JAVA 프로그래밍

전광욱

따라하며 익히는 자바프로그램 언어

학습내용

- 자바로 프로그램 만들고 실행
- 콘솔 실행 방법(시작-cmd)
- 프로그램 컴파일 및 실행(자바 설치 및 환경 설정)
- path 이해 및 설정
- 자바 IDE 도구 설치
- 자바언어의 이해

자바로 프로그램 만들고 실행

1. **Notepad** 편집기 실행
2. 코딩하기

```
public class First{
public static void main(String[] arg){
System.out.println(“hello java”);
}
}
```

3. 저장하기**(First.java)**

4. 탐색기를 실행하여 저장 위치를 열고 주소 줄에 **cmd** 명령 입력 후 **Enter**

5. 실행 첫 번째 방법(인터프리터 방식)

java First.java

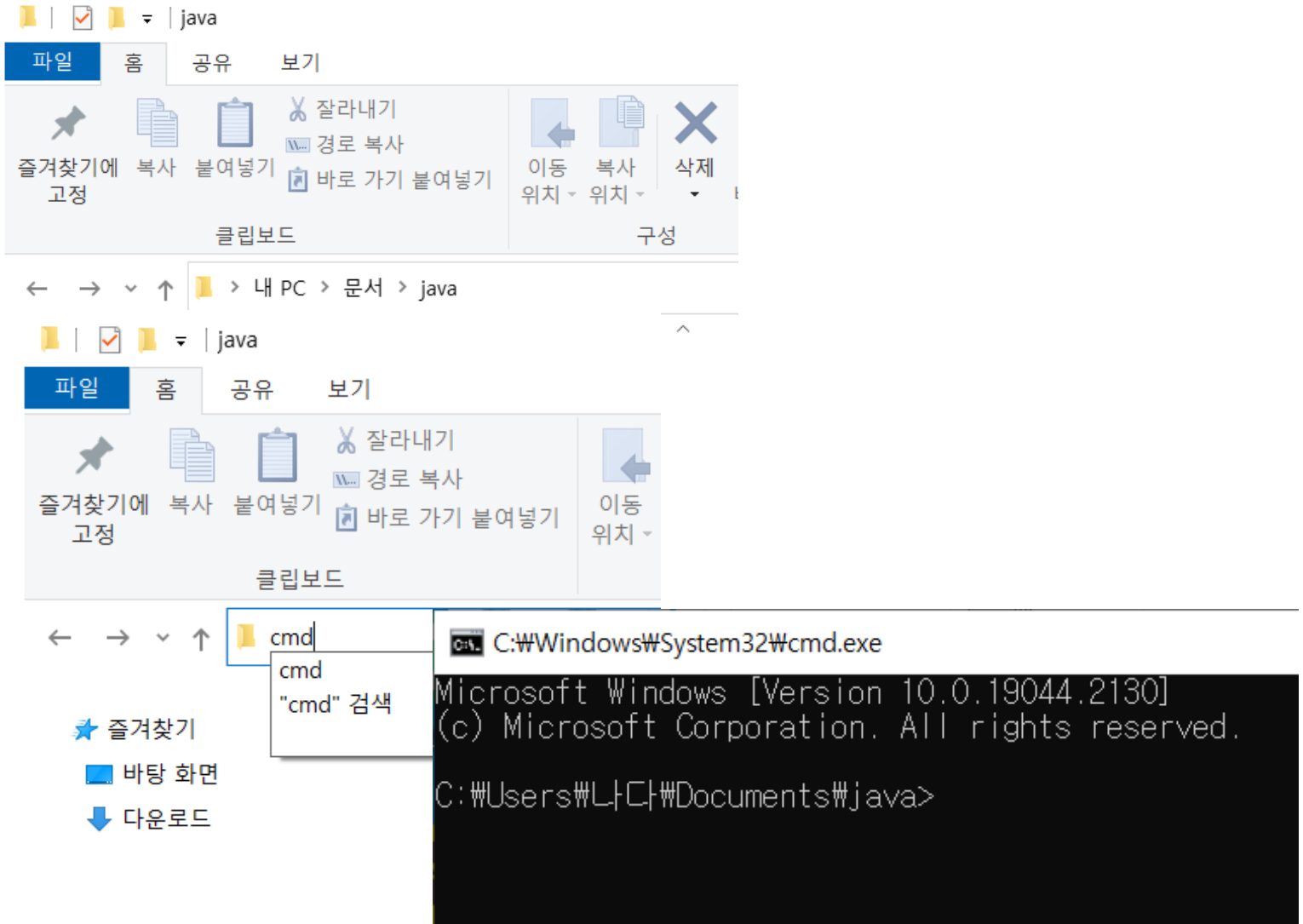
5. 실행 두 번째 방법(컴파일 방식)

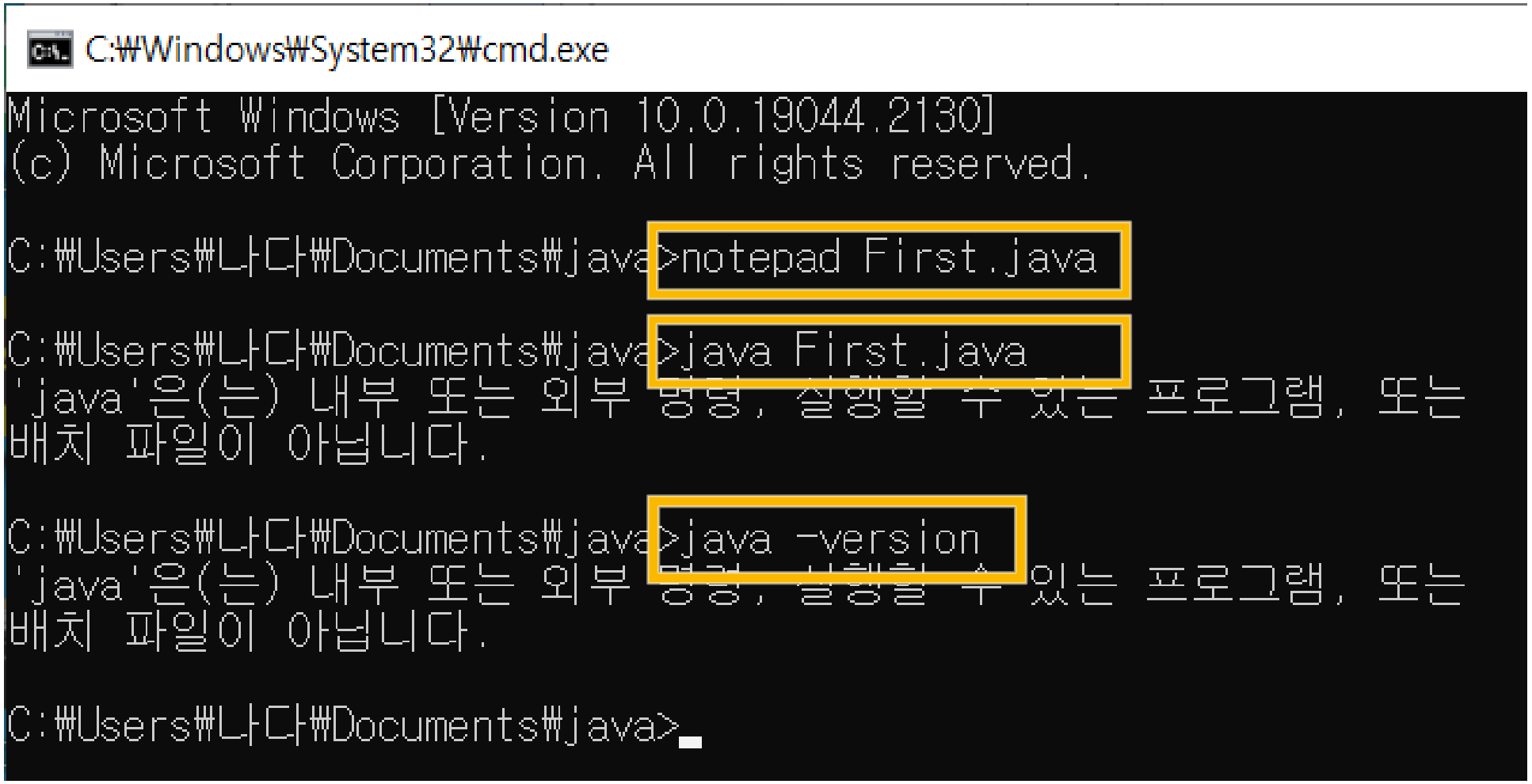
javac First.java

java First

콘솔실행방법

1) 시작 – cmd(내 문서로 이동)
2) 브라우저로 경로 이동 후 cmd





위의 코드의 문제점과 해결방법
문제점 : 프로그램이 실행이 되지 않음
해결 방법: 자바도구를 설정하기(환경설정)

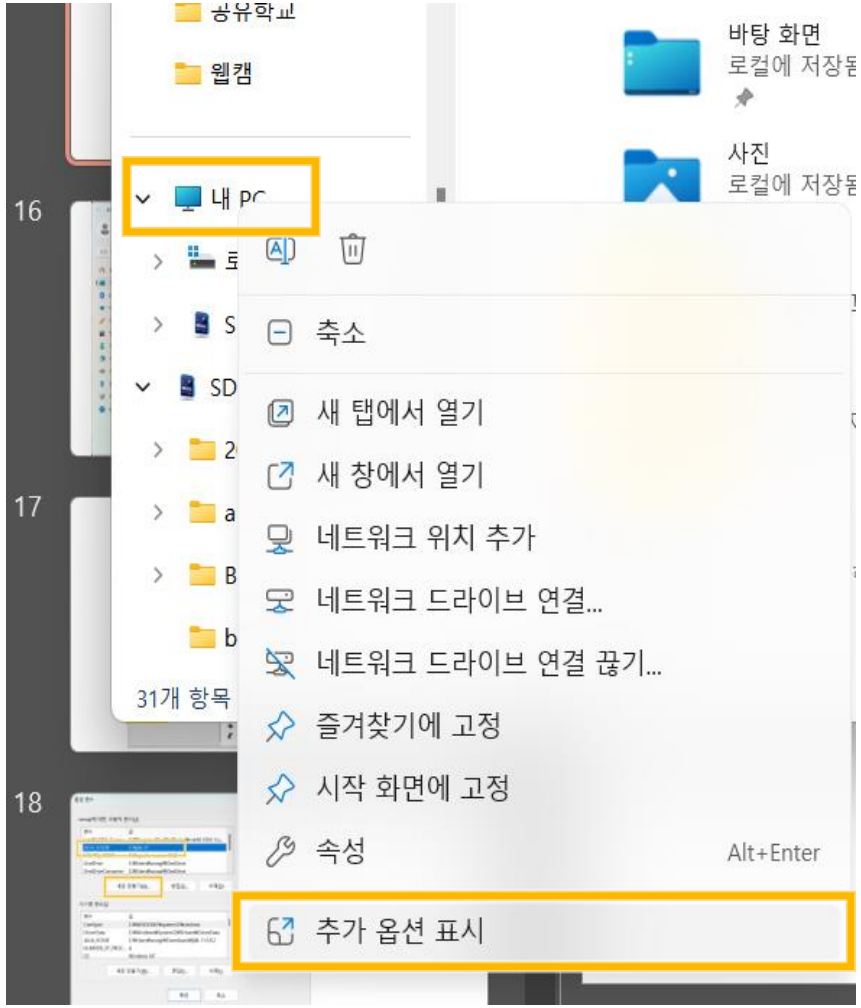
프로그램 컴파일 및 실행(자바 설치 및 환경 설정)

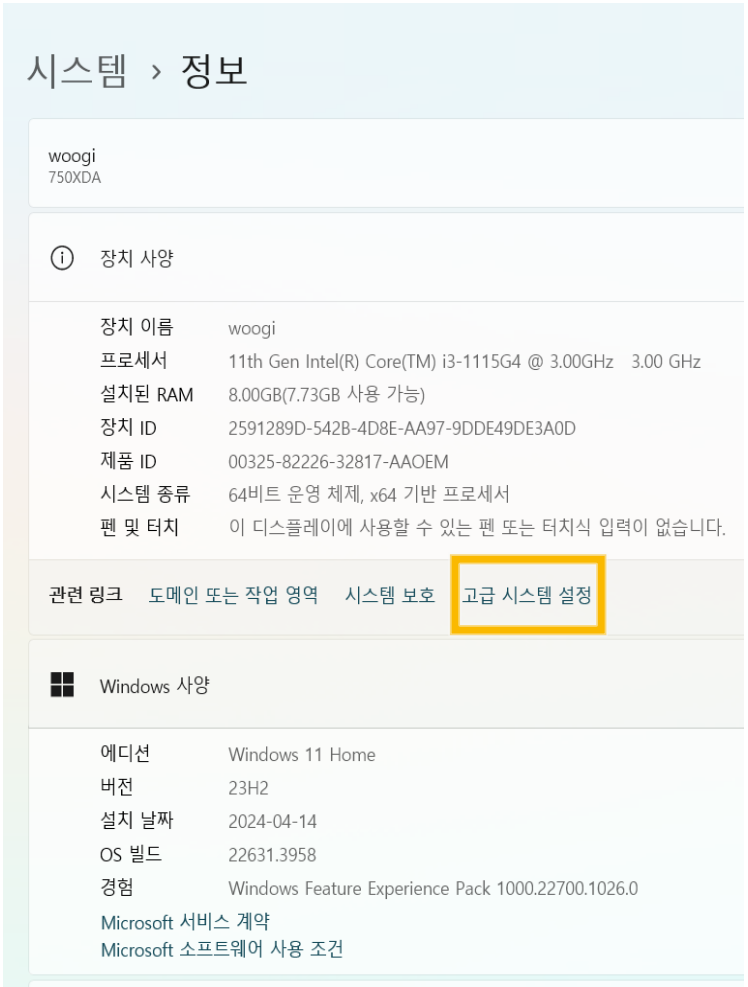
웹 브라우저를 실행하고 jdk.java.net 홈페이지로 이동

path 이해 및 설정

- 문제점 : 경로의 복잡함을 path설정으로 없애기
- 윈도우에서 path설정하기
- 콘솔에서 path설정하기

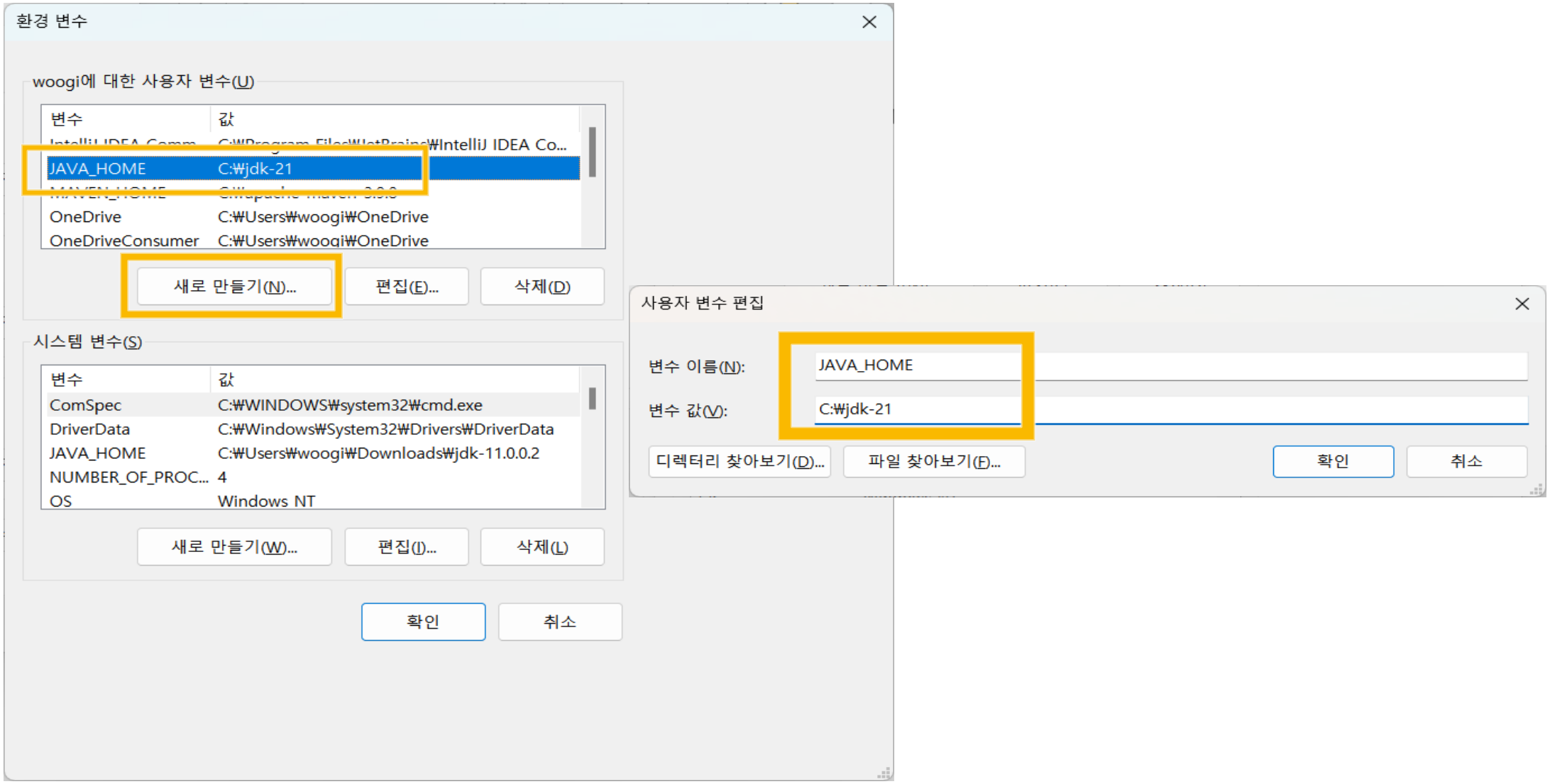
탐색기를 열고-내 PC 오른쪽 마우스 클릭-추가옵션표시 (win10에서는 속성)-고급시스템설정 선택



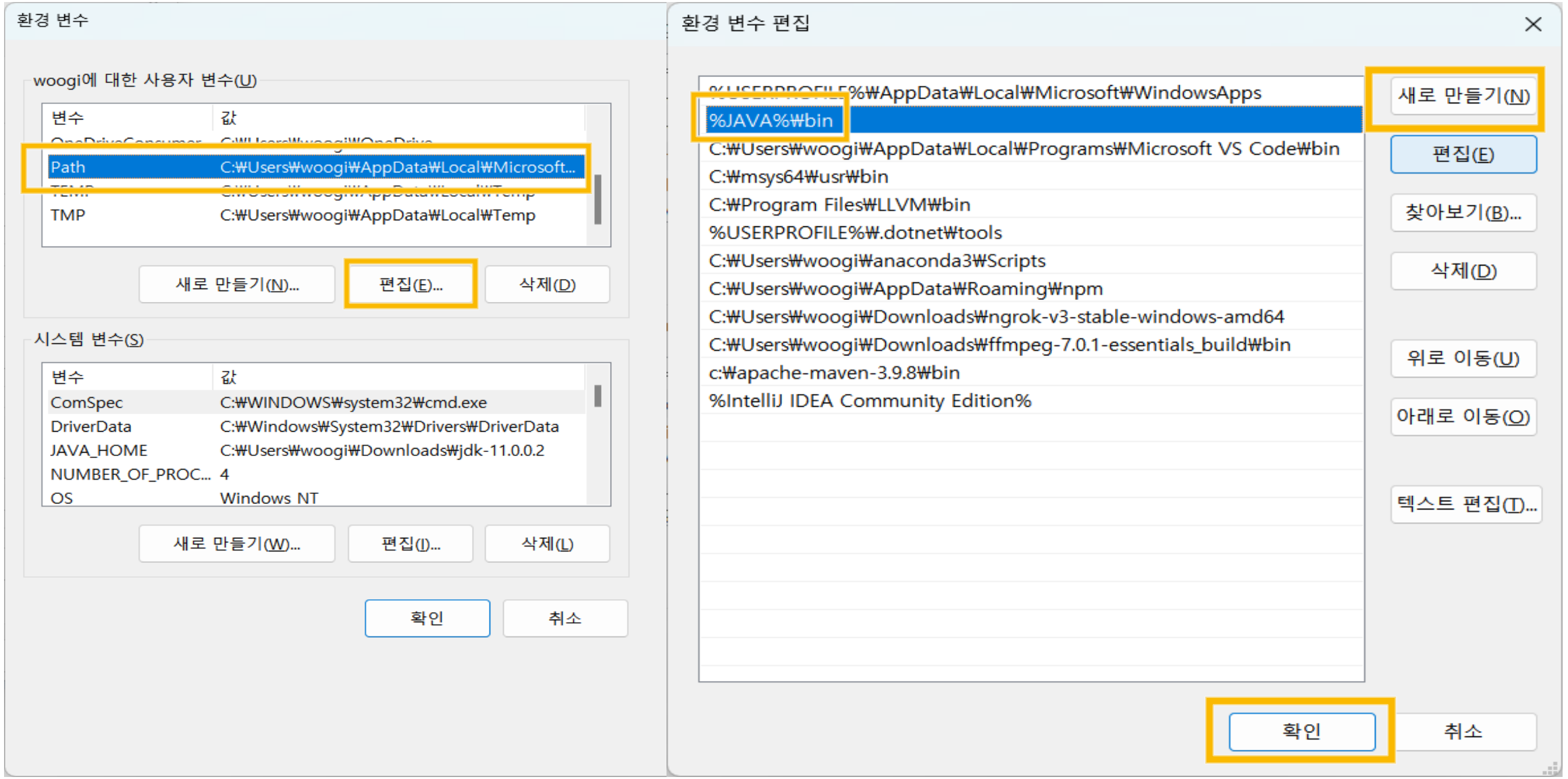


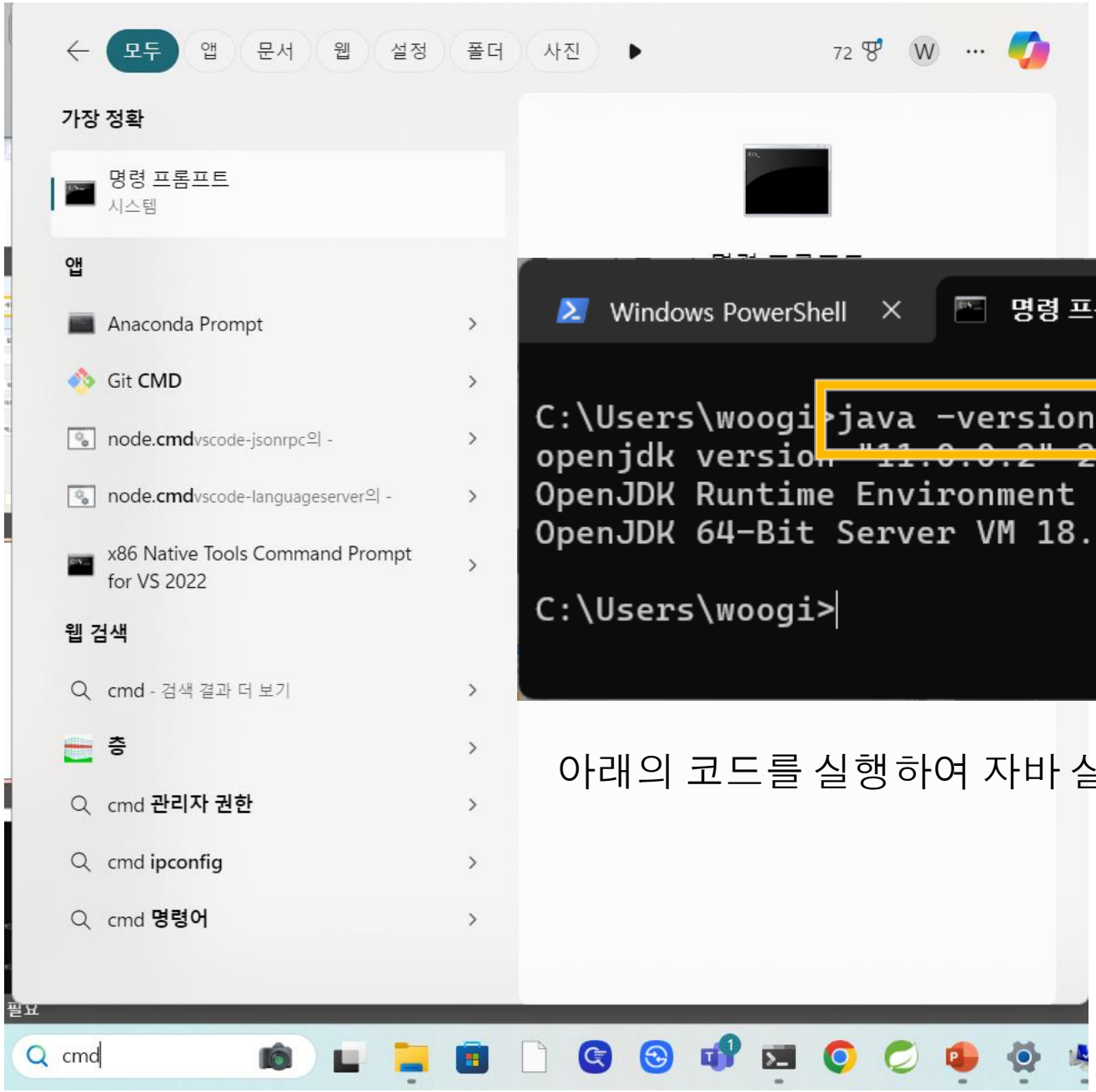
정보-고급시스템설정-시스템 속성창의 환경변수를 선택

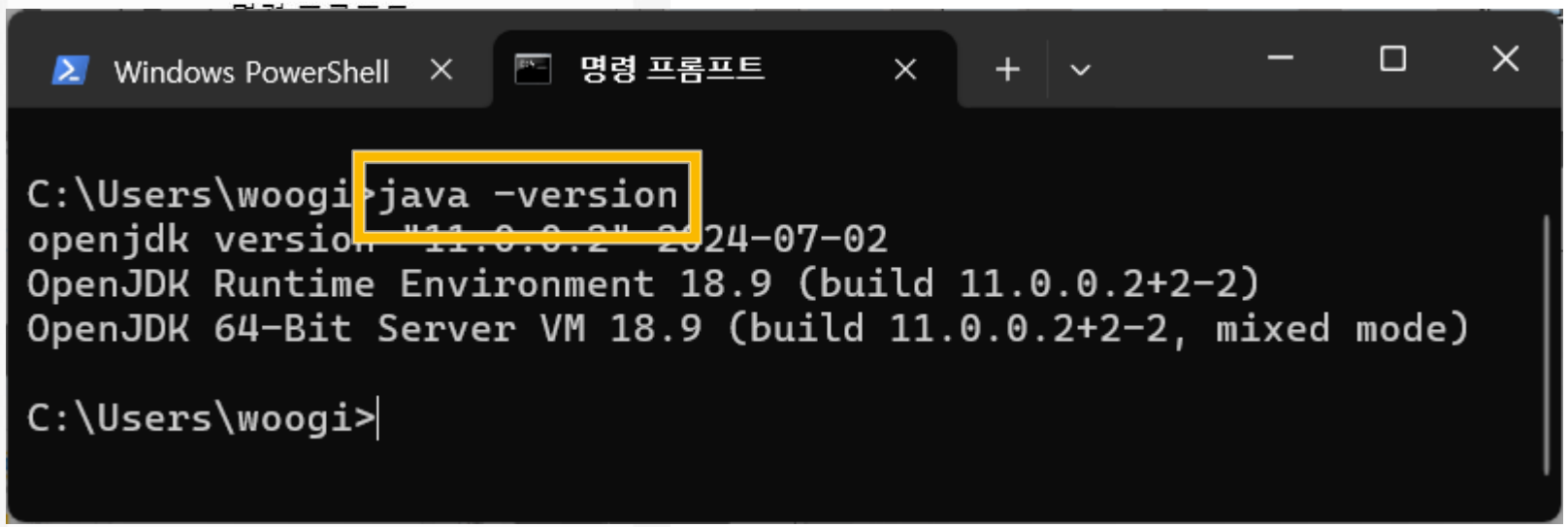
새로만들기 버튼을 클릭하고-변수이름 :JAVA_HOME, 변수값 c:\jdk-21 입력 후 확인버튼



Path를 선택-편집버튼 클릭-새로만들기 클릭-%JAVA_HOME\bin 입력 후 확인







아래의 코드를 실행하여 자바 실행도구가 설치 되어 있는지 확인

왼쪽 command 명령 해석

Java 디렉토리를 만들고
Java 디렉토리로 이동하고
노트패드를 실행하여 First.java파일 생성하고
컴파일없이 소스코드 실행

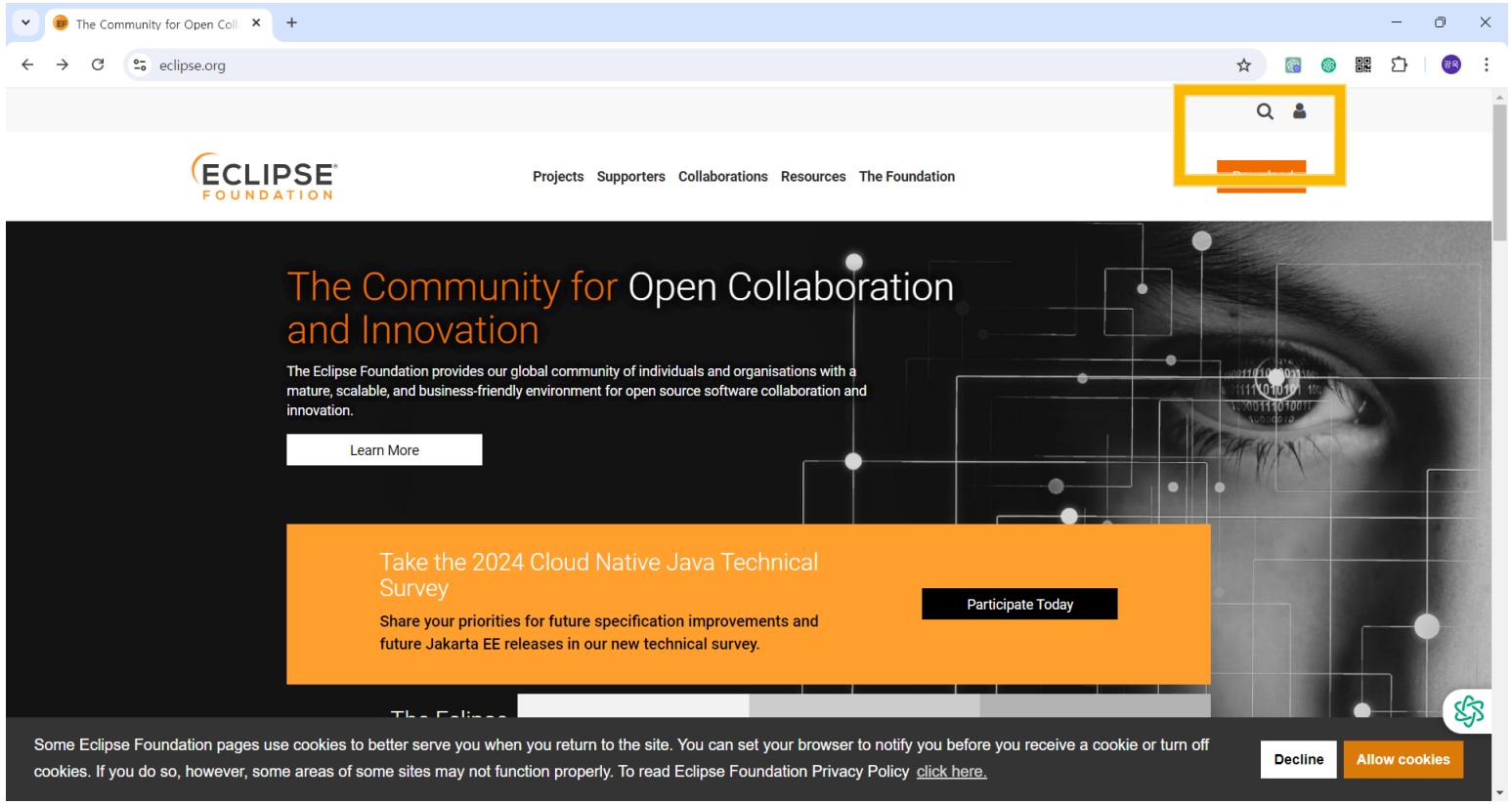
컴파일을 하면 First.class파일 생성
컴파일한 class파일을 실행

문제점 : 위의 코드로 프로그래밍하면 코드관리의 어려움이 발생함

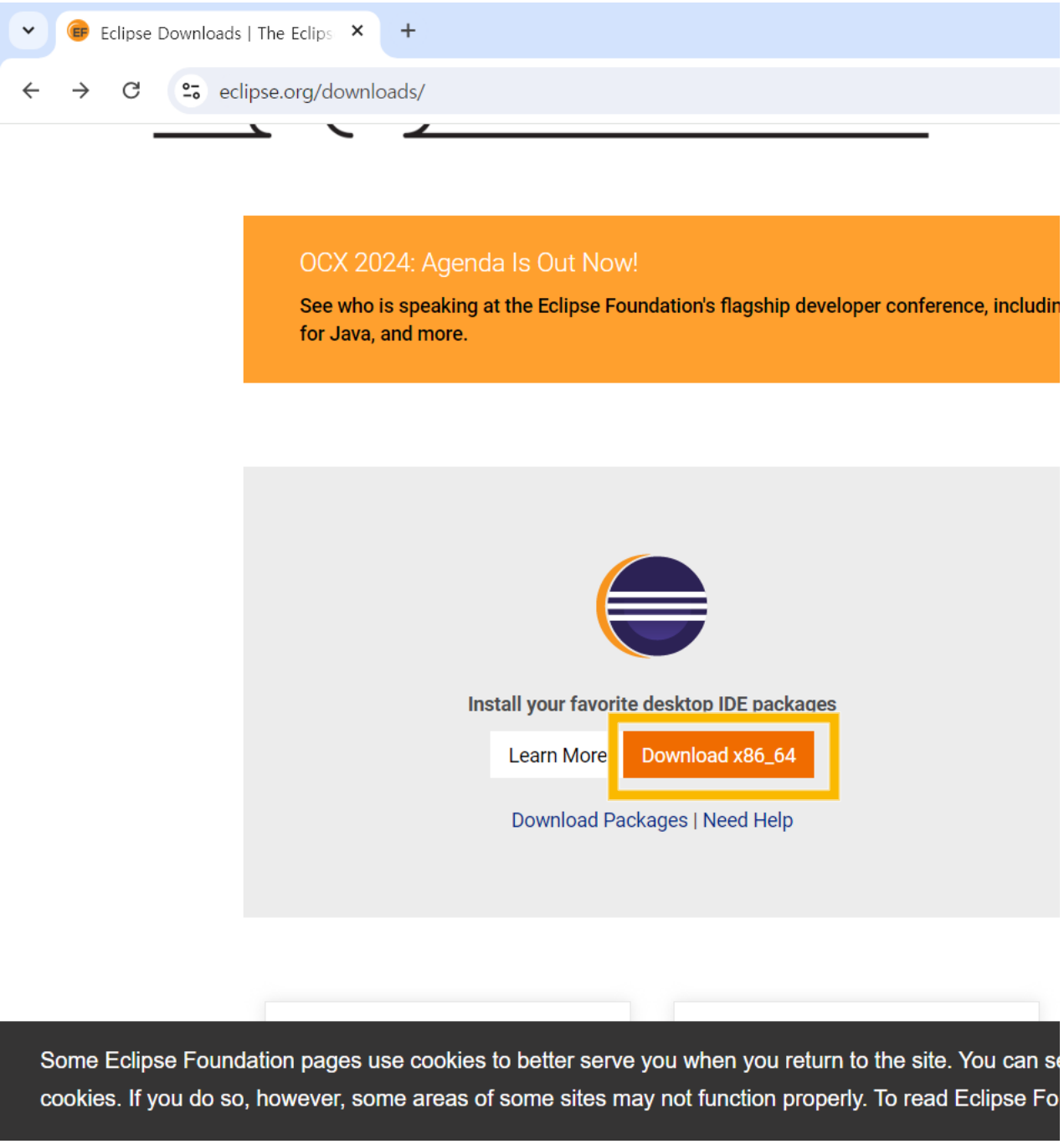
해결방법 : IDE도구 설치

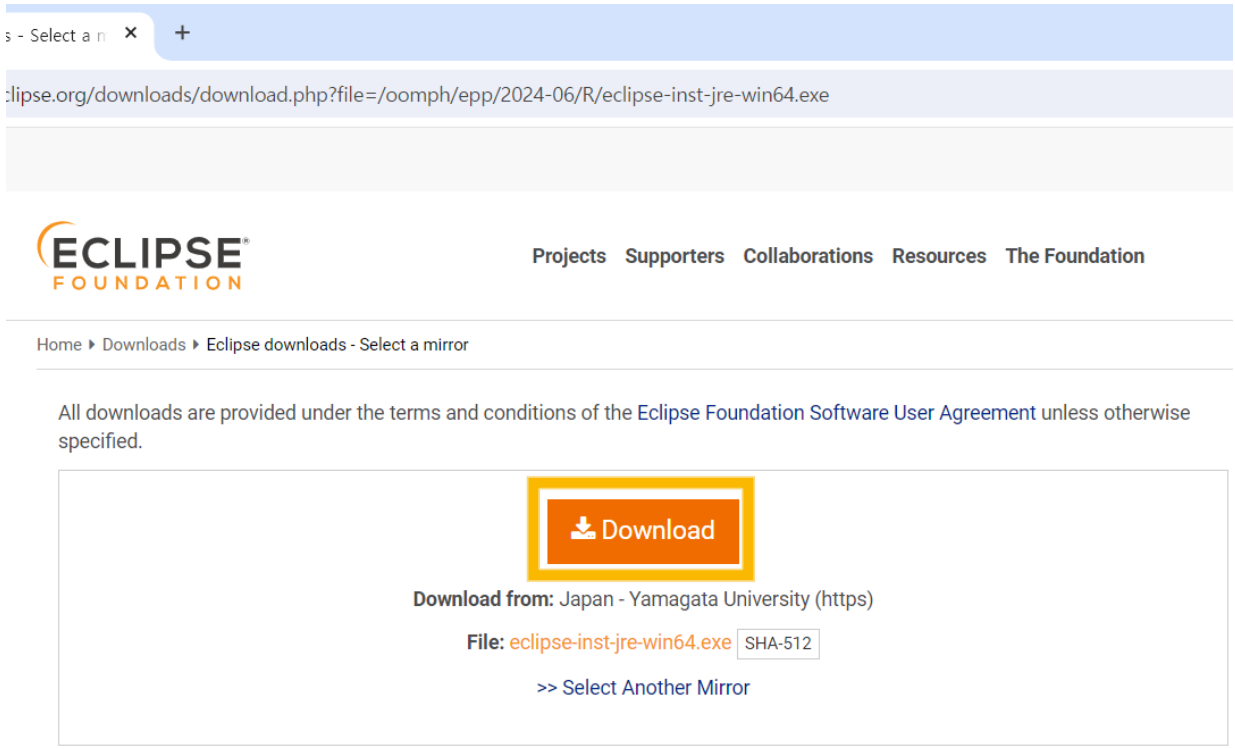
자바 IDE 도구 설치

이클립스 홈페이지 이동하여 프로그래 다운로드
eclipse.org

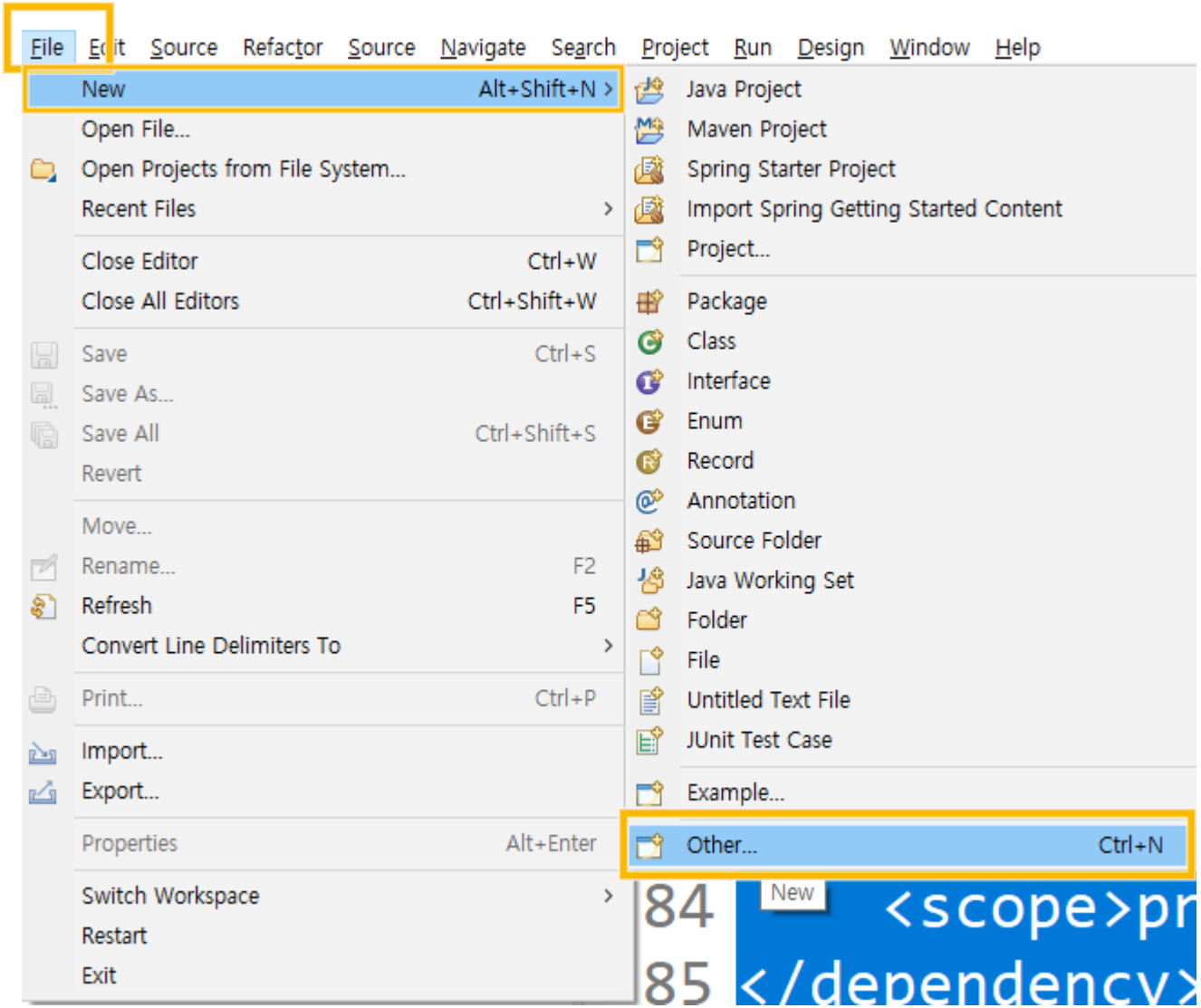


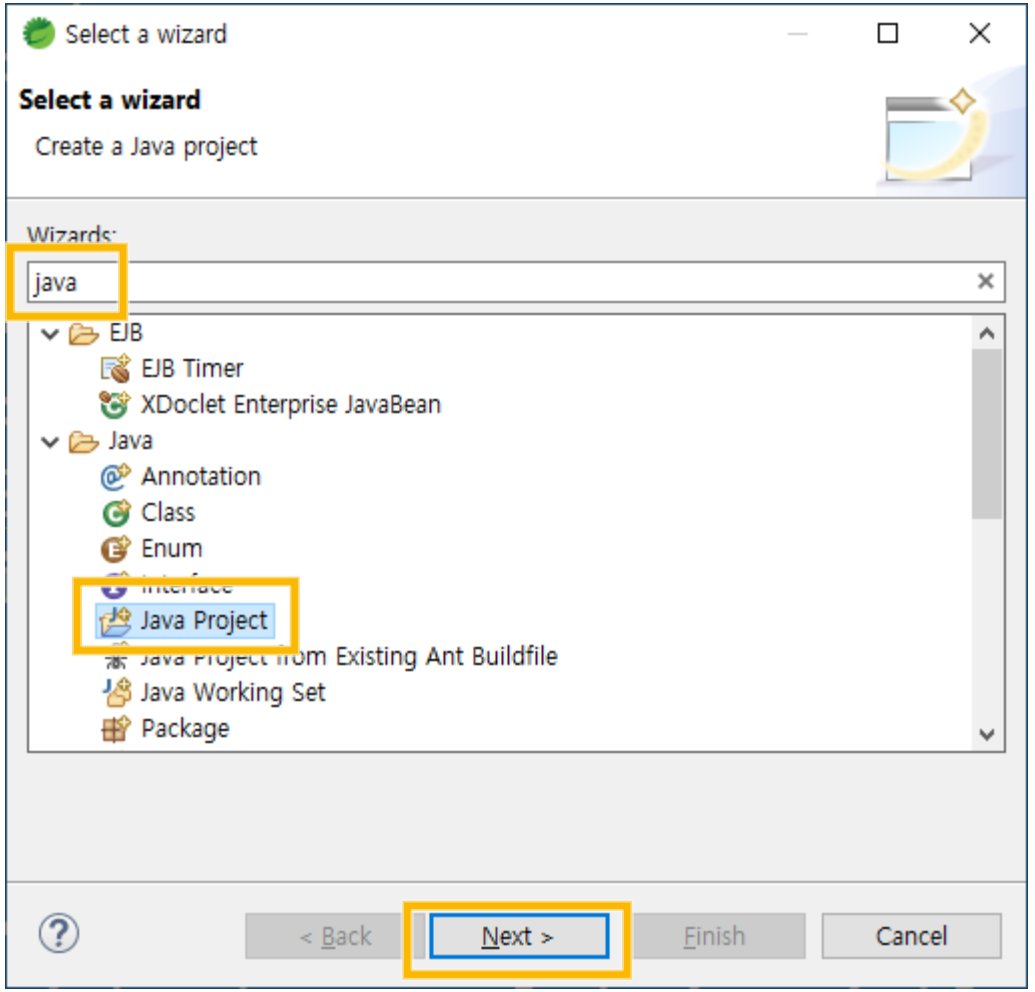
사이트로 이동하여 다운로드 후 프로그램 설치
프로그램 설치는 다운로드 후 파일을 클릭하여
다음 버튼을 계속 선택하여 설치하며 됨.

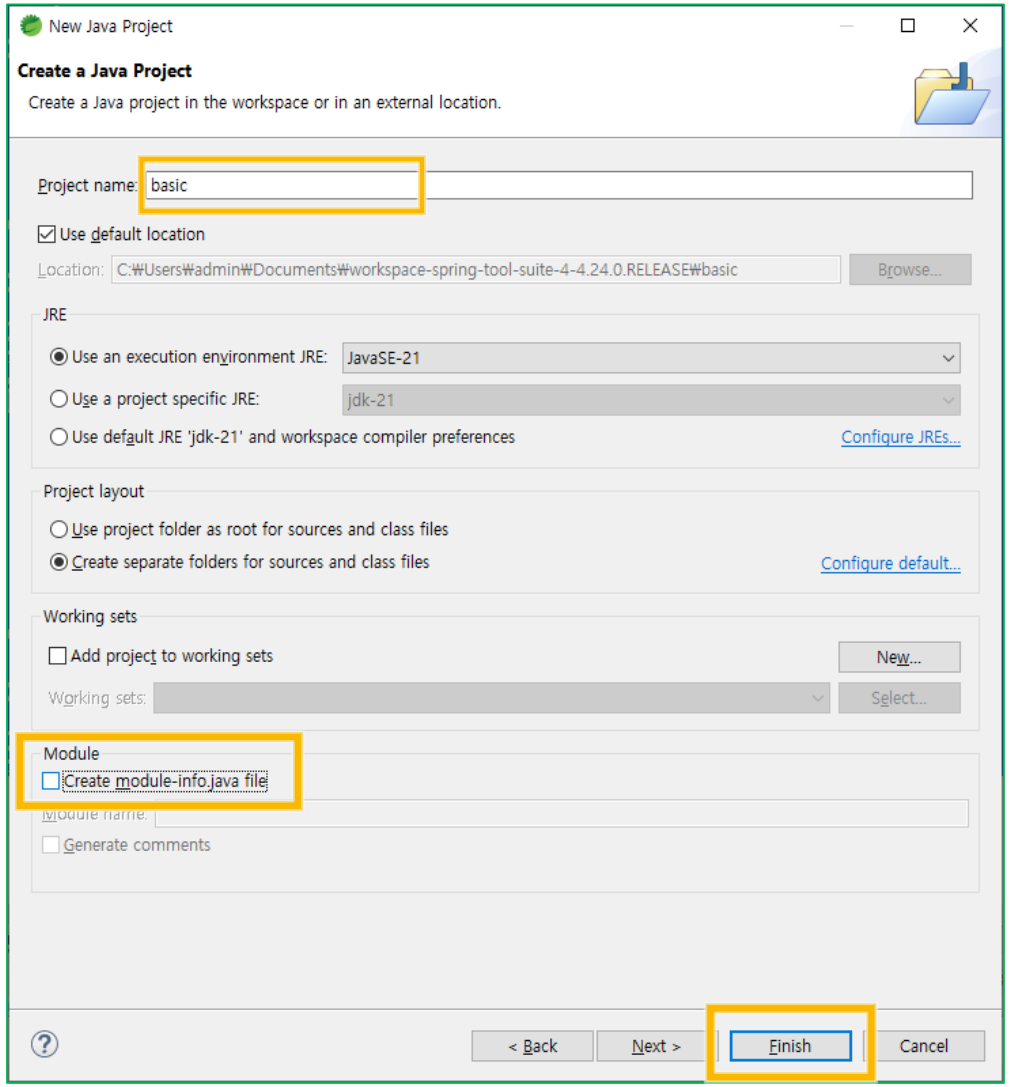




File-New-Other 선택하면 프로젝트 선택창 열림
편집창에 java검색-java Project선택-next

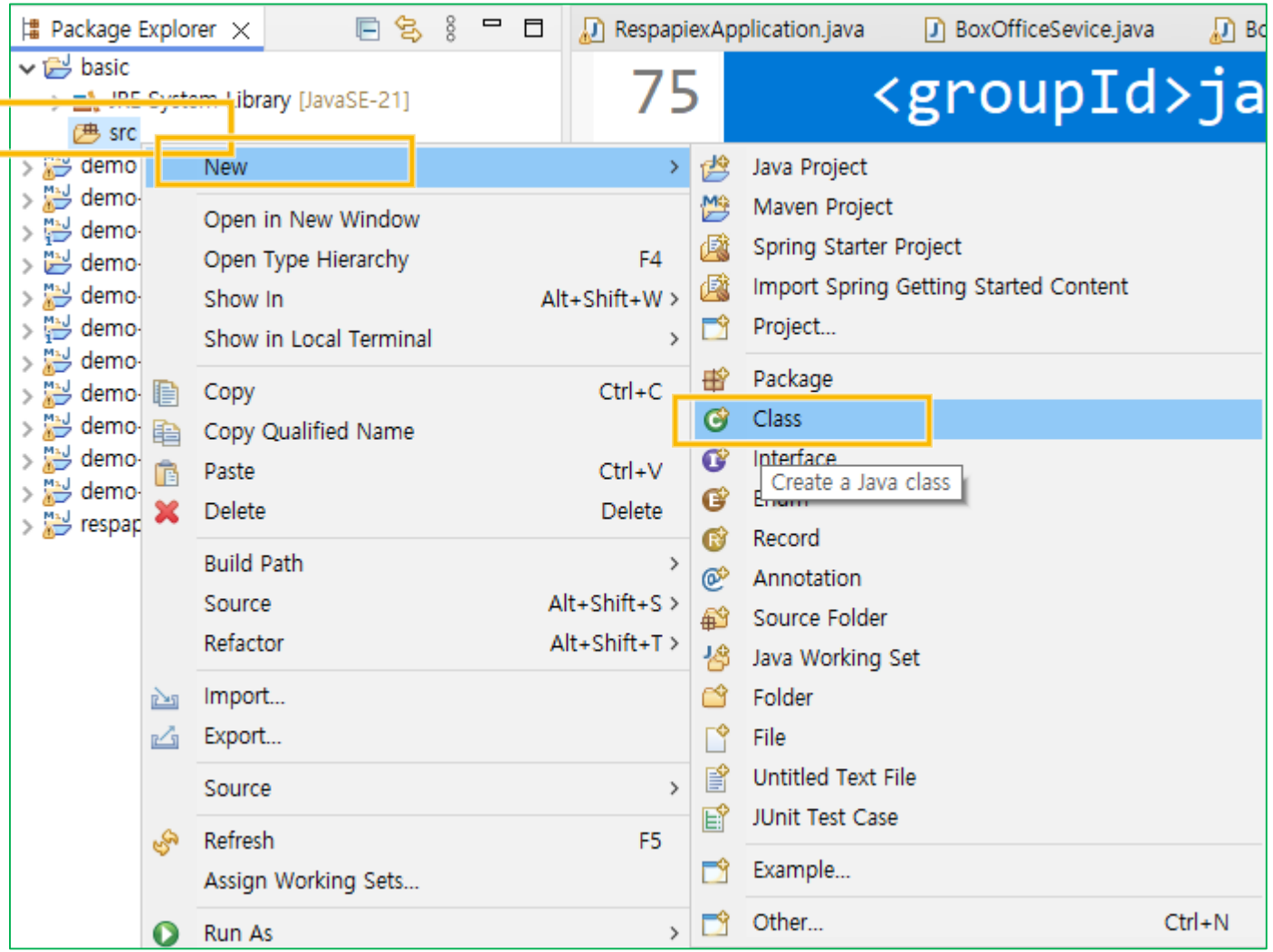


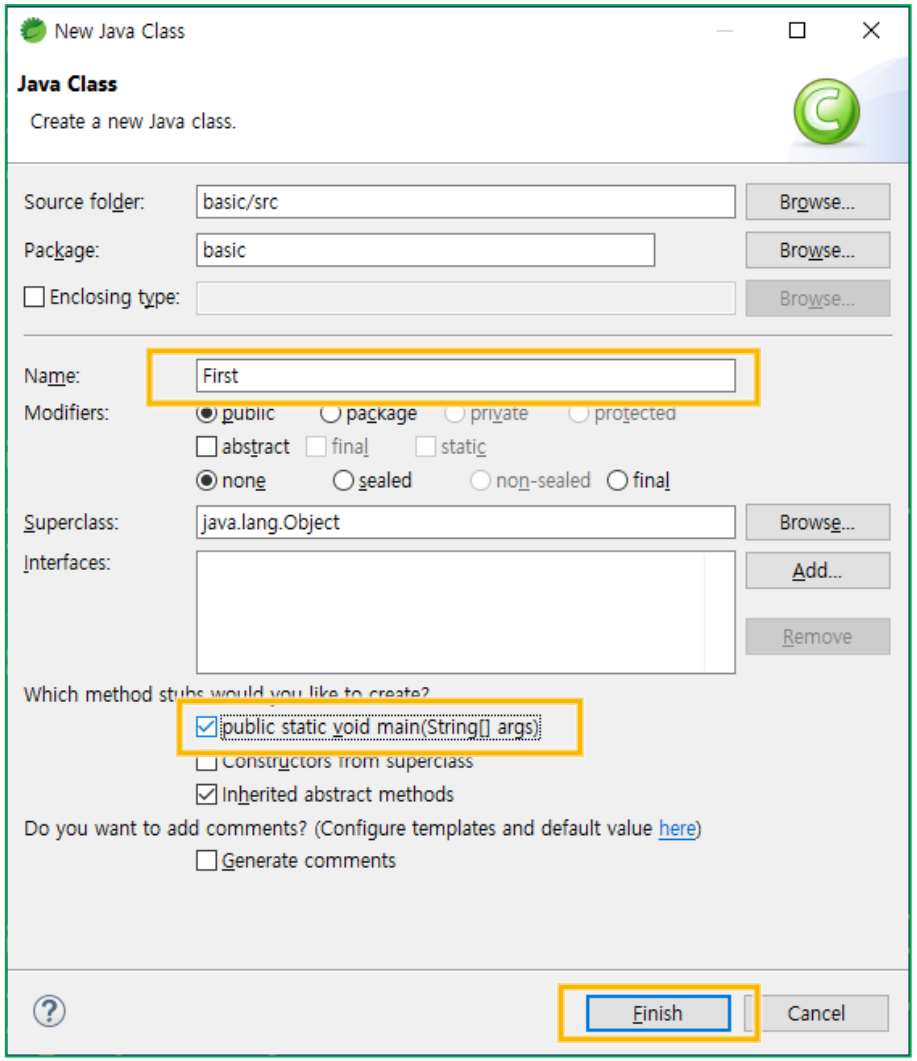




프로젝트 이름 :basic
Create module-info.java file 체크해제
Finish를 클릭하여 자바프로젝트 생성

basic프로젝트가 생성된 폴더에서 src선택-new-class를 선택
Name : First 라는 클래스명을 입력 – public static void main(String[] args)체크-Finish 선택하여 클래스를 생성함





자바 언어 이해

자바 클래스 구조

클래스는 함수와 변수로 구성되어 있다.

```
public class First{
public static void main(String[] arg){
System.out.println("hello java");
}
}
```

클래스를 정의

```
public class 클래스이름{

}
```

클래스 이름은 내 마음대로

클래스의 첫 문자는 대문자

클래스를 한글로 만들 수 있다.

```
public class Main {

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

}
```

한글로 쿠키 만들기

쿠키 객체 만들기

쿠키객체의 이름과 앞으로 가, 뒤로가 함수 만들기

메인 함수에서 쿠키 2개 만들기

쿠키를 이동하는 함수 만들기

```
public class 쿠키{
public String 이름;
public void  앞으로가라(){
System.out.println("["+이름+"]쿠키 앞으로 이동했습니다.");
}
public void  뒤로가라(){
System.out.println("["+이름+"]쿠키 뒤로 이동했습니다.");
}
public void 이름말해(){
System.out.println("나의 이름은 "+이름+"이야");
}

public void 우로이동(int mv){
if(mv>0 && mv<120){
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("   o   ");
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("   ^   ");
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("   |    ");
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("["+이름+"]");
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("   |   ");
for(int i=0;i<mv;i++) { System.out.print(" ");}
System.out.println("   ^  ");
}
}
```

프로그램 실행의 시작점

public , static 은 일단 신경 쓰지 말자.
모든 프로그램을 시작하기 위해서는
Main()함수가 반드시 존재해야 한다.

Main()함수는 다른 이름을 가질 수 없고
반드시 이름은 main이여야 한다.
()안에 String[] args는 필요한 전달 값이다.

```
Public class First{
Public static void main(String[] arg){
System.out.println(“hello java”);
}
}
```

컴 퓨 터 구조의 이해

1) 컴퓨터 장치

CPU : 연산기능

RAM : 데이터 저장 기능

기타 장치 : 모니터,키보드,HDD,프린터 …

2)컴퓨터는 메모리에 저장된 데이터를 이용하여 이 모든 장치의 표현을 다르게 할 뿐이다.

3)위의 기능을 이용하여 조건적, 반복적으로 수행하는 것이 컴퓨터의 역할이다.

변 수 와 함수의 이해

변수 : 데이터를 저장하는 공간

함수 : 필요한 것이 무엇?

필요한 요소를 이용하여 명령을 실행하라라는 명령의 집합이다.

```
public class Calculator {

	int plus(int a, int b){
		int result=a+b;
		return result;
	}
	int minus(int a, int b){
		return a-b;
	}
	int mux(int a, int b){
		return a*b;
	}
	float div(int a, int b){
		return (float)a/b;
	}
}
```

```
public class CalMain {

	public static void main(String[] args) {
		Calculator cal=new Calculator();
		int result=cal.plus(100, 200);
		System.out.println(result);

		System.out.println(cal.minus(200,100));
		System.out.println(cal.mux(10,200));
		System.out.println(cal.div(5,2));

	}

}
```

변수

변수 : 데이터를 저장하는 공간

메모리는 번지에 값이 저장되어 있음

하지만 메모리 번지를 사람이 외우기는 힘듦

이를 해결하기 위해 번지 대신 변수이름을 사용

프로시저형 함수

프로시저 함수는 단순히 동작만을 처리하는 함수이다.

자동차(객체)

앞으로 움직여라(go())　　10초 동안 움직여: go(10)

뒤로 움직여라(back())

정지해라(stop())

리턴형 함수

- 리턴형 함수는 필요한 값을 입력하면 하나의 결과를 돌려주는 함수이다.

- 두 수를 입력하면 더한 값을 출력하는 함수

더하라(10,20) -> 결과 30

int 합계변수=더하라(10,20); => int result=add(10,20)

합계변수(result)에는 30 값이 입력되어 있음

- 함수정의

```
int add(int a, int b){
int s=a+b;
return s;
}
```

- 정의한 함수 사용하기

```
int result=add(10,20)
```

```
public class Calculator {


	int plus(int a, int b){
		int result=a+b;
		return result;
	}
	int minus(int a, int b){
		return a-b;
	}
	int mux(int a, int b){
		return a*b;
	}
	float div(int a, int b){
		return (float)a/b;
	}
}
```

```
public class CalMain2 {

	public static void main(String[] args) {
		Calculator cal=new Calculator();

		//입력값이 직접 값을 입력하는 형태
		int result=cal.plus(10,20);
		System.out.println(result);

		//변수를 입력으로 사용한 형태
		int a=10; int b=20;
		result=cal.minus(b, a);
		System.out.println(result);

		//함수를 입력값으로 사용하는 형태
		result=cal.mux(cal.plus(10, 20), cal.minus(a, b));
		System.out.println(result);

		//혼합으로 사용하는 형태
		cal.div(a, cal.plus(a,b));

	}

}
```

순차적 프로그래밍

- Com:두 수를 입력하세요.
- My:두 수 입력
- Com:연산자를 선택하세요.
- My:연산자를 입력
- Com:결과를 출력

```
public class Main2 {
	public static void main(String[] args) {
		//두 수를 입력하고 입력된 두 수를 더하세요.\

		//su1이라는 변수에 10을 입력하고
		//su2이라는 변수에 20을 입력 후
		//result라는 변수에 su1과 su2의 값을 더해서
		//저장하고 출력하시오.
		int su1=10;
		int su2=20;
		int result=su1+su2;
		System.out.println("두수의 합은 "+result);
	}
}
```

```
import java.util.Scanner;

public class Main3 {

	public static void main(String[] args) {

		Scanner scan=new Scanner(System.in);

		System.out.print("첫번째 수를 입력하세요.");
		int su1=scan.nextInt();
		System.out.print("두번째 수를 입력하세요.");
		int su2=scan.nextInt();
		int result=su1+su2;
		System.out.println("두 수의 합:"+result);

	}

}
```

객체지향 프로그래밍

- 자동차는 모델명과 배기량cc의 정보를 가지고 있다.

자동차는 전진, 후진, 정지기능을 가지고 있다.

객체(클래스) 자동차{

문자열형 모델명;

숫자형 cc;

전진(){}

후진(){}

정지(){}

}

객체지향 프로그래밍

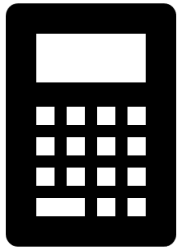
- 계산기는 더하기, 빼기, 곱하기, 나누기 기능을 가지고 있다.

```
public class Cal {

public int add(int a, int b) {
return a+b;
}
public int minus(int a, int b) {
return a-b;
}
public int mux(int a, int b) {
return a*b;
}
public float div(int a, int b) {
return (float)a/b;
}
}
```

앞에서 만든 클래스는 실제 메모리에 존재하는 것이 아니며
메모리에 해당객체를 올리기 위해서는 생성자 함수를 알아야 하고

실제 프로그램이 실행되는 함수(main함수)에서
객체를 생성하고(인스턴스생성) 기능을 사용해야 한다.

```
public class Main{
 public static void main(String[ ] args){
        Cal cal=new Cal();
        int result=cal.add(10,20);
}}
```

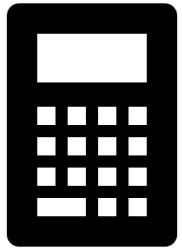


cal이라는 이름의 계산기 1개 생성

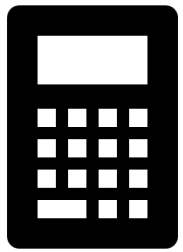
2개의 계산기 만들기

```
public class Main{
 public static void main(String[ ] args){
        Cal cal1=new Cal();
        int result1=cal1.add(10,20);

         Cal cal2=new Cal();
        int result2=cal2.add(10,20);

}
}
```

cal1계산기



cal2계산기



계산기에 display 기능을 추가하고 이 기능은 화면에 연산한 결과를 출력해주는 기능이므로 연산한 결과를 저장해주는 변수(속성)가 추가하도록 한다.

```
public class Cal {
int su1;
int su2;
Int result;

Int add(int su1, int su2){
        this.su1=su1;
        this.su2=su2
        result=su1+su2;
        return result;
}

void display(){
        System.out.println(result);
}
```

```
//사람 객체를 만들고
//사람은 이름과 나이와 키의 속성을 가진다.
//사람의 기능은 이름을 말할 수 있고
//사람의 기능은 나이를 말할 수 있고
//사람의 기능은 키를 말할 수 있습니다.
public class Person {
//클래스=속성+기능
	//속성=변수=맴버변수
	String name;
	int age;
	float height;


	//기능=함수=메소드
	public void speakname() {
		System.out.println("저의 이름은 "+name+"입니다.");
	}
	public void speakage() {
		System.out.println("저의 나이는 "+age+"살 입니다.");
	}
	public void speakheight() {
		System.out.println("저의 키는 "+height+"cm입니다.");
	}
}
```

```
public class PersonMain {

public static void main(String[] args) {
	Person p1=new Person();
	Person p2=new Person();
	Person p3=new Person();
	p1.name="철수";
	p1.age=20;
	p1.height=179.8f;
	p1.speakname();
	p1.speakage();
	p1.speakheight();
	//p2에 이름은 영희, 나이는 23, 키는 169.3으로
설정하고
	//이름을 말하기 기능을 시키세요.
	p2.name="영희"; p2.age=23; p2.height=169.3f;
	p2.speakname();
	}
}
```

콘솔입력함수

- 콘솔에서 한문자 입력
- System.in.read();
- 예) 한문자 입력받기

```
int key=System.in.read();
System.out.println(key);
```

- 다양한 형태의 문자를 콘솔에 입력하는 도구
- Scanner scan=new Scanner(System.in)
- 예)문자열 입력받기 , 숫자 입력받기

```
Scanner scan=new Scanner(System.in);
String str=scan.next(); int su=scan.nextInt();
```

- 문자를 한 줄에 여러 개 입력 받기
- 10 20 add

```
int su1=scan.nextInt();
int su2=scan.nextInt();
String command=scan.nextLine();
```

콘솔출력함수

- ()안에 필요한 문자열, 숫자등을 입력하면 화면에 출력됨.

System.out.println( )

- 1바이트 이상의 데이터를 출력하는 함수

System.out.write(); System.out.flush()

콘솔입력예제

```
public static void main(String[] args) {
Scanner scan=new Scanner(System.in);
while(true) {
int su1=scan.nextInt();
int su2=scan.nextInt();
String command=scan.nextLine(); //문자열 앞에 스페이스를 포함
//String command=scan.next(); //스페이스를 포함하지 않음
System.out.println(su1);
System.out.println(su2);
System.out.println(command);
System.out.println("프로그램 다시 실행");
}


}
```

조건문

- 조건문은 명령을 실행할지 여부를 결정한다.
- 예시)입력한 값이 y문자일 경우 프로그램을 종료한다.

```
Scanner scan=new Scanner(System.in);
String answer=scan.next();
if(answer.equals(“y”)){ system.out.println(“프로그램종료”); }
```

- 조건문의 다양한 형태

```
if(조건) { }
switch() { }
```

반복문

- 반복문은 패턴을 분석해야한다.
- 1에서 10까지 출력하시오 라는 명령을 반복문에서 확인할 사항
- 시작값 1, 종료값 10, 증가값 1
- for(int i=1;i<=10;i++){ System.out.println(i);}

- 반복문의 다양한 형태
- for(int i=0;i<10;i++){ }
- for(String s: arr) { }
- arr.forEach((s) -> System.out.println(s));
- While() { }

```
public class Calculator {


	int plus(int a, int b){
		int result=a+b;
		return result;
	}
	int minus(int a, int b){
		return a-b;
	}
	int mux(int a, int b){
		return a*b;
	}
	float div(int a, int b){
		return (float)a/b;
	}
}
```

```
import java.util.Scanner;

public class CalMain3 {

	public static void main(String[] args) {

		Scanner scan=new Scanner(System.in);
		Calculator cal=new Calculator();

		while(true) {
		System.out.print("연산할 첫번째 수를 입력하세요.");
		int su1=scan.nextInt();

		System.out.print("연산자를 입력하세요.(+,-,*,/)");
		String op=scan.next();

		System.out.print("연산할 두번째 수를 입력하세요.");
		int su2=scan.nextInt();

		float result=0;
		if(op.equals("+")) {
			result=(float)cal.plus(su1, su2);
		}else if(op.equals("-")) {
			result=(float)cal.minus(su1, su2);
```

입력스트림

- 스트림은 데이터의 흐름을 의미한다.
- 입력스트림은 메모리에 데이터가 입력되는 흐름을 의미한다.
- **InputStream in=new InputStream();**

출력스트림

- 스트림은 데이터의 흐름을 의미한다.
- 출력스트림은 데이터가 메모리에서 출력되는 것을 의미한다.
- **OutputStream out=new OutputStream();**