



개발 스터디 자바기초 #1



자바 스터디 운영을 위해 작성하는 문서입니다.

2022.06.04 (토) 작성완료

작성자 : daria ([git](#)) ([velog](#))

▼ 자바란? 😎

- 자바(Java)는 1995년 썬 마이크로시스템즈에서 발표한 객체 지향 프로그래밍 언어 (현재는 오라클)
- **"Programmers write once, run anywhere(WORA)"**
한번 작성한 코드를 모든 플랫폼에서 작동 시킬 수 있는 범용적인 언어
- 대한민국 전자정부표준 프레임워크는 Java 프레임워크인 Spring

▼ 자바 코드 예제

```
class HelloJava {  
    public static void main(String[] args) {  
        System.out.println("Hello Java!!!");  
    }  
}
```

클래스 이름

메인 메소드

본문(Body)

클래스 이름: 저장 파일 이름

메인 메소드: 자바 애플리케이션이 실행되면 가장먼저 실행되는 부분

본문: 자바 문법에 맞게 코드를 작성

▼ 자바동작 원리

1. 자바 코드 작성 ([HelloWorld.java](#))

```
public class HellowJava {
    public static void main(String[] args){
        System.out.println("Hello Java!");
    }
}
```

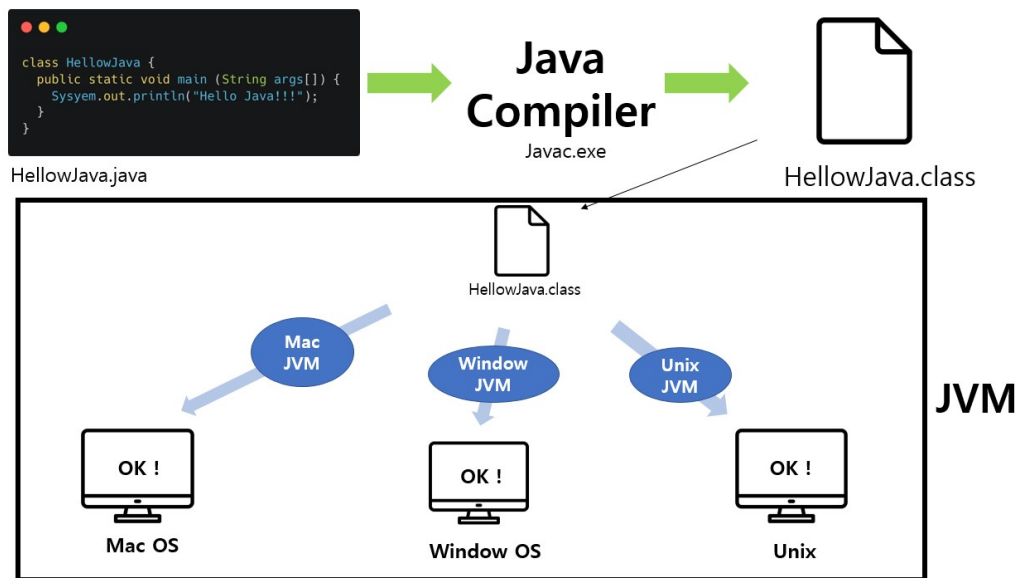
2. 컴파일러를 통해(javac) 컴파일 -바이트코드(반기계어)인 ooo.class 파일 생성

```
> javac HellowJava.java
```

3. JVM을 통해 실행 -컴퓨터가 해석할 수 있는 바이너리 코드로 변경

```
> java HellowJava

Hello World!
```



▼ 자바의 5개념

- **Objects(객체)** : 데이터, 행위, 아이덴티티를 가지고 있는 것
- **Classes(클래스)** : 객체를 생성하는 청사진
- **Encapsulation(캡슐화)** : 행위와 상태를 포장하고 외부에 노출할 것과 감출 것을 결정하는 것. 외부에 노출되는 모든것을 Interface라고 할 수 있음
- **Inheritance(상속)** : 일종의 '가족관계' 표현. 자식은 부모의 자산을 이용할 수 있으며, 코드 재사용이 첫 번째 달성되는 것. 상속은 클래스의 계층 구조를 표현할 수 있게 되는데, 계층 구조는 그 구조 자체만으로는 수많은 정보를 포함할 수 있음. 계층구조를 표현하는 것이 두번

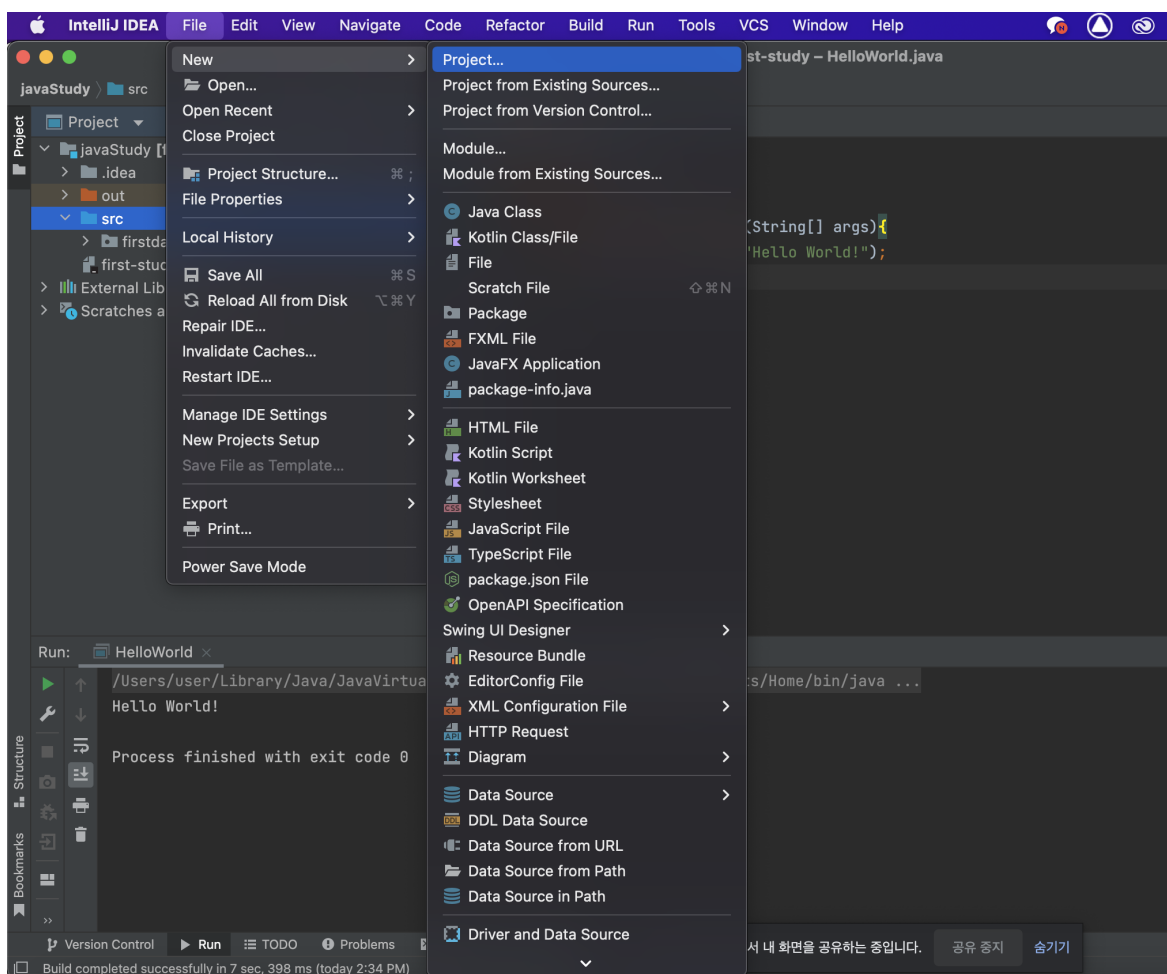
때 핵심. 계층구조를 포함할 목적이 아니라면 상속을 사용해서는 안되며, 코드 재사용보다 계층 구조의 표현이 상속에서 더욱 중요한 개념.

- **Polymorphism(다형성)** : 캡슐화, 상속과 함께 동작함으로써 객체지향 프로그램의 흐름제어를 객체로 처리하도록 단순화 하는 것.

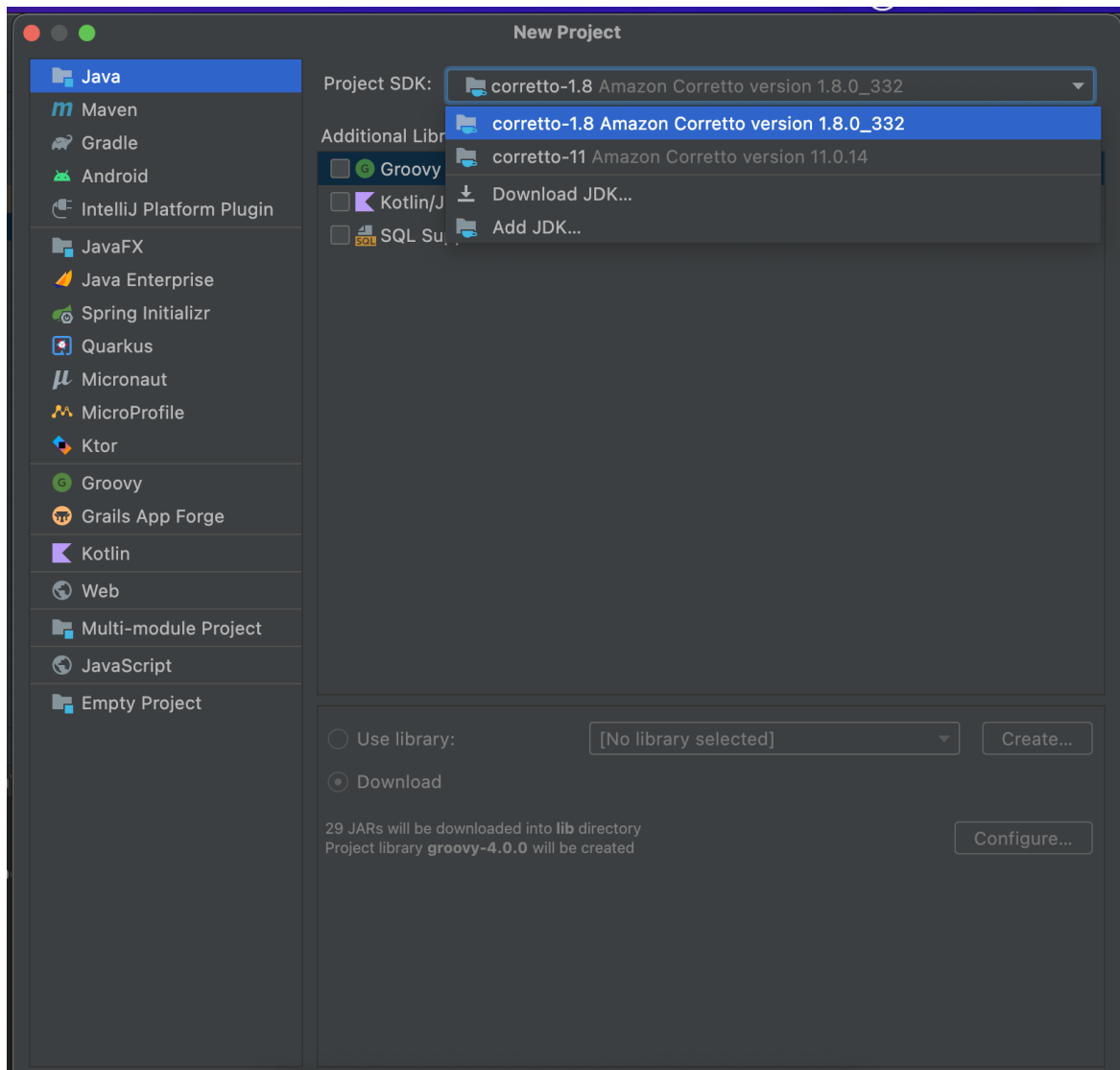
▼ 자바의 5원칙

- **SRP(the Single Responsibility Principle)** : 단일 책임의 원칙
- **OCP(the Open Closed Principle)** : 개방 폐쇄의 원칙
- **LSP(the Liskov Substitution Principle)** : 리스코프 치환의 원칙
- **ISP(the Interface Segregation Principle)** : 인터페이스 분리의 원칙
- **DIP(the Dependency Inversion Principle)** : 의존 역전 원칙
- 참고 : [\(링크1\)](#) [\(링크2\)](#)

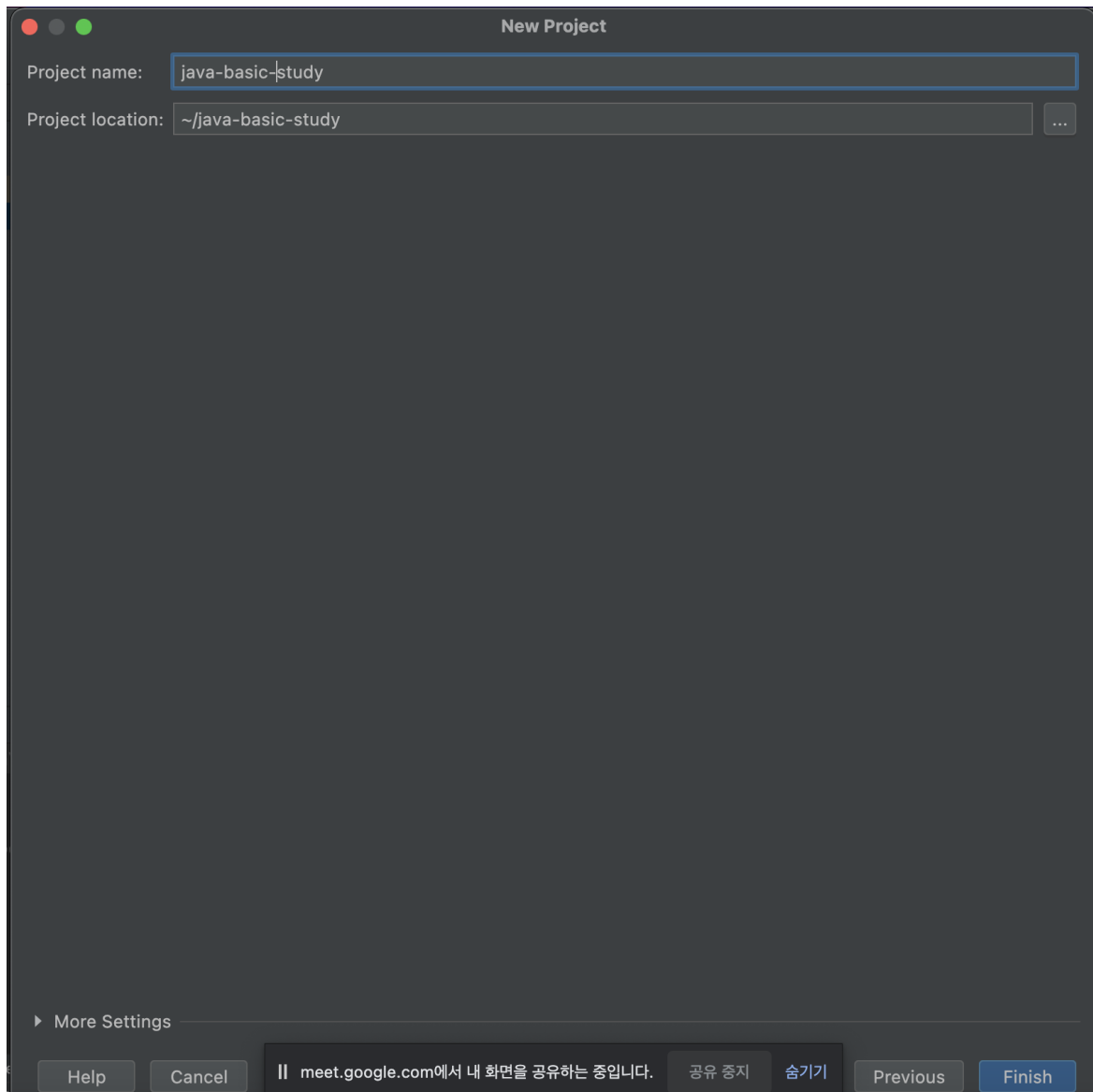
▼ 프로젝트 생성해보기 👁👁



| File > New > Project 클릭



java > Project SDK corretto-1.8 으로 선택 후 next > next



| project name 입력 (일단 java-basic-study 로 통일할게요!)

▼ github 사용해보기 🧐

깃을 설치해봅시다!! ([링크](#))

Git이란?

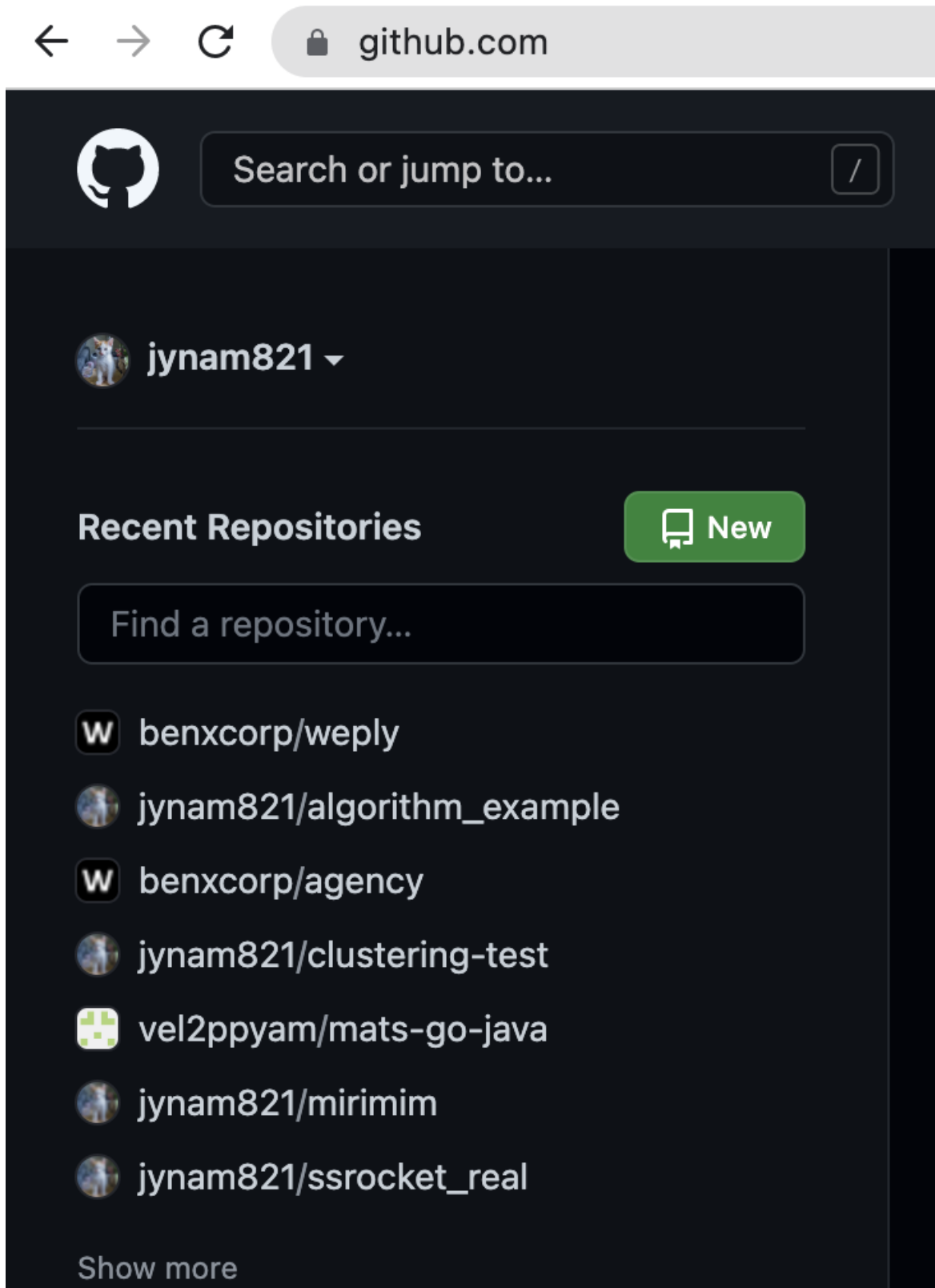
| Git은 형상 관리 도구 중 하나로, 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템

쉽게 말해 코드를 관리하기 쉽도록 하는 공개 소프트웨어

Github란?

깃허브(Github)는 분산 버전 관리 툴인 깃(Git)를 사용하는 프로젝트를 지원하는 웹호스팅 서비스

쉽게 말해 git을 편리하게 사용할 수 있는 호스팅 서비스입니다.




[깃허브 접속](#) > New

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 jynam821 ▾ / java-basic-study ✓

Great repository names are short and memorable. Need inspiration? How about **psychic-bassoon?**

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Java ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

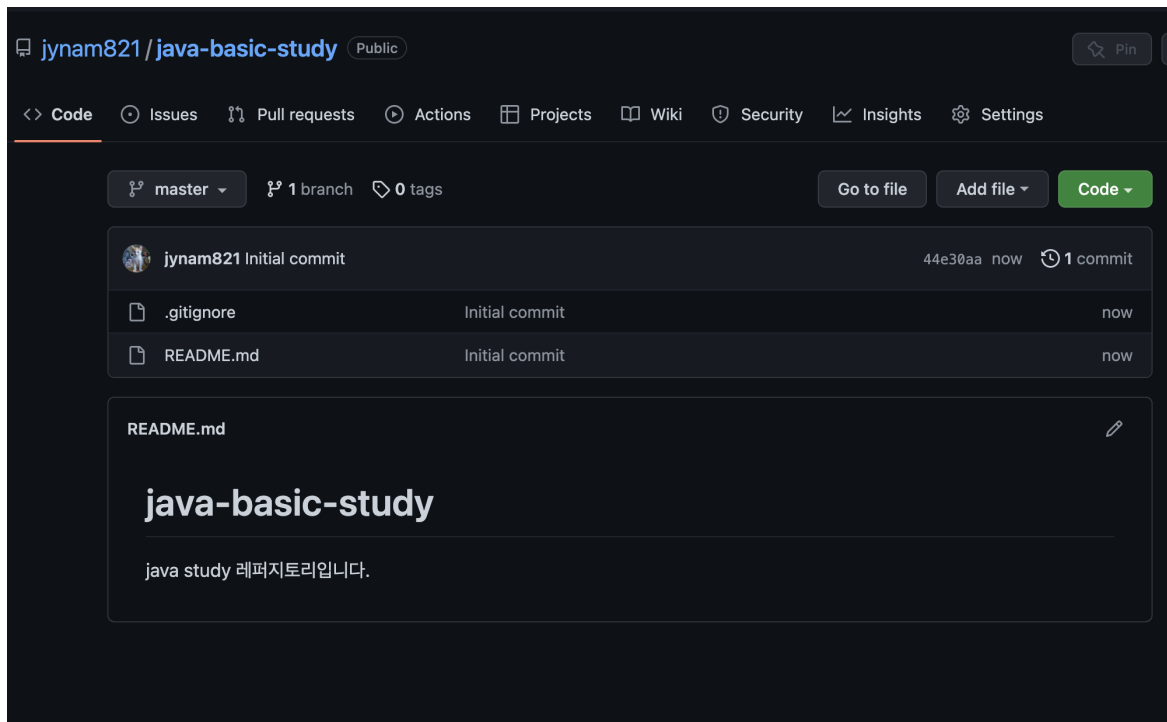
License: None ▾ **||** meet.google.com에서 내 화면을 공유하는 중입니다. 공유 중지 숨기기

Repository name - insert

Add a README file - check

Add .gitignore - java

위처럼 값을 채워넣고 repository를 생성합니다.



생성된 repository !!

```
git init
git status // 코드에 대한 스냅샷 상황
git add . //해당 디렉토리의 코드들을 add 함
git commit -m 'first commit' // 커밋
git remote add origin <github repository 주소> //해당 레포지터리에 자료를 올림
git push -u origin master // 서버에서 github 저장소로 자료 전송
```

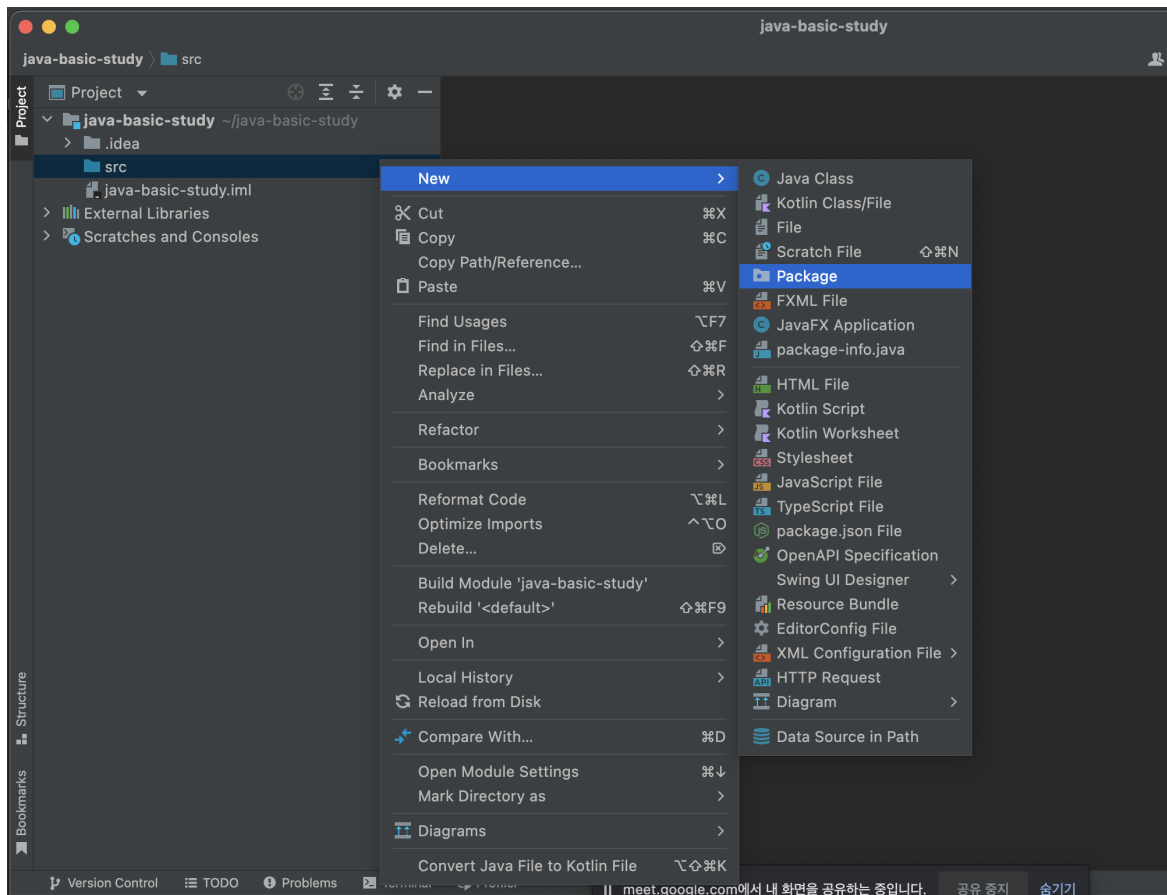


마법의 git GUI 소스트리 사용하기! ((현장에서 해봐요!!))

▼ Hello world 출력해보기 💪

모든 개발 언어의 첫 장은 'Hello World!'

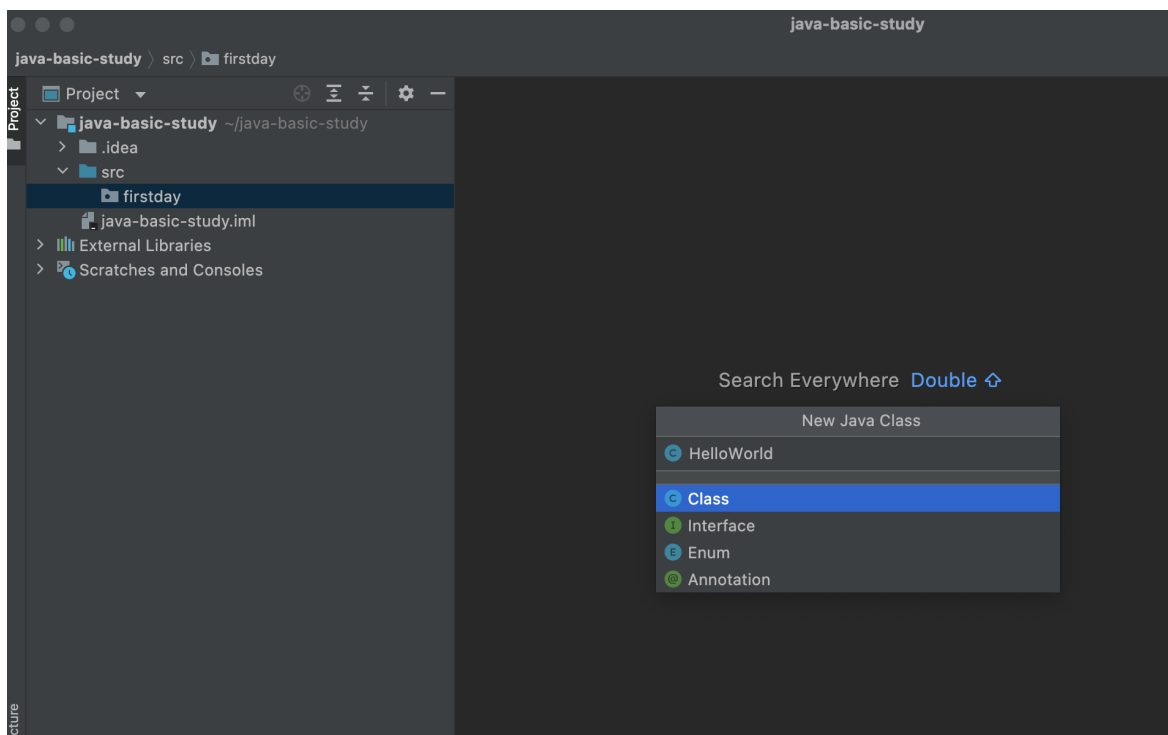
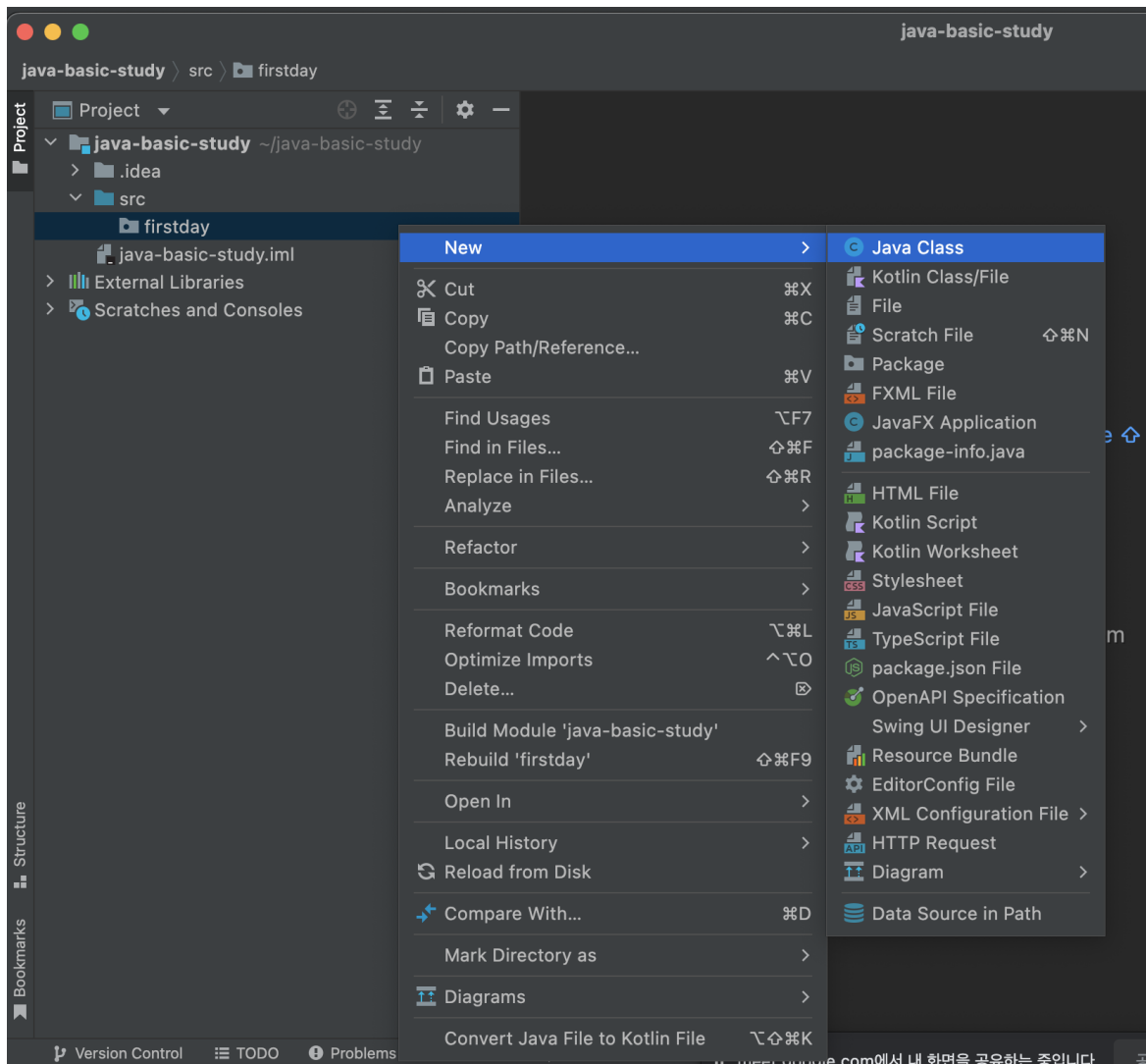
Hello World의 유래가 궁금하다면 ? 💪



앞으로 java 코드들은 src 폴더에 저장될 거예요.

src > New > Package

오늘은 첫 날이니 firstday 라는 패키지를 생성해줍니다 !



firstday에 마우스를 클릭한 후 new > Java class > Class
(나중에 배우지만, Class와 Interface와 Enum과 Annotation은 사용용도가
달라요)

HelloWorld 라는 class를 생성해줍니다.

생성된 코드는 위처럼 작성해줍니다.

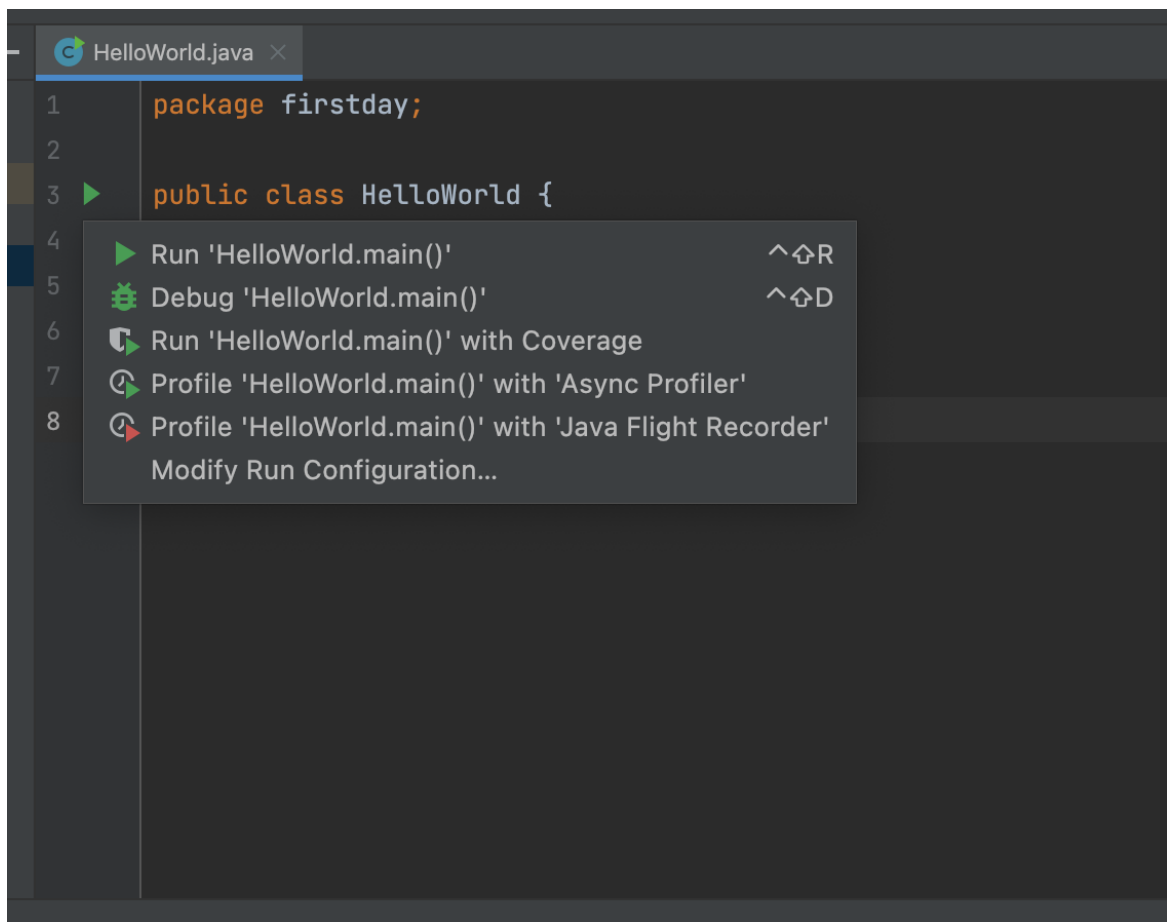
package는 src의 어떤 디렉토리에 해당 파일이 있는지 명시해주는 용도예요!

java는 실행할 때 main 함수부터 찾아요.

```
public static void main(String[] args){}
```



이것은 Java의 규칙 !



초록 실행 버튼을 누르고 Run 'HelloWorld.main()' 을 해주면

```

HelloWorld x
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
Hello World!

Process finished with exit code 0

```

짜잔~ Hellow World! 출력 완료 ~!

▼ 변수란 ? 🤔



변수(Variable)란 값(Value)을 저장할 수 있는 메모리의 공간 & 변할 수 있는 수

- 값을 담을 수 있는 상자
- 자료값을 지정해야함

▼ 자료형의 종류

	정수형	실수형	문자형	논리형
1 byte	byte			boolean
2 byte	short		char	
3 byte	int	float		
4 byte	long	double		

- 변수의 규칙제약을 지켜야함

▼ 이름 규칙제약

제약 사항	예시
변수 이름은 영문자와 숫자를 사용할 수 있고, \$, _ 특수문자를 사용 할 수 있다.	abc123, abc_def, abc\$123, _abcdef, ... (사용 가능)
변수 이름은 숫자로 시작할 수 없다.	123abc (사용불가)
자바에서 사용되는 예약어는 사용할 수 없다.	int, for, break, while, ... (사용불가)

▼ 변수 예제

- ▼ 예제 전에 잠깐 ! (개발자들이 intelliJ를 선호하는 이유)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/957eb704-3f7b-4058-b94a-d0266c49d5c0/화면_기록_2022-06-04_오후_3.43.21.mov

자동완성이 너무 잘되어있어요..

굳이 public static void main을 제가 다 칠 필요가 없답니다?

하지만 처음에는 손으로 치는 걸 추천드려요 :)

▼ VarTest1.java

```
package firstday;

public class VarTest1 {
    public static void main(String[] args) {
        //Java에서 '//'는 주석
        String message; //문자형 변수 message를 선언
        message = "Hello World"; //message변수에 "Hello World"를 대입
        System.out.println(message);
    }
}
```

```
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
Hello World

Process finished with exit code 0
```

▼ VarTest2.java

```
package firstday;

public class VarTest2 {
    public static void main(String[] args) {
        String name = "남정윤";
        int age = 21;
        System.out.println("나의 이름은 " + name + "이고, 나이는 " + age + " 입니다.");
    }
}
```

```
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
나의 이름은 남정윤이고, 나이는 21 입니다.

Process finished with exit code 0
```

▼ 실수란 ? 🤔



실수(Constant)란 변하지 않는 수

- final 예약어를 사용하여 선언
- 변하지 않는 값을 선언할 때 사용 (ex : PI = 3.14)

```
final double PI = 3.14;

//PI = 5.14; (불가능)
```

▼ 입출력

▼ System.in.read()

- 단일 문자입력
- 입력 Buffer로부터 아스키코드를 입력받는다.
- `IOException` 예외처리가 필요하다.
- 오라클 공식 문서 ([링크](#))

▼ read() 예제

▼ InputTest1.java

```
package firstday;

import java.io.IOException;

public class InputTest1 {
    public static void main(String[] args) throws IOException {
        int val, val2, val3;
        System.out.print("입력 : ");
        val = System.in.read();
        val2 = System.in.read();
        val3 = System.in.read();//문자 하나만 처리가능
        // System.in.read는 무조건 int타입의 변수로 받아줘야함
        System.out.println("입력 data : " + val);
        System.out.println("입력 data : " + val2);
        System.out.println("입력 data : " + val3);
    }
}
```

```
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
입력 : ABC
입력 data : 65
입력 data : 66
입력 data : 67

Process finished with exit code 0
```

앵 이게 뭐야? 싶죠??

나는 ABC는 입력했는데 왜 숫자가 나오지?

이게 바로 아스키코드이다. 자세한 건 [링크](#) 참고

▼ InputTest2.java

```
package firstday;

import java.io.IOException;

public class InputTest2 {
    public static void main(String[] args) throws IOException {
        System.out.print("문자 입력: ");
        int num = System.in.read();
        System.out.println("==== 출 력 ====");
        System.out.println("입력하신 문자 : " + (char)num);
        System.out.println("소문자로 변환 : " + (char)(num+32));
    }
}
```

```
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
문자 입력: A
==== 출 력 ====
입력하신 문자 :A
소문자로 변환 :a

Process finished with exit code 0
```

아스키 코드를 위처럼 char 자료형으로 변환해서 사용할 수 있다 :)

▼ Scanner

- 문장 입력 가능
- java.util.Scanner를 import 해야함.
- 오라클 공식 문서 ([링크](#))

▼ scanner 예제

▼ InputTest3.java

```
package firstday;

import java.util.Scanner;

public class InputTest3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int age;
        String name;

        System.out.println("---입력---");
        System.out.print("이름 입력:");
        name = input.next();
        System.out.print("나이: ");
        age = input.nextInt();
        System.out.println("---출력---");
        System.out.println("이름: " + name);
        System.out.println("나이: " + age);
    }
}
```



```
}
}
```

```
InputTest3 x
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
---입력---
이름 입력: 남정윤
나이: 21
---출력---
이름: 남정윤
나이: 21

Process finished with exit code 0
```

▼ InputTest4.java

```
package firstday;

import java.util.Scanner;

public class InputTest4 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String name;
        String sentence;

        System.out.println("---입력---");
        System.out.print("당신의 이름은 무엇입니까?:");
        name = input.next();
        input.nextLine(); //엔터를 제거하기위함
        System.out.print("당신이 가장 좋아하는 문장은 무엇입니까?:");
        sentence = input.nextLine();
        System.out.println("---출력---");
        System.out.println(name + "님이 가장 좋아하시는 문장은 '" + sentence + "'입니
다.");

        // next() 띄어쓰기를 기준으로 입력을 받음
        // nextLine() 개행문자를 기준으로 입력을 받음
    }
}
```

```
InputTest3 x InputTest4 x
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
---입력---
당신의 이름은 무엇입니까?: 남정윤
당신이 가장 좋아하는 문장은 무엇입니까?: 나는 해야한다. 그래서 나는 할 수 있다.
---출력---
남정윤님이 가장 좋아하시는 문장은 '나는 해야한다. 그래서 나는 할 수 있다.'입니다.

Process finished with exit code 0
```

▼ 연산



연산자(Operator)란 연산에 사용되는 표시나 기호

ex) 산술연산자 +, -, *, /, %

- 연산 : 프로그램에서 데이터를 처리하여 결과를 산출하는 것
- 피연산자(operand) : 연산되는 데이터
- 연산 순서 : 산술 → 비교 → 논리 → 대입

▼ 대입연산자



=, 연산자의 오른쪽에 있는 값(B)을 왼쪽(A)에 대입한다.

ex) A는 B이다.

```
int num = 4; // num은 4이다.
String name = "daria"; //name은 daria이다.
char alpha = 'B'; // alpha는 B이다.
```

▼ 산술연산자

+	a와 b의 값을 더한다. ex) num = a + b;
-	a에서 b를 뺀다. ex) num = a - b;
*	a와 b의 값을 곱한다. ex) num = a * b;
/	a에서 b를 나눈다. ex) num = a / b; //몫을 반환
%	a에서 b를 나눈다. ex) num = a % b; //나머지를 반환

▼ OperatorTest1.java

```
package firstday;

import java.util.Scanner;

public class OperatorTest1 {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        System.out.println(" 두개의 정수를 입력하세요 :");

        int num1 = scan.nextInt();
        int num2 = scan.nextInt();

        System.out.println("더한 결과 : " + (num1+num2));
        System.out.println("뺀 결과 : " + (num1-num2));
        System.out.println("곱한 결과 : " + (num1*num2));
        System.out.println("나눈 결과 : " + (num1/num2));
        System.out.println("나머지 : " + (num1%num2));
    }
}
```

```

/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
두개의 정수를 입력하세요 :
7
3
더한 결과 : 9
뺀 결과 : 5
곱한 결과 : 14
나눈 결과 : 3
나머지 : 1

Process finished with exit code 0

```

▼ 복합대입연산자

아래 표를 보시면 무슨 용도인지 이해가 가실 거예요 !

왼쪽 데이터와 오른쪽 데이터는 같은 의미입니다.

a += b	a = a + b;
a -= b	a = a - b;
a *= b	a = a * b;
a /= b	a = a / b;
a %= b	a = a % b;

▼ OperatorTest2.java

```

package firstday;

import java.util.Scanner;

public class OperatorTest2 {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        System.out.println(" 두개의 정수를 입력하세요 :");

        int num1 = scan.nextInt();
        int num2 = scan.nextInt();

        System.out.println("num1 : " + num1);
        num1 += num2;
        System.out.println("num1 += num2 : " + num1);
        num1 -= num2;
        System.out.println("num1 -= num2: " + num1);
        num1 *= num2;
        System.out.println("num1 *= num2: " + num1);
        num1 /= num2;
        System.out.println("num1 /= num2: " + num1);
        num1 %= num2;
        System.out.println("num1 %= num2: " + num1);
    }
}

```

```

/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
두개의 정수를 입력하세요 :
2 3
num1 : 2
num1 += num2 : 5
num1 -= num2: 2
num1 *= num2: 6
num1 /= num2: 2
num1 %= num2: 2
Process finished with exit code 0

```

▼ 관계(비교)연산자



크고 작음과 같고 다름을 비교

ex) >, <, >=, <=, ==, !=

▼ OperatorTest3.java

```

package firstday;

import java.util.Scanner;

public class OperatorTest3 {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        System.out.println(" 정수를 입력하세요 (num1):");
        int num1 = scan.nextInt();
        System.out.println(" 정수를 입력하세요 (num2):");
        int num2 = scan.nextInt();
        System.out.println("num1 > num2: " + (num1 > num2) );
        System.out.println("num1 < num2: " + (num1 < num2) );
        System.out.println("num1 >= num2: " + (num1 >= num2) );
        System.out.println("num1 <= num2: " + (num1 <= num2) );
        System.out.println("num1 == num2: " + (num1 == num2) );
        System.out.println("num1 != num2: " + (num1 != num2) );
    }
}

```

```

/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
정수를 입력하세요 (num1):
2
정수를 입력하세요 (num2):
3
num1 > num2: false
num1 < num2: true
num1 >= num2: false
num1 <= num2: true
num1 == num2: false
num1 != num2: true
Process finished with exit code 0

```

▼ 논리연산자



둘 이상의 조건을 '그리고(AND) 또는 (OR)'으로 연결하여 하나의 식으로 표현ex)
>, <, >=, <=, ==, !=

▼ OperatorTest4.java

```
package firstday;

import java.util.Scanner;

public class OperatorTest4 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println(" 정수를 입력하세요 (num1):");
        int num1 = scan.nextInt();
        System.out.println(" 정수를 입력하세요 (num2):");
        int num2 = scan.nextInt();
        System.out.println("(num1 > num2) && (num1 == num2): " + ((num1 > num2) && (num1 == num2)));
        System.out.println("(num1 > num2) && (num1 == num2): " + ((num1 > num2) && (num1 == num2)));
        System.out.println("(num1 != num2) || (num1 == num2): " + ((num1 != num2) || (num1 == num2)));
    }
}
```

```
/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
정수를 입력하세요 (num1):
정수를 입력하세요 (num2):
(num1 > num2) && (num1 == num2): false
(num1 > num2) && (num1 == num2): false
(num1 != num2) || (num1 == num2): true
Process finished with exit code 0
```

▼ 증감연산자



피연산자에 저장된 값을 1 증가 또는 1 감소

- 증감 연산자는 전위형과 후위형으로 나뉨.
- 전위형 (++a) - 명령어보다 먼저 실행함.
- 후위형 (b++) - 명령어후에 실행함.

▼ OperatorTest5.java

```
package firstday;

import java.util.Scanner;
```

```

public class OperatorTest5 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println(" 정수를 입력하세요 (num):");
        int num = scan.nextInt();
        System.out.println("num: " + num);
        System.out.println("num++: " + num++);
        System.out.println("num: " + num);
        System.out.println("++num: " + ++num);
        System.out.println("num: " + num);
        System.out.println("num--: " + num--);
        System.out.println("num: " + num);
        System.out.println("--num: " + --num);
        System.out.println("num: " + num);
    }
}

```

```

/Users/user/Library/Java/JavaVirtualMachines/corretto-1.8.0_332/Contents/Home/bin/java ...
정수를 입력하세요 (num):
3
num: 3
num++: 3
num: 4
++num: 5
num: 5
num--: 5
num: 4
--num: 3
num: 3

Process finished with exit code 0

```

- 비트연산자
- 시프트연산자

▼ 실습 예제

2022.06.05 실습 예제