 명사로 표기

네이밍 정의 방법 – 예시 (Java 스타일)

```
public class Keonhee {
    private int money;
    private boolean bool_bossam;
    private int key;

    public boolean getBool_bossam(int status);
    public int printMoney();
};
```

네이밍 정의 방법 – 예시 (C++ 스타일)

```
class Keonhee {
private:
    int money;
    boolean bool_bossam;
    int key;

public:
    boolean get_bool_bossam(int status);
    int print_money();
};
```

구현 및 테스트

구현은 어떻게 하면 효율적일까?

- **작은 컴포넌트 -> 큰 컴포넌트로** 구현
- 함수나 클래스를 구현하면서 **테스트 케이스**를 놓자
- **역할군 혹은 클래스** 별로 파일을 나누자 - **모듈화**

왜 구현을 작은 단위부터 할까?

- 큰 단위의 계획은 설계에서 했다.
- 작은 곳부터 **에러나 버그없이**해야 한다.
- **역할군을 명확**하게 하기 위해서

컴포넌트가 구현되면?

- 반드시 **테스트**를 진행해야 한다.
  - 안하게 되면 추후에 버그가 겉잡을 수 없을 정도로 터진다.
  - **테스트 케이스**를 반드시 만들자. (printf 등을 적극적으로 활용)
  - 주석이나 문서로 **반드시 설명을 적어주자.**
    - Contributing이나 프로젝트 진행에 따라서 매우 중요함

왜 파일을 나누는 것일까?

- **효율적인 소스코드 관리를 위해서**
  - 팀원이 만약 main함수에 소스코드를 모두 작성했다면 너의 기분은?
  - 팀원이 1000줄이 넘는 소스코드를 인계 해줬다면?

왜 파일을 나누는 것일까?

- **효율적**
  - 팀원
  - 팀원

다면 너의 기분은?

?

각 컴포넌트 테스트 및 구현이 끝나면?

- 팀원들끼리 컴포넌트를 합친다.
- 합친 후의 코드를 작성한다.
- 테스트 및 테스트 케이스를 작성한다.
- **만약 팀원들 간의 네이밍 규칙이나 정의를 안해놨다면?**

각 컴포넌트 테스트 및 구현이 끝나면?

- 팀원들끼리 컴
- 합친 후의 코드
- 테스트 및 테스
- **만약 팀원들 간의 안해봤다면?**

테스트 케이스는 왜 작성해야죠?

- **버그가 터지는지 안터지는지** 어떻게 알죠?
- 어떻게 이 함수나 변수를 사용해야죠?
- **즉, 예제를 작성하는 것이 테스트 케이스**

만약 테스트가 안된 코드를 합친다면?

FRODO

만약 테스트가 안된 코드를 합친다면?

Github

Github 의 기능들

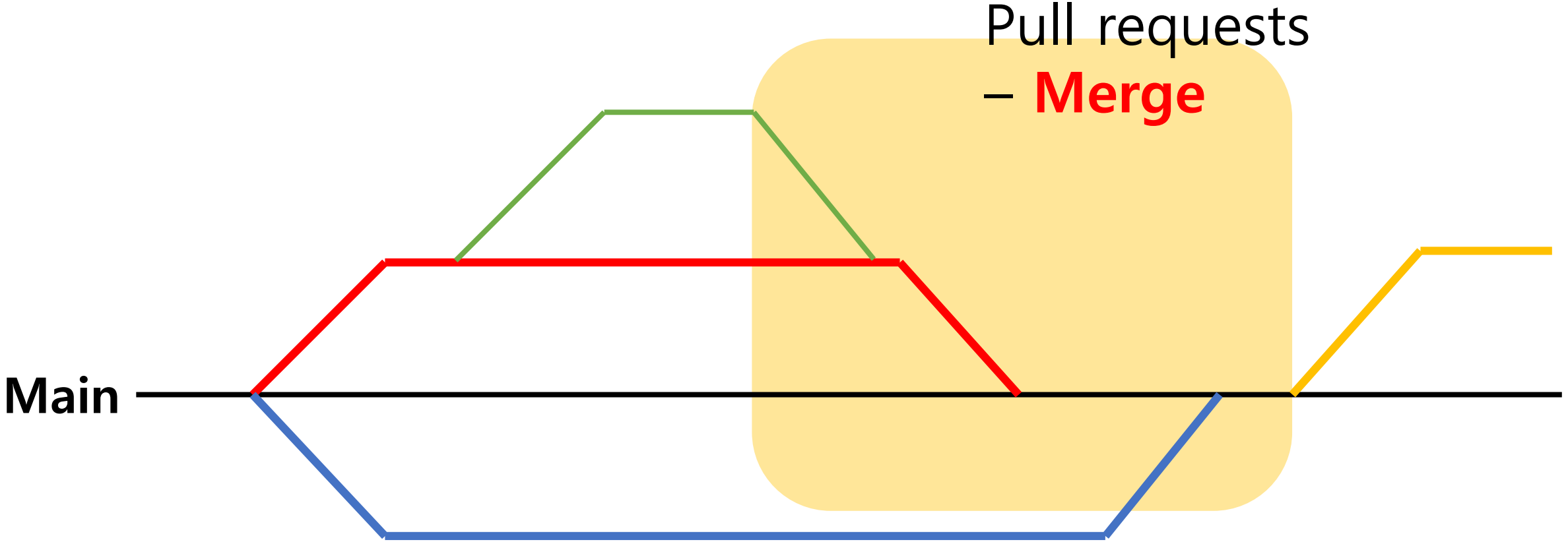
- 소스 코드 저장
- 소스 코드 버전별 정리 - tag / version
- 소스 코드 분기별 관리 - branch

- 프로젝트 전반적인 관리
- 해당 프로젝트에 이슈 남기기 기능 – Issue 카테고리
- 소스코드 수정 건의 기능 – Pull requests 카테고리
- 소스코드 문서화

branch

- **나뭇가지, 나뭇가지 모양의 물건**
- **Subversion**
- 소스코드 **분기**

branch 활용 예제



이게 그럼 중요할까?

실전에서 매우 중요하다

- 설계 과정이 제대로 안되면 구현에서 문제가 나온다.
- 이전 버전 관리를 안해놓으면 처음부터 다시 짤지도 모른다.
- 안하면 소스코드가 미친듯이 꼬인다.

Github는 왜 알려주었는가?

- **프로젝트 관리를 메인으로 하는 가장 유명한 툴**이다.
- 잘 사용하면 이만큼 편한 툴이 없다.
  (특히 리눅스, Mac환경에선 매우 편리함)
- 취업 준비할 때 이만큼 좋은 포트폴리오가 없다.

그 외에 알아둬야할 점

- 창설 혹은 팀 단위의 경진대회 같은데에서
  싸우기 싫으면 알아두는게 가장 좋음.

- 디자인 패턴은 전부 다 알필욘 없다.
  하지만 실전에서 자주 쓰이는 패턴 정도는 알아두자.(**특히 MVC**)

- 실제로 개발자들이 중요하게 생각하는 과목
  - 운영체제, 네트워크, 소프트웨어공학, 데이터베이스, **자료구조**

그 외에 팀 프로젝트할때 쓸만한 것

- **Goorm IDE** - 소스코드 공유 및 실시간 작성 툴
- **AWS Cloud** - 원격 서버. 실제로 배그도 여기에 서버 둠
- **PuTTy, 팀 뷰어 –** 원격 제어 프로그램
- **Eclipse Coverage 기능 –** 소스코드 활용도 체크 기능

컴공에게 중요한 것

1. **버그 없는 것**
2. **버그 없애는 것**
3. **버그 없이 잘 돌아가는 것**