1. 클래스와 데이터

#1.인강/0.자바/2.자바-기본

- /프로젝트 환경 구성
- /클래스가 필요한 이유
- /클래스 도입
- /객체 사용
- /클래스, 객체, 인스턴스 정리
- /배열 도입 - 시작
- /배열 도입 - 리펙토링
- /문제와 풀이
- /정리

프로젝트 환경 구성

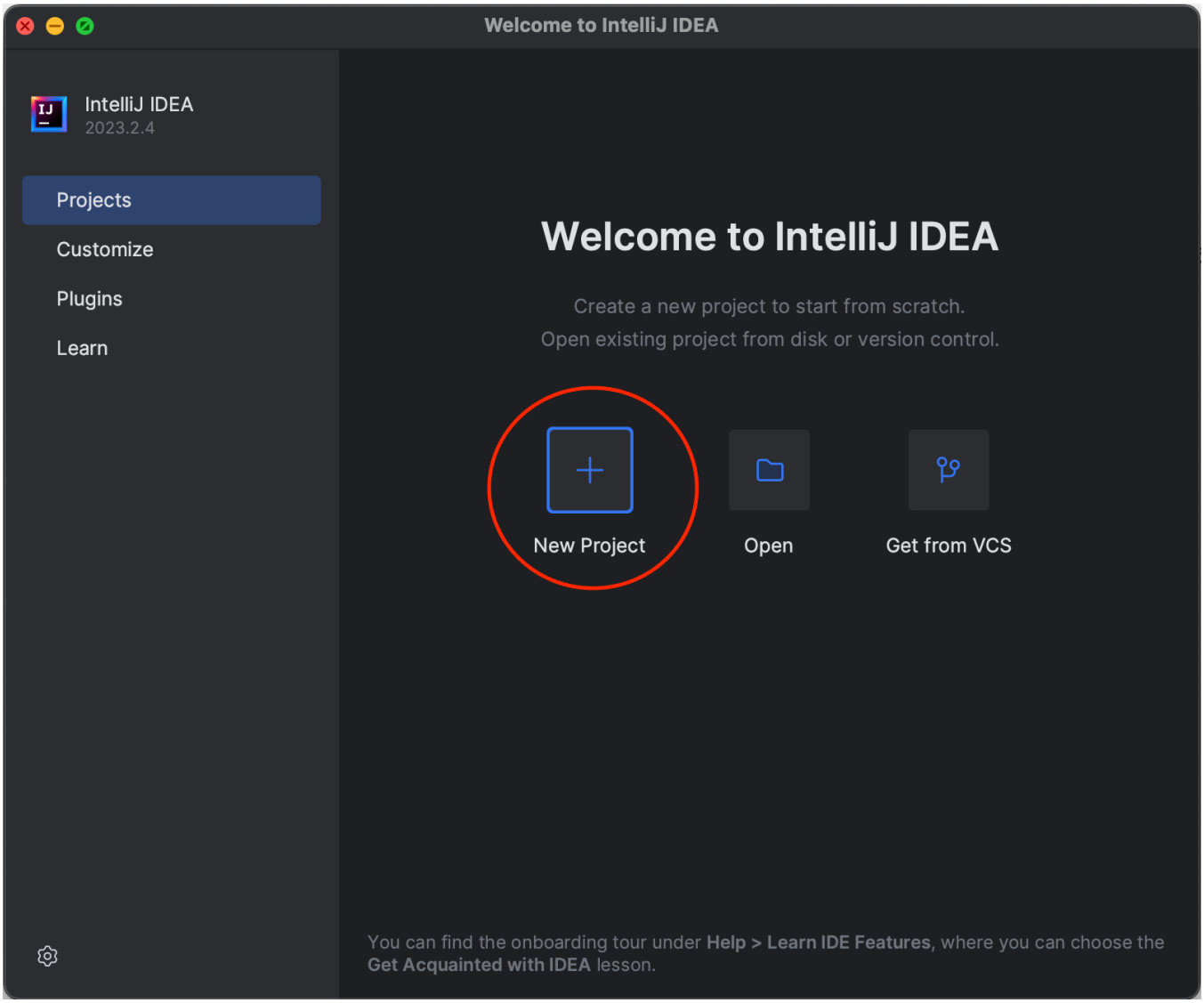
자바 입문편에서 인텔리제이 설치, 선택 이유 설명

프로젝트 환경 구성에 대한 자세한 내용은 자바 입문편 참고

여기서는 입문편을 들었다는 가정하에 자바 기본편 설정 진행

인텔리제이 실행하기

New Project



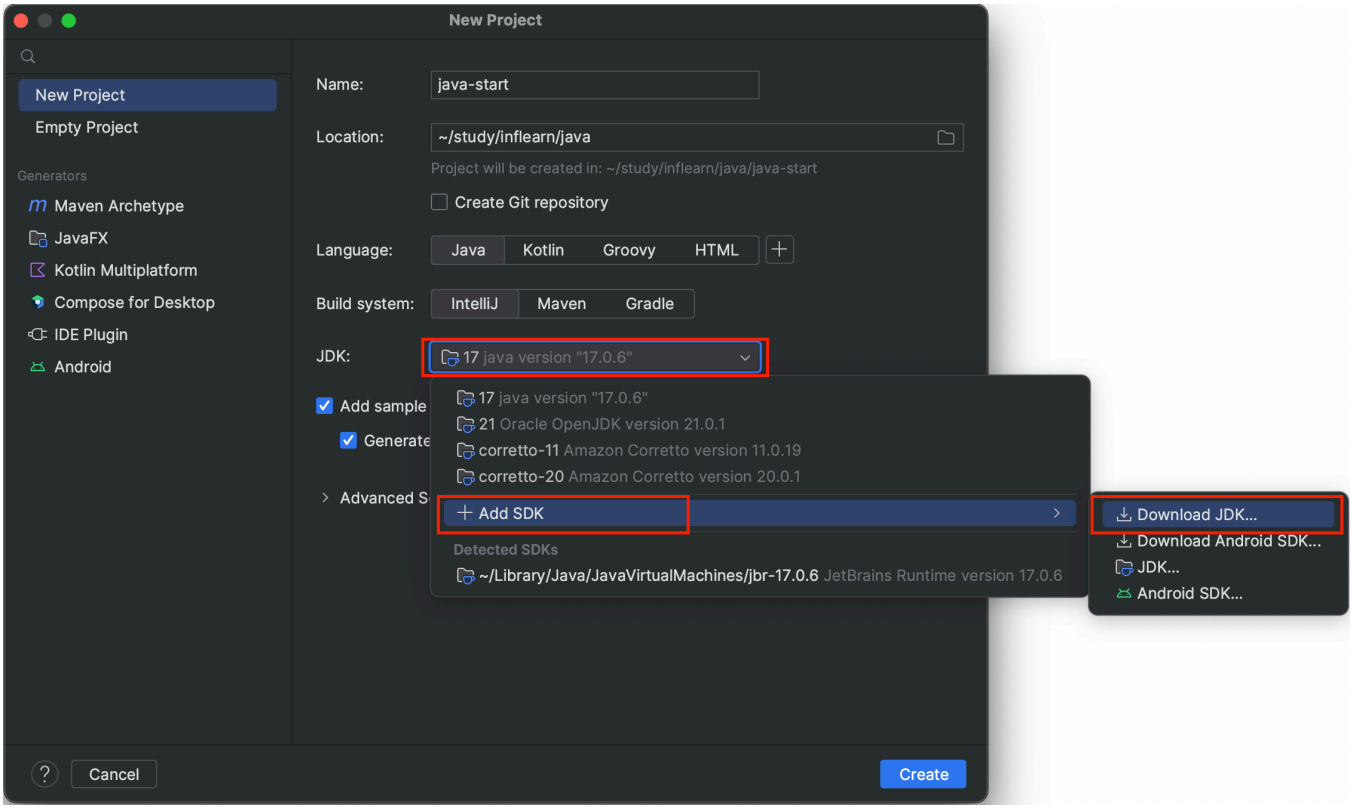
- New Project를 선택해서 새로운 프로젝트를 만들자

New Project 화면

- Name:
  - 자바 입문편 강의: java-start
  - 자바 기본편 강의: **java-basic**
- Location: 프로젝트 위치, 임의 선택
- Create Git repository 선택하지 않음
- Language: Java
- Build system: IntelliJ
- JDK: 자바 버전 17 이상
- `Add sample code` 선택

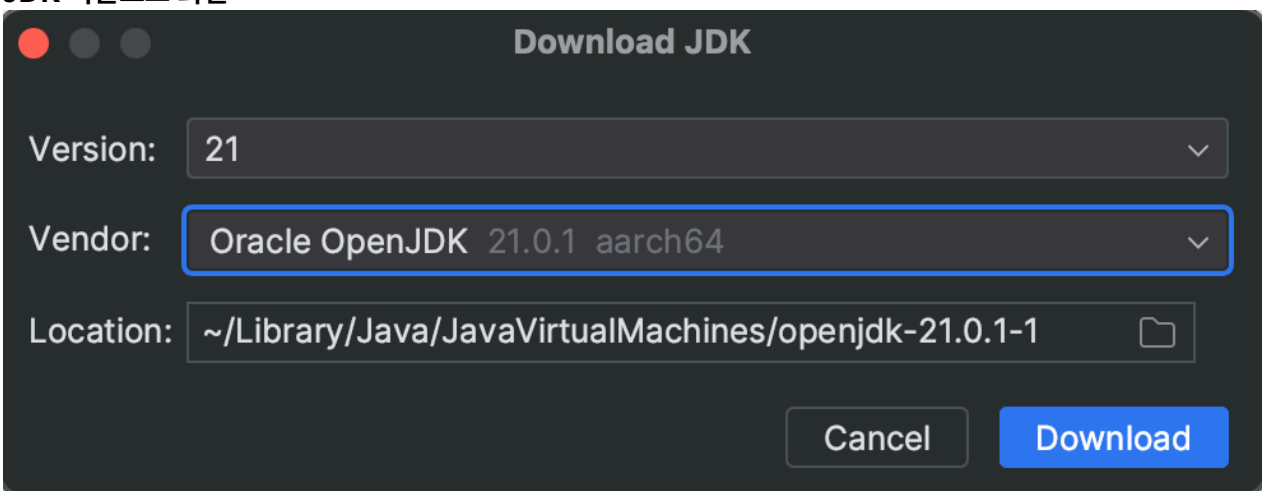
JDK 다운로드 화면 이동 방법

자바로 개발하기 위해서는 JDK가 필요하다. JDK는 자바 프로그래머를 위한 도구 + 자바 실행 프로그램의 묶음이다.



- **Name:**
  - 자바 입문편 강의: java-start
  - 자바 기본편 강의: **java-basic**

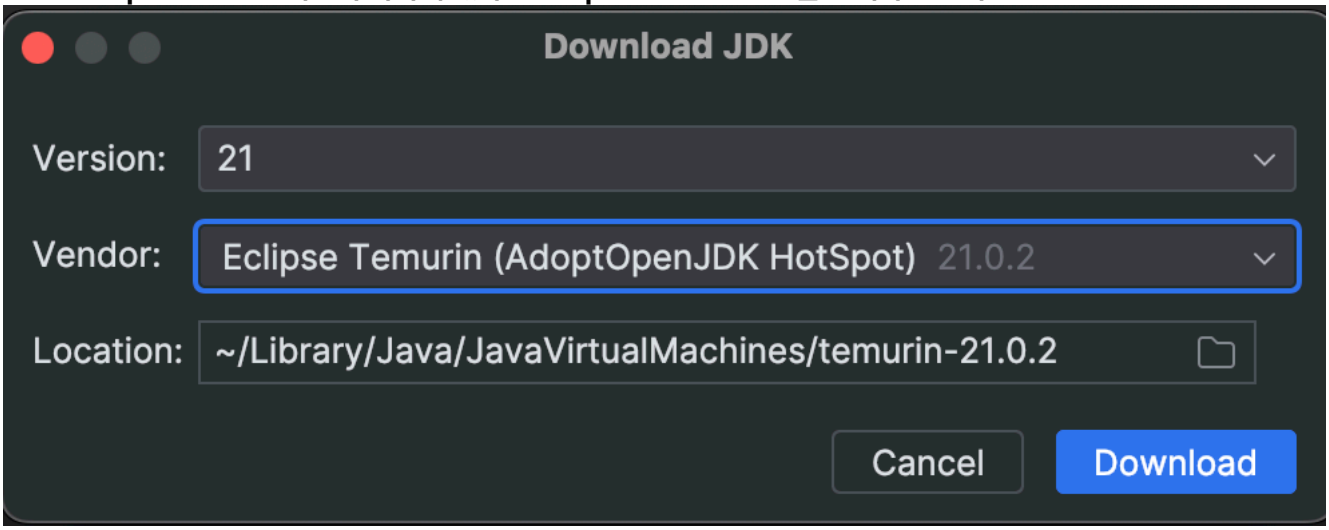
JDK 다운로드 화면



- Version: 21을 선택하자.
- Vendor: Oracle OpenJDK를 선택하자. 다른 것을 선택해도 된다.
  - aarch64: 애플 M1, M2, M3 CPU 사용시 선택, 나머지는 뒤에 이런 코드가 붙지 않은 JDK를 선택하면 된다.
- Location: JDK 설치 위치, 기본값을 사용하자.

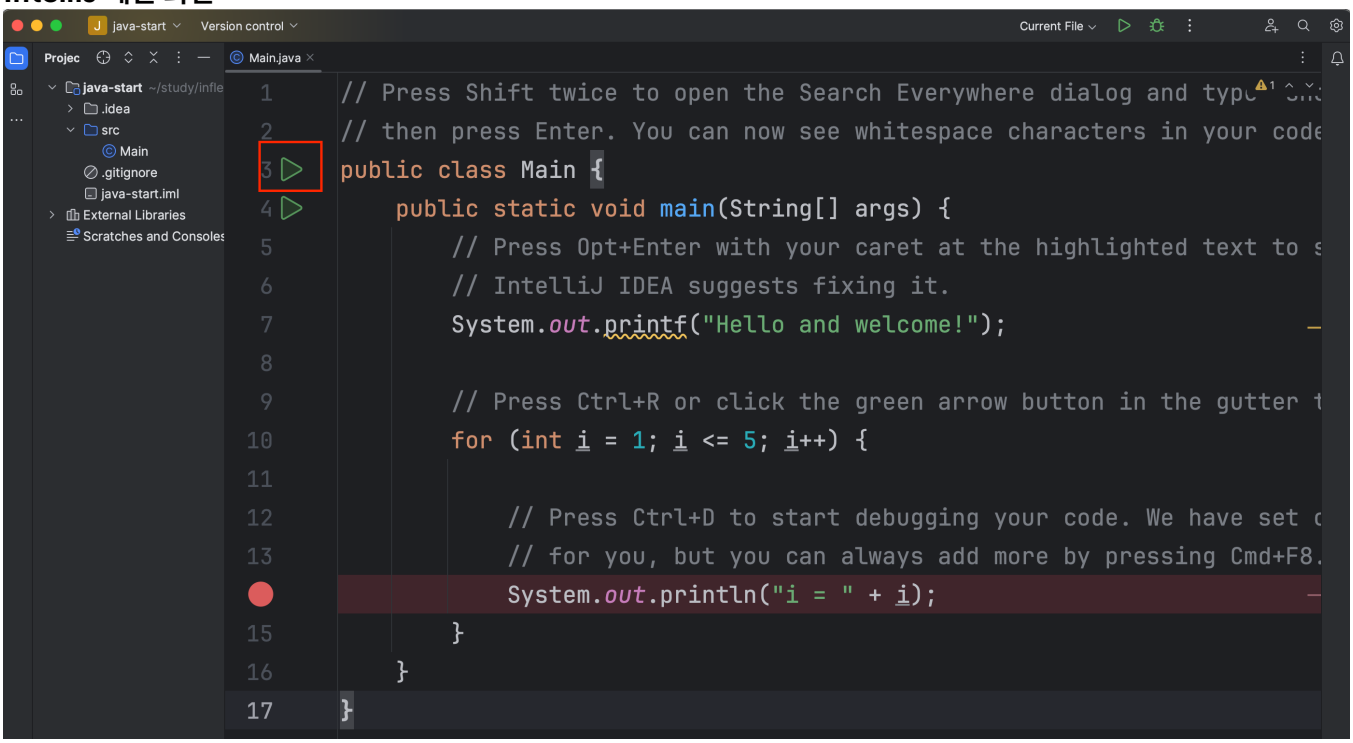
주의 - 변경 사항

Oracle OpenJDK 21 버전이 목록에 없다면 Eclipse Temurin 21을 선택하면 된다.



`Download` 버튼을 통해서 다운로드 JDK를 다운로드 받는다.
다운로드가 완료 되고 이전 화면으로 돌아가면 `Create` 버튼 선택하자. 그러면 다음 IntelliJ 메인 화면으로 넘어간다.

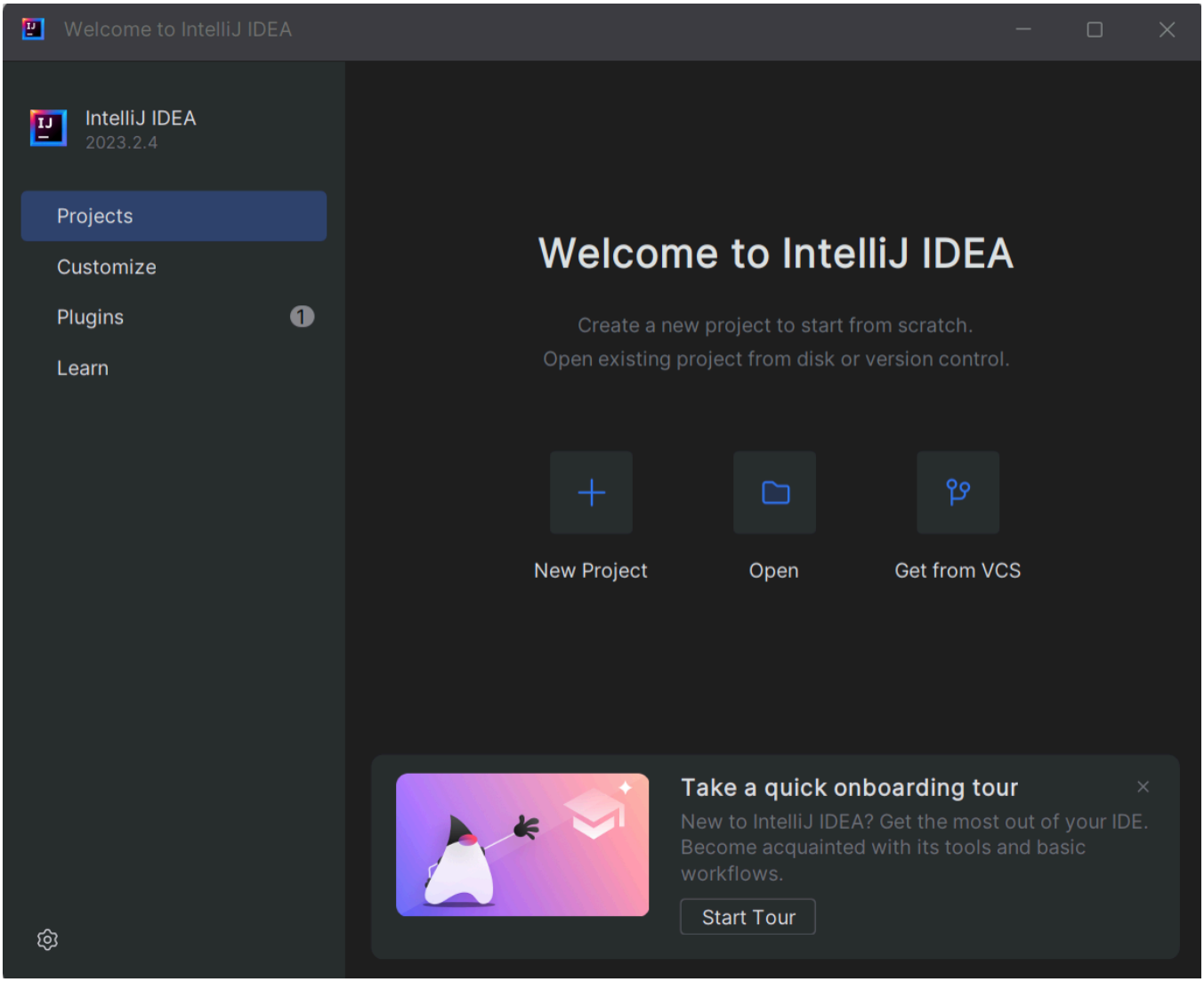
IntelliJ 메인 화면



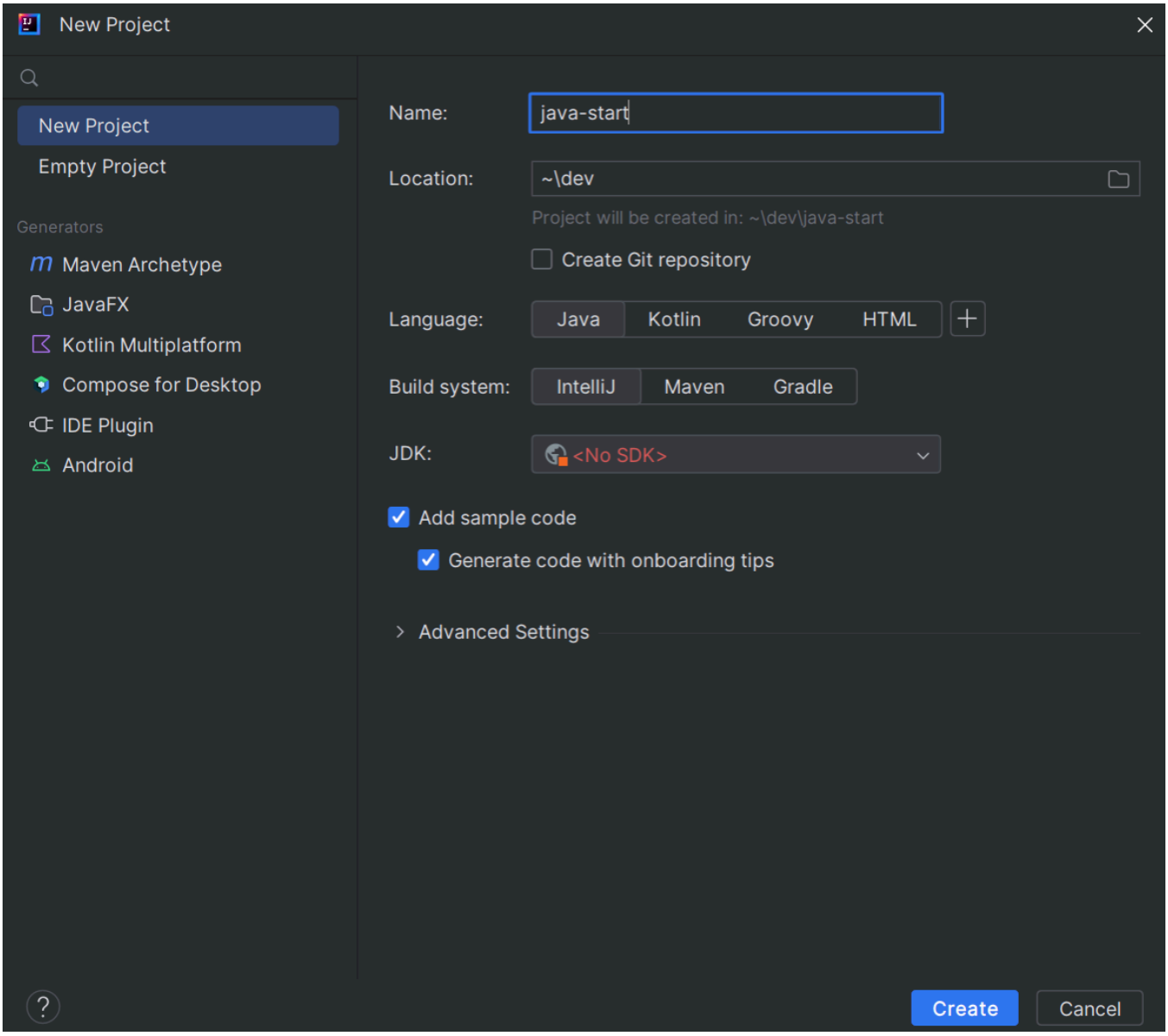
- 앞서 `Add sample code` 선택해서 샘플 코드가 만들어져 있다.
- 위쪽에 빨간색으로 강조한 초록색 화살표 버튼을 선택하고 `Run 'Main.main()` 버튼을 선택하면 프로그램이 실행된다.

윈도우 사용자 추가 설명서

윈도우 사용자도 Mac용 IntelliJ와 대부분 같은 화면이다. 일부 다른 화면 위주로 설명하겠다.

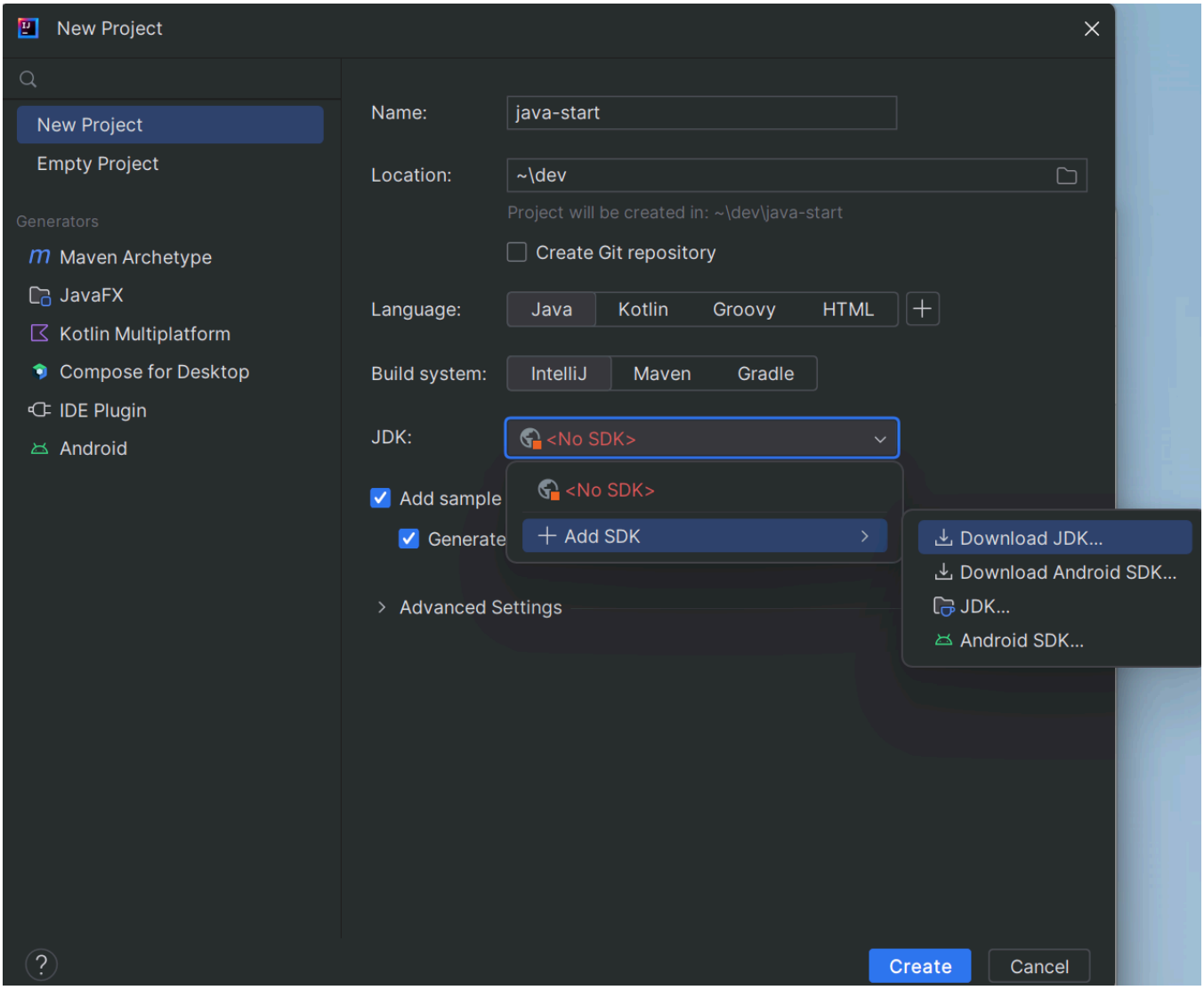


- 프로그램 시작 화면
- New Project 선택

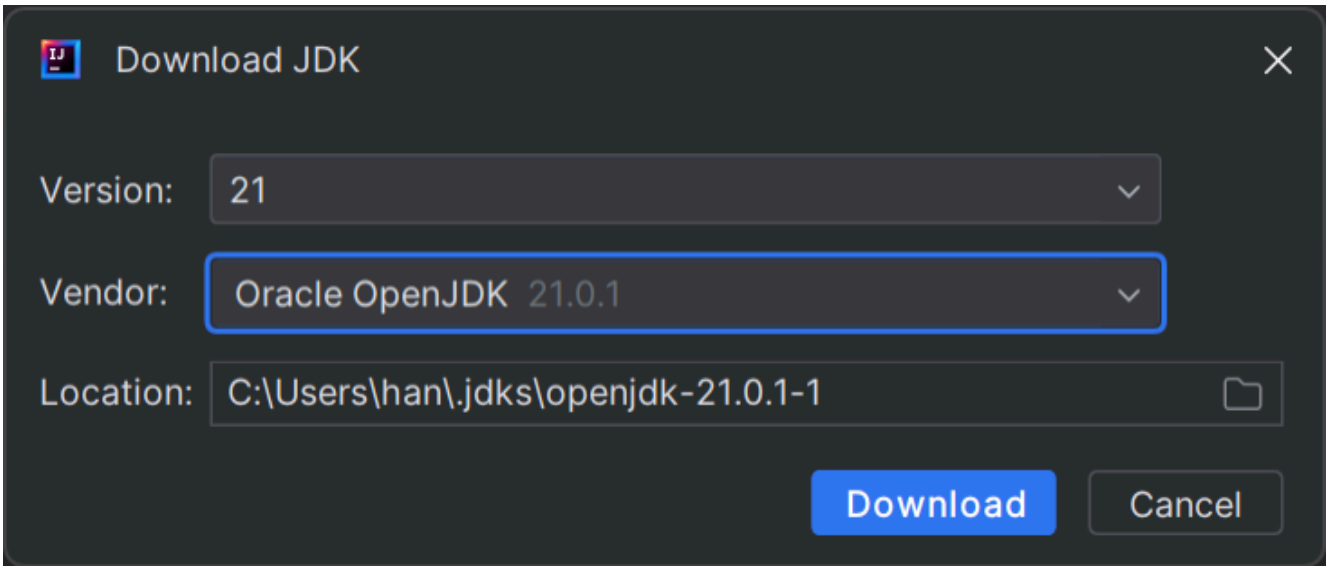


New Project 화면

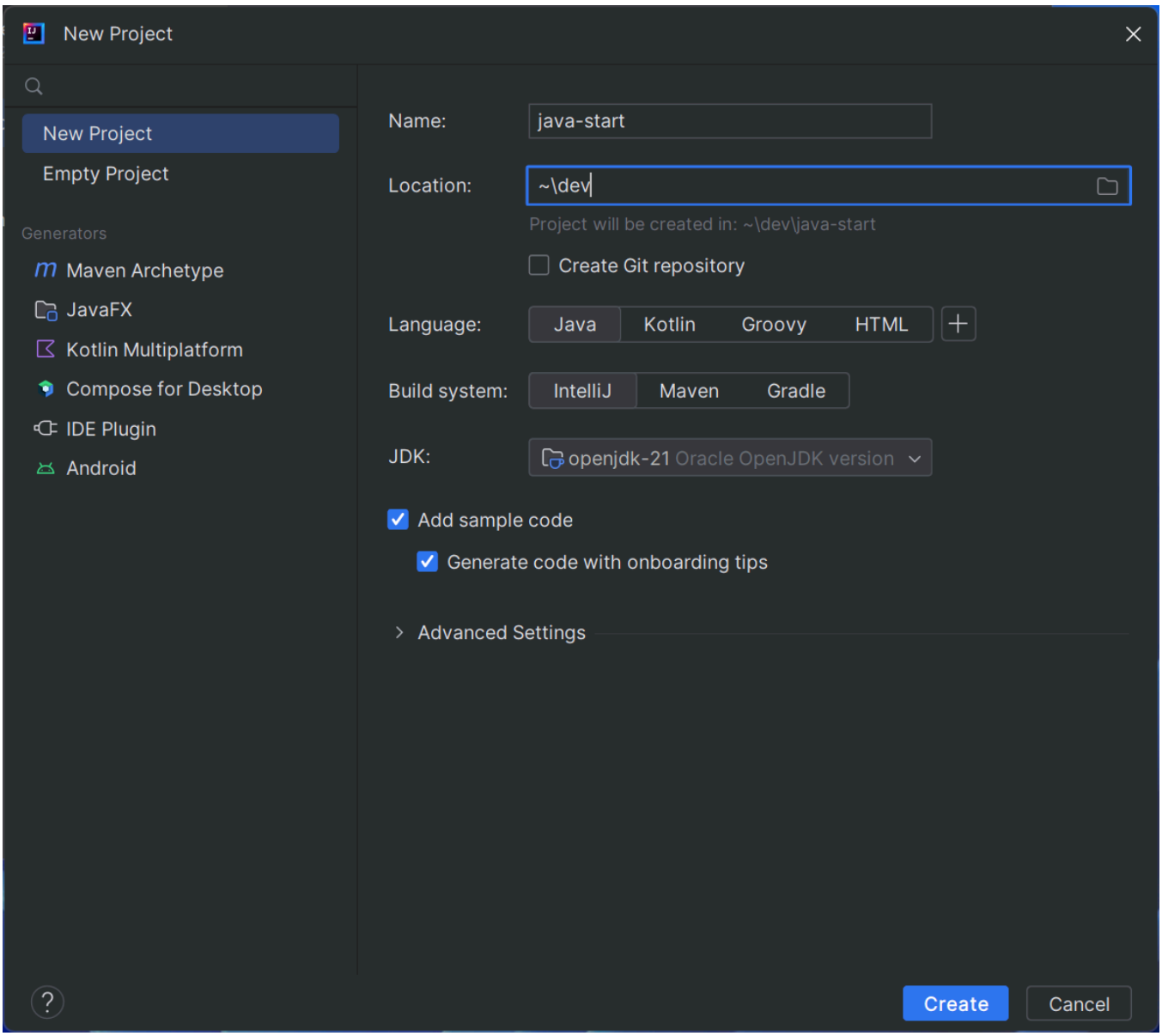
- **Name:**
  - 자바 입문편 강의: java-start
  - 자바 기본편 강의: **java-basic**
- Location: 프로젝트 위치, 임의 선택
- Create Git repository 선택하지 않음
- Language: Java
- Build system: IntelliJ
- JDK: 자바 버전 17 이상
- `Add sample code` 선택



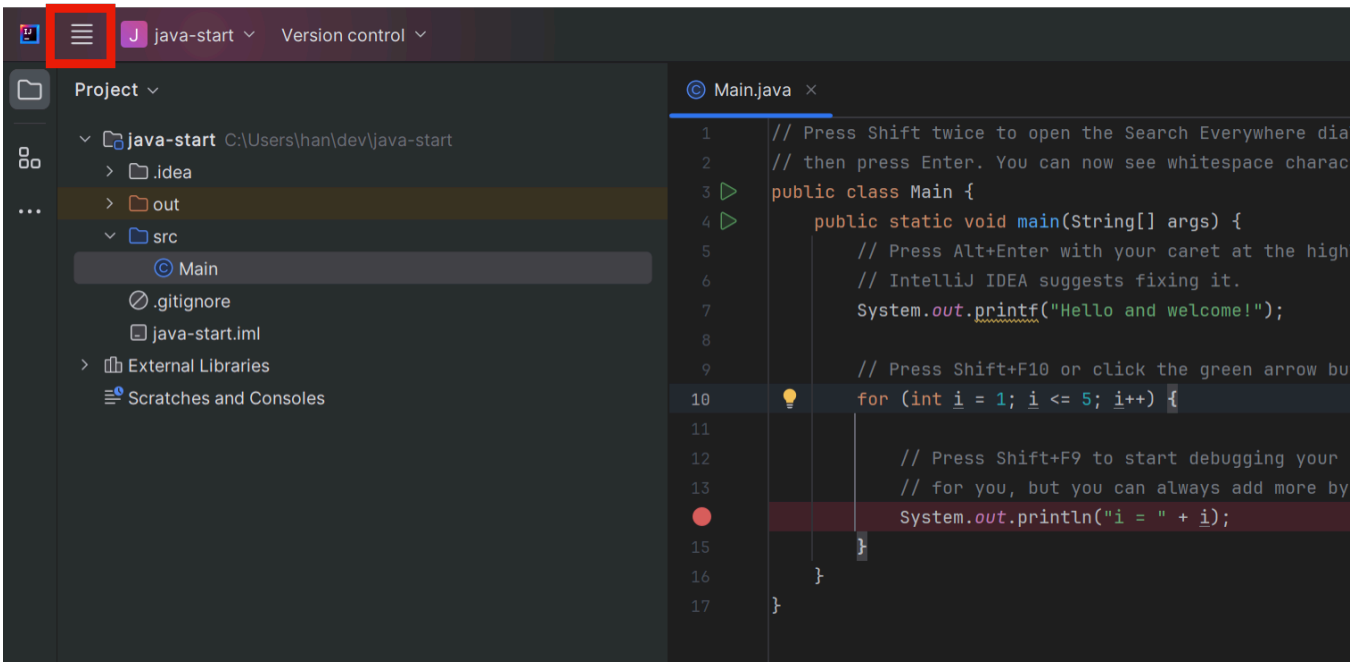
JDK 설치는 Mac과 동일하다.



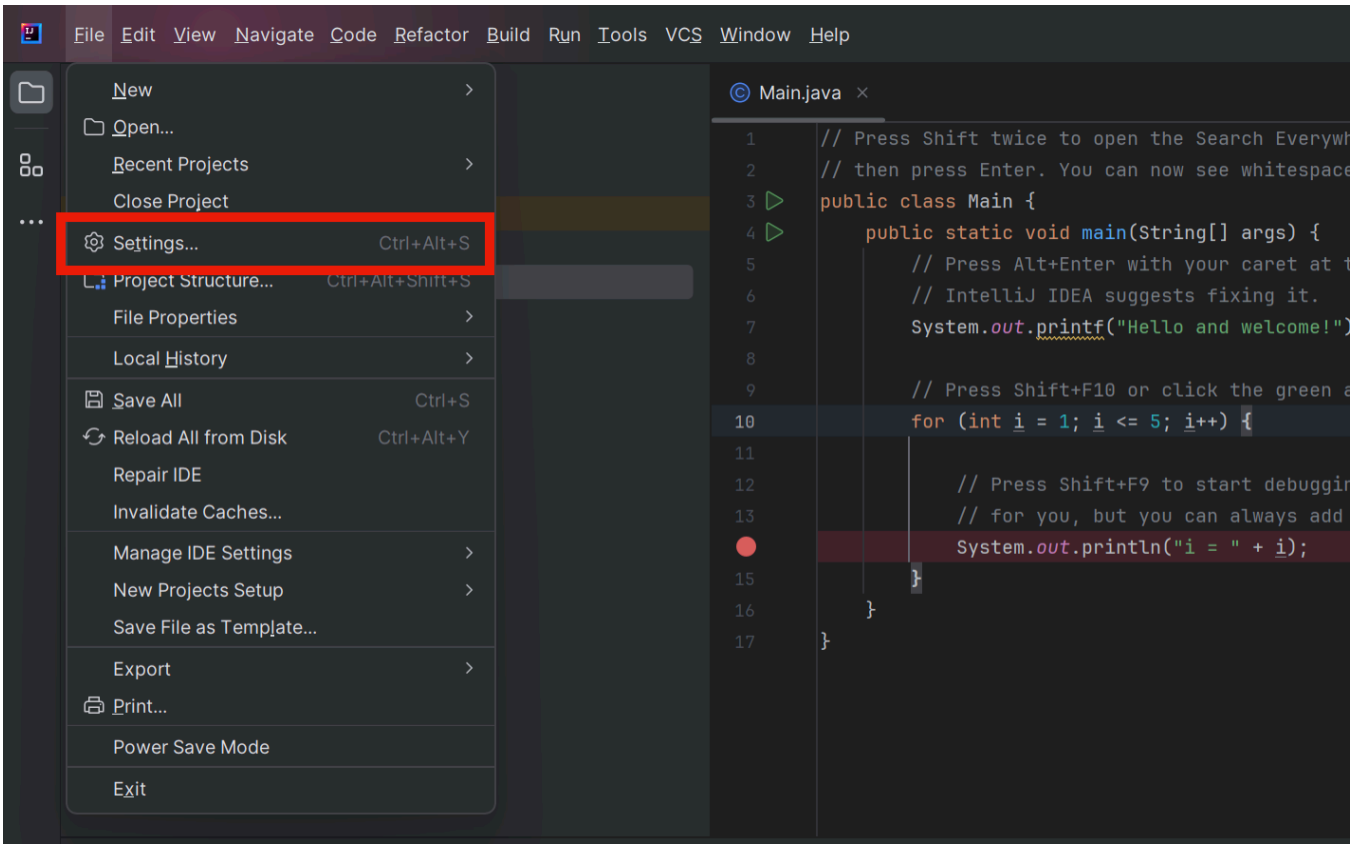
- Version: 21
- Vendor: Oracle OpenJDK
- Location은 가급적 변경하지 말자.



- New Project 완료 화면



- 윈도우는 메뉴를 확인하려면 왼쪽 위의 빨간색 박스 부분을 선택해야 한다.



- Mac과 다르게 `Settings...` 메뉴가 `File`에 있다. 이 부분이 Mac과 다르므로 유의하자.

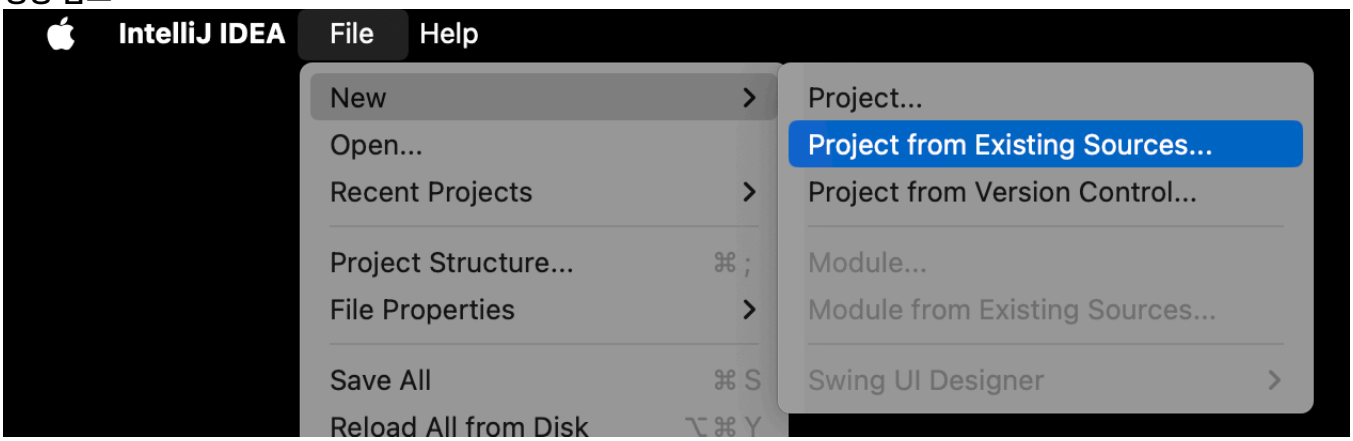
한글 언어팩 → 영어로 변경

- IntelliJ는 가급적 한글 버전 대신, 영문 버전을 사용하자. 개발하면서 필요한 기능들을 검색하게 되는데, 영문으로 된 자료가 많다. 이번 강의도 영문을 기준으로 진행한다.

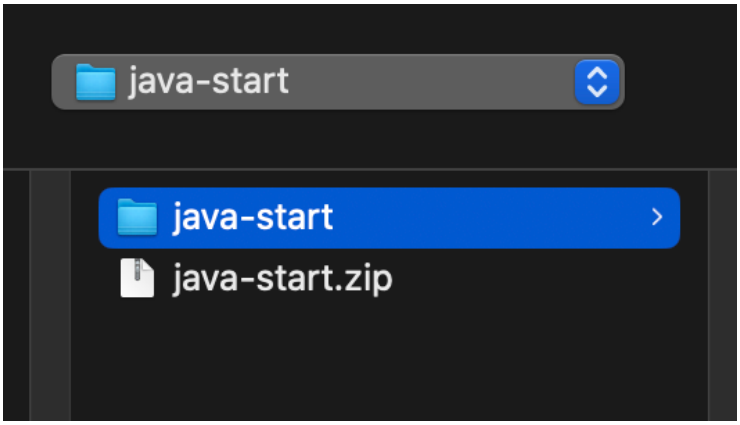
- 만약 한글로 나온다면 다음과 같이 영문으로 변경하자.
- **Mac**: IntelliJ IDEA(메뉴) ➜ Settings... ➜ Plugins ➜ Installed
- **윈도우**: File ➜ Settings... ➜ Plugins ➜ Installed
  - Korean Language Pack 체크 해제
  - OK 선택후 IntelliJ 다시 시작

다운로드 소스 코드 실행 방법

영상 참고

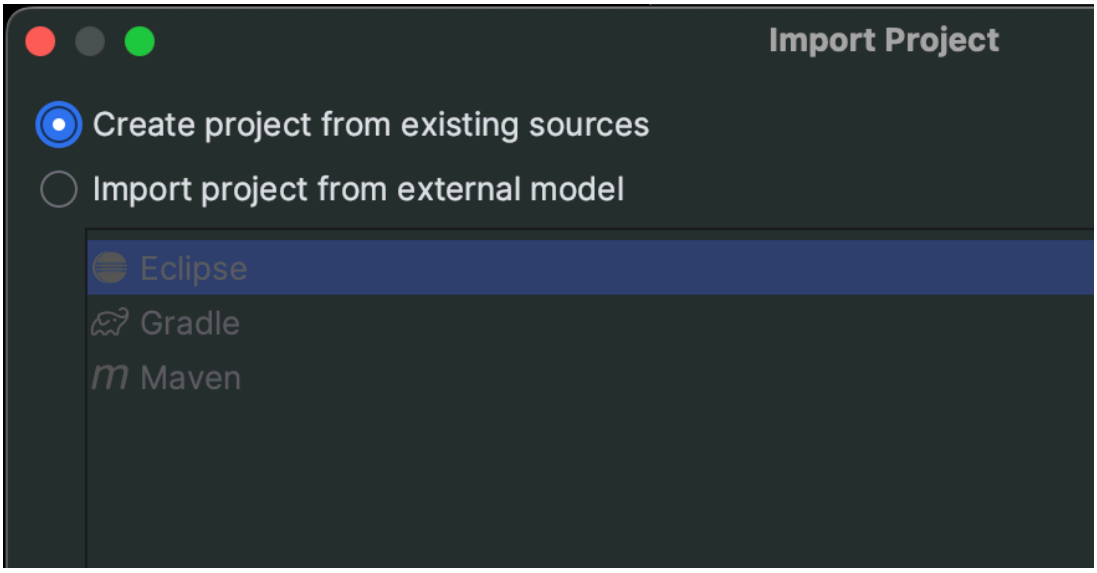


`File -> New -> Project from Existing Sources...` 선택



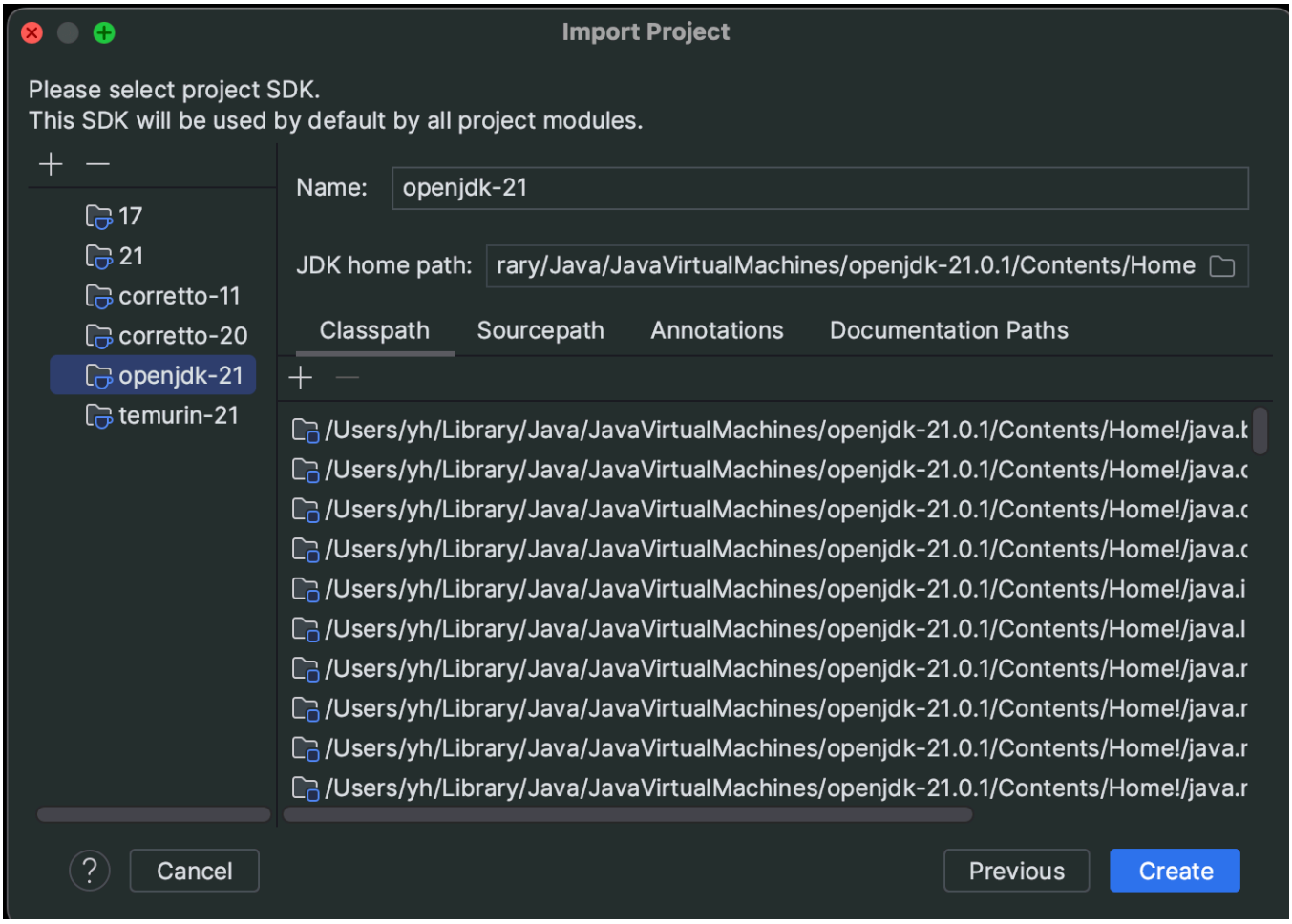
압축을 푼 프로젝트 폴더 선택

- 자바 입문 강의 폴더: java-start
- 자바 기본 강의 폴더: **java-basic**

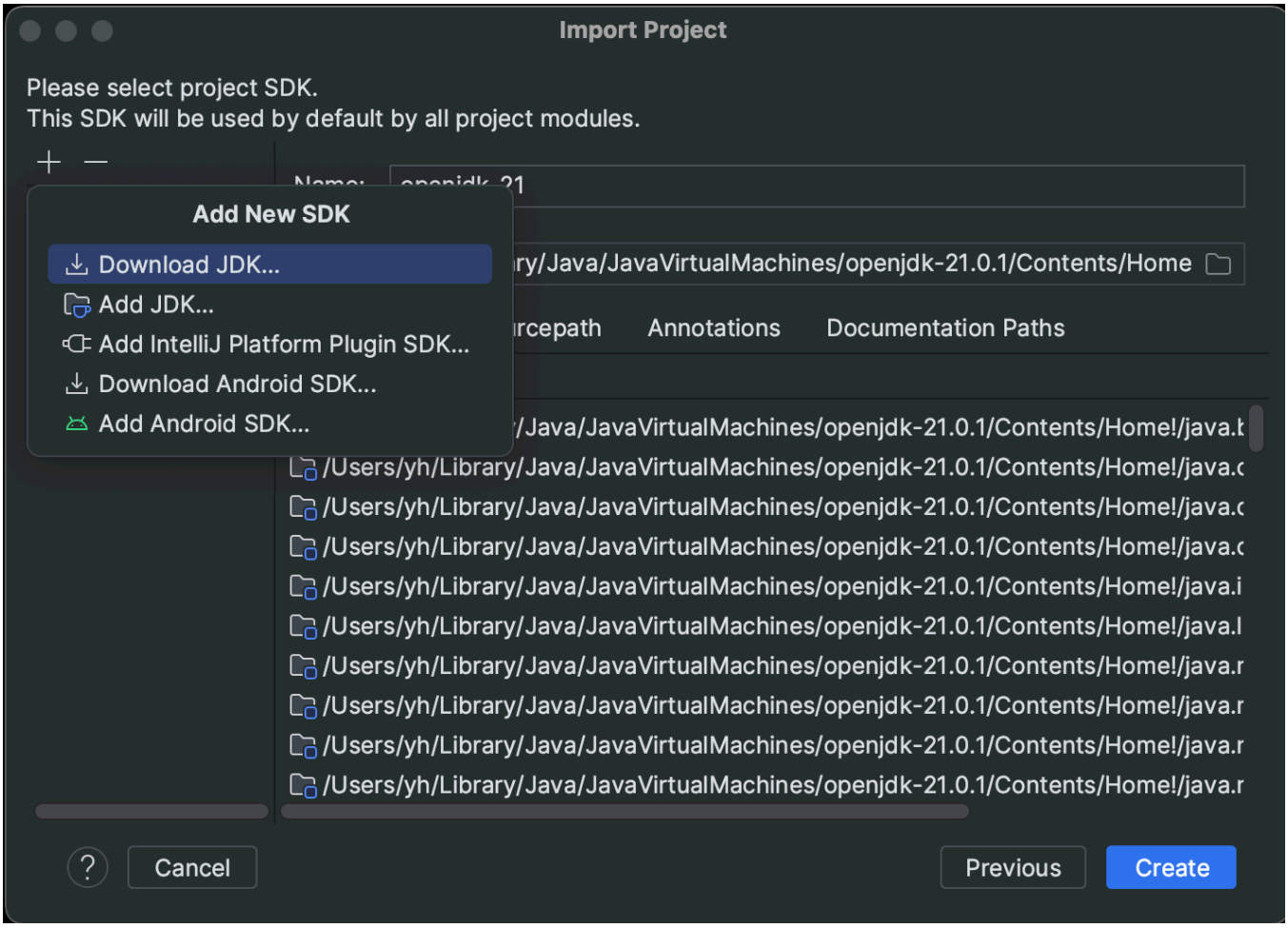


`Create project from existing sources` 선택

이후 계속 `Next` 선택



openjdk-21 선택



만약 JDK가 없다면 왼쪽 상단의 + 버튼을 눌러서 openjdk 21 다운로드 후 선택

이후 `Create` 버튼 선택

클래스가 필요한 이유

자바 세상은 클래스와 객체로 이루어져 있다. 그만큼 클래스와 객체라는 개념은 중요하다. 그런데 클래스와 객체는 너무 많은 내용을 포함하고 있어서 한번에 이해하기 쉽지 않다. 여기서는 클래스와 객체라는 개념이 왜 필요한지 부터 시작해서, 클래스가 어떤 방식으로 발전하면서 만들어졌는지 점진적으로 알아보겠다.

먼저 클래스가 왜 필요한지 이해하기 위해 다음 문제를 풀어보자.
지금까지 학습한 내용을 기반으로 문제를 풀어보면 된다.

문제: 학생 정보 출력 프로그램 만들기

당신은 두 명의 학생 정보를 출력하는 프로그램을 작성해야 한다. 각 학생은 이름, 나이, 성적을 가지고 있다.

요구 사항:

1. 첫 번째 학생의 이름은 "학생1", 나이는 15, 성적은 90입니다.
2. 두 번째 학생의 이름은 "학생2", 나이는 16, 성적은 80입니다.
3. 각 학생의 정보를 다음과 같은 형식으로 출력해야 합니다: `"이름: [이름] 나이: [나이] 성적: [성적]"`
4. 변수를 사용해서 학생 정보를 저장하고 변수를 사용해서 학생 정보를 출력해야 합니다.

예시 출력:

```
이름: 학생1 나이: 15 성적: 90
이름: 학생2 나이: 16 성적: 80
```

변수를 사용해서 이 문제를 풀어보면 다음과 같다.

ClassStart1 - 변수 사용

```
package class1;

public class ClassStart1 {
    public static void main(String[] args) {
        String student1Name = "학생1";
        int student1Age = 15;
        int student1Grade = 90;

        String student2Name = "학생2";
        int student2Age = 16;
        int student2Grade = 80;

        System.out.println("이름:" + student1Name + " 나이:" + student1Age + " 성
적:" + student1Grade);
        System.out.println("이름:" + student2Name + " 나이:" + student2Age + " 성
적:" + student2Grade);
    }
}
```

학생 2명을 다루어야 하기 때문에 각각 다른 변수를 사용했다. 이 코드의 문제는 학생이 늘어날 때 마다 변수를 추가로 선언해야 하고, 또 출력하는 코드도 추가해야 한다.
이런 문제를 어떻게 해결할 수 있을까? 앞서 배운 배열을 사용하면 문제를 해결할 수 있다.

문제: 이전 문제에 배열을 사용하자

이전 문제에 배열을 적용하자. 그래서 학생이 추가되어도 코드 변경을 최소화 해보자.

ClassStart2

```
package class1;

public class ClassStart2 {
    public static void main(String[] args) {
        String[] studentNames = {"학생1", "학생2"};
        int[] studentAges = {15, 16};
        int[] studentGrades = {90, 80};

        for (int i = 0; i < studentNames.length; i++) {
            System.out.println("이름:" + studentNames[i] + " 나이:" +
studentAges[i] + " 성적:" + studentGrades[i]);
        }
    }
}
```

배열을 사용한 덕분에 학생이 추가되어도 배열에 학생의 데이터만 추가하면 된다. 이제 변수를 더 추가하지 않아도 되고, 출력 부분의 코드도 그대로 유지할 수 있다.

학생 추가 전

```
String[] studentNames = {"학생1", "학생2"};
int[] studentAges = {15, 16};
int[] studentGrades = {90, 80};
```

학생 추가 후

```
String[] studentNames = {"학생1", "학생2", "학생3", "학생4", "학생5"};
int[] studentAges = {15, 16, 17, 10, 16};
int[] studentGrades = {90, 80, 100, 80, 50};
```

배열 사용의 한계

배열을 사용해서 코드 변경을 최소화하는데는 성공했지만, 한 학생의 데이터가 `studentNames[]`, `studentAges[]`, `studentGrades[]` 라는 3개의 배열에 나누어져 있다. 따라서 데이터를 변경할 때 매우 조심해서 작업해야 한다. 예를 들어서 학생 2의 데이터를 제거하려면 각각의 배열마다 학생2의 요소를 정확하게 찾아서 제거해주어야 한다.

학생2 제거

```
String[] studentNames = {"학생1", "학생3", "학생4", "학생5"};
int[] studentAges = {15, 17, 10, 16};
int[] studentGrades = {90, 100, 80, 50};
```

한 학생의 데이터가 3개의 배열에 나누어져 있기 때문에 3개의 배열을 각각 변경해야 한다. 그리고 한 학생의 데이터를 관리하기 위해 3개 배열의 인덱스 순서를 항상 정확하게 맞추어야 한다. 이렇게 하면 특정 학생의 데이터를 변경할 때 실수할 가능성이 매우 높다.
이 코드는 컴퓨터가 볼 때는 아무 문제가 없지만, 사람이 관리하기에는 좋은 코드가 아니다.

정리

지금처럼 이름, 나이, 성적을 각각 따로 나누어서 관리하는 것은 사람이 관리하기 좋은 방식이 아니다.
사람이 관리하기 좋은 방식은 학생이라는 개념을 하나로 묶는 것이다. 그리고 각각의 학생 별로 본인의 이름, 나이, 성적을 관리하는 것이다.

클래스 도입

앞서 이야기한 문제를 클래스를 도입해서 해결해보자.
클래스를 사용해서 학생이라는 개념을 만들고, 각각의 학생 별로 본인의 이름, 나이, 성적을 관리하는 것이다.
우선 코드를 작성하고 실행해보자.

Student 클래스

```
package class1;

public class Student {
    String name;
    int age;
    int grade;
}
```

`class` 키워드를 사용해서 학생 클래스(`Student`)를 정의한다. 학생 클래스는 내부에 이름(`name`), 나이(`age`), 성적(`grade`) 변수를 가진다.

이렇게 **클래스에 정의한 변수들을 멤버 변수, 또는 필드**라 한다.

- **멤버 변수(Member Variable)**: 이 변수들은 특정 클래스에 소속된 멤버이기 때문에 이렇게 부른다.

- **필드(Field)**: 데이터 항목을 가리키는 전통적인 용어이다. 데이터베이스, 엑셀 등에서 데이터 각각의 항목을 필드라 한다.
- 자바에서 멤버 변수, 필드는 같은 뜻이다. 클래스에 소속된 변수를 뜻한다.

클래스는 관례상 대문자로 시작하고 낙타 표기법을 사용한다.
예): `Student`, `User`, `MemberService`

이제 학생 클래스를 사용하는 코드를 작성해보자.

ClassStart3

```
package class1;

public class ClassStart3 {
    public static void main(String[] args) {
        Student student1;
        student1 = new Student();
        student1.name = "학생1";
        student1.age = 15;
        student1.grade = 90;

        Student student2 = new Student();
        student2.name = "학생2";
        student2.age = 16;
        student2.grade = 80;

        System.out.println("이름:" + student1.name + " 나이:" + student1.age + " 성적:" + student1.grade);
        System.out.println("이름:" + student2.name + " 나이:" + student2.age + " 성적:" + student2.grade);
    }
}
```

실행 결과

```
이름:학생1 나이:15 성적:90
이름:학생2 나이:16 성적:80
```

클래스와 사용자 정의 타입

- 타입은 데이터의 종류나 형태를 나타낸다.
- `int`라고 하면 정수 타입, `String`이라고 하면 문자 타입이다.
- 학생(`Student`)이라는 타입을 만들면 되지 않을까?

- 클래스를 사용하면 `int`, `String`과 같은 타입을 직접 만들 수 있다.
- 사용자가 직접 정의하는 사용자 정의 타입을 만들려면 설계도가 필요하다. 이 **설계도가 바로 클래스**이다.
- 설계도인 클래스를 사용해서 **실제 메모리에 만들어진 실체를 객체 또는 인스턴스**라 한다.
- 클래스를 통해서 사용자가 원하는 종류의 데이터 타입을 마음껏 정의할 수 있다.

용어: 클래스, 객체, 인스턴스

클래스는 설계도이고, 이 설계도를 기반으로 실제 메모리에 만들어진 실체를 객체 또는 인스턴스라 한다. 둘다 같은 의미로 사용된다.

여기서는 학생(`Student`) 클래스를 기반으로 학생1(`student1`), 학생2(`student2`) 객체 또는 인스턴스를 만들었다.

이제 코드를 하나씩 분석해보자.

1. 변수 선언

```
Student student1 //Student 변수 선언
```

Student **student1**

- `Student student1`
  - `Student` 타입을 받을 수 있는 변수를 선언한다.
  - `int`는 정수를, `String`은 문자를 담을 수 있듯이 `Student`는 `Student` 타입의 객체(인스턴스)를 받을 수 있다.

2. 객체 생성

```
student1 = new Student() //Student 인스턴스 생성
```

Student **student1**

x001

String name
int age
int grade

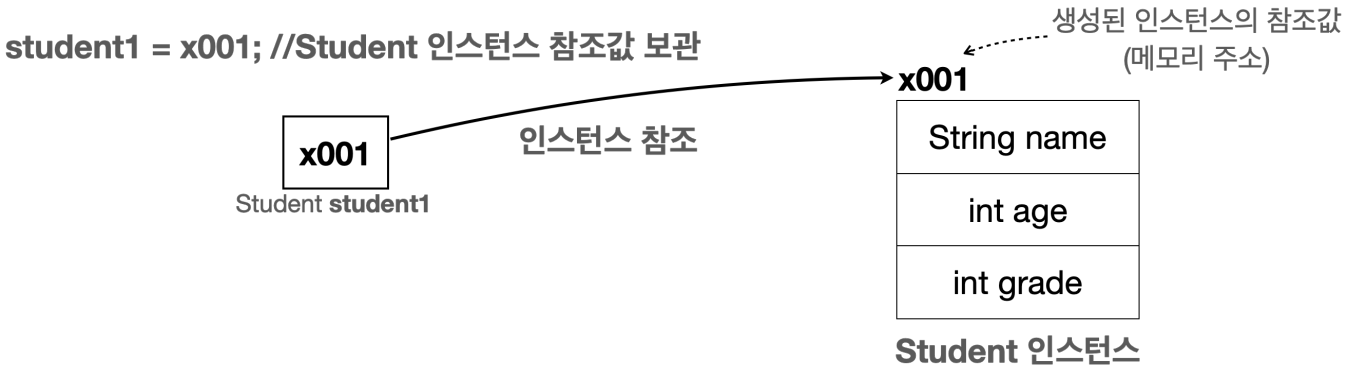
Student 인스턴스

`student1 = new Student()` 코드를 나누어 분석해보자.

- 객체를 사용하려면 먼저 설계도인 클래스를 기반으로 객체(인스턴스)를 생성해야 한다.

- `new Student()` : `new`는 새로 생성한다는 뜻이다. `new Student()`는 `Student` 클래스 정보를 기반으로 새로운 객체를 생성하라는 뜻이다. 이렇게 하면 메모리에 실제 `Student` 객체(인스턴스)를 생성한다.
- 객체를 생성할 때는 `new 클래스명()`을 사용하면 된다. 마지막에 `()`도 추가해야 한다.
- `Student` 클래스는 `String name`, `int age`, `int grade` 멤버 변수를 가지고 있다. 이 변수를 사용하는 데 필요한 메모리 공간도 함께 확보한다.

3. 참조값 보관



- 객체를 생성하면 자바는 메모리 어딘가에 있는 이 객체에 접근할 수 있는 참조값(주소)(`x001`)을 반환한다.
  - 여기서 `x001`이라고 표현한 것이 참조값이다. (실제로 `x001`처럼 표현되는 것은 아니고 이해를 돕기 위한 예시이다.)
- `new` 키워드를 통해 객체가 생성되고 나면 참조값을 반환한다. 앞서 선언한 변수인 `Student student1`에 생성된 객체의 참조값(`x001`)을 보관한다.
- `Student student1` 변수는 이제 메모리에 존재하는 실제 `Student` 객체(인스턴스)의 참조값을 가지고 있다.
  - `student1` 변수는 방금 만든 객체에 접근할 수 있는 참조값을 가지고 있다. 따라서 이 변수를 통해서 객체를 접근(참조)할 수 있다. 쉽게 이야기해서 `student1` 변수를 통해 메모리에 있는 실제 객체를 접근하고 사용할 수 있다.

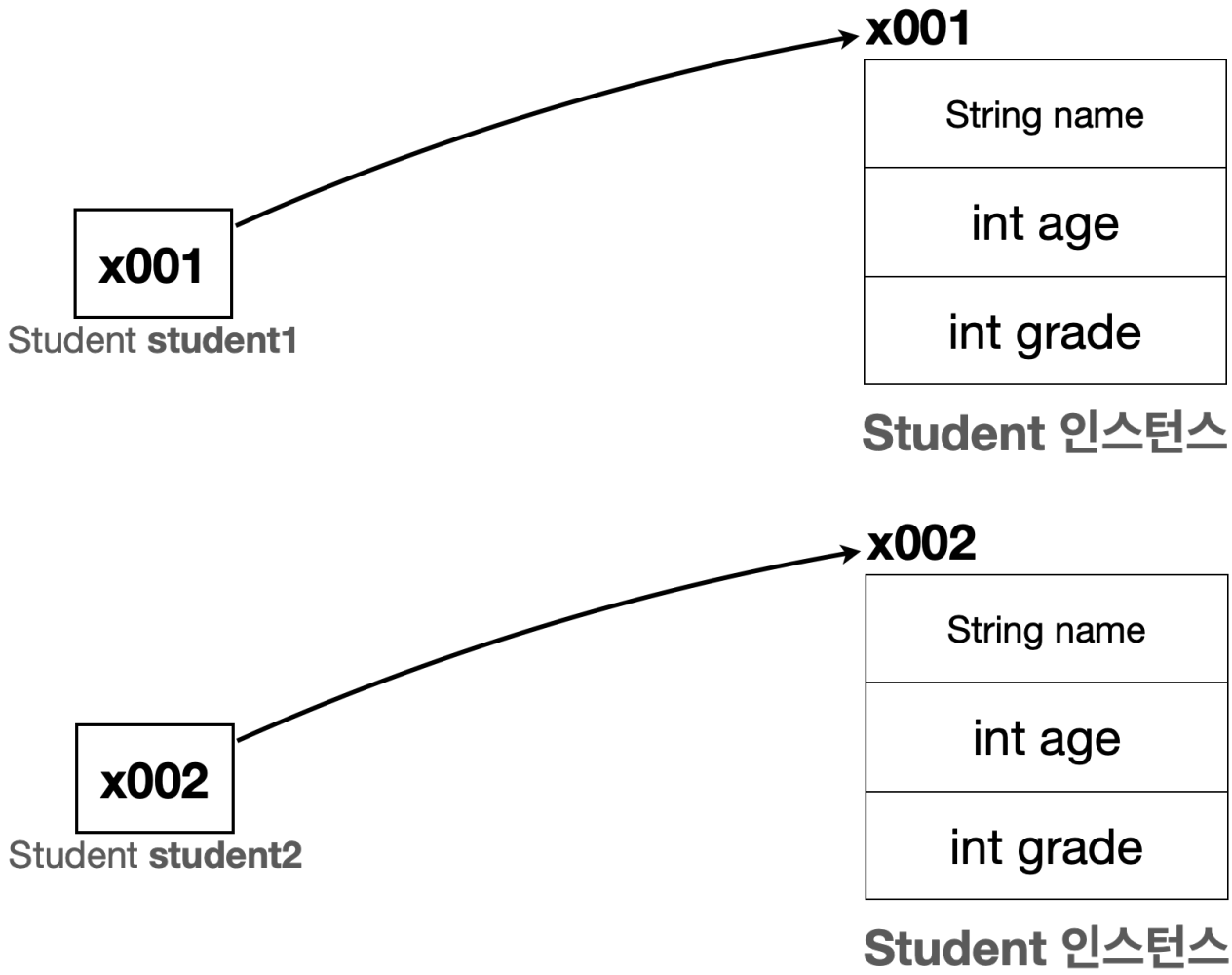
참조값을 변수에 보관해야 하는 이유

객체를 생성하는 `new Student()` 코드 자체에는 아무런 이름이 없다. 이 코드는 단순히 `Student` 클래스를 기반으로 메모리에 실제 객체를 만드는 것이다. 따라서 생성한 객체에 접근할 수 있는 방법이 필요하다. 이런 이유로 객체를 생성할 때 반환되는 참조값을 어딘가에 보관해두어야 한다. 앞서 `Student student1` 변수에 참조값(`x001`)을 저장해두었으므로 저장한 참조값(`x001`)을 통해서 실제 메모리에 존재하는 객체에 접근할 수 있다.

지금까지 설명한 내용을 간단히 풀어보면 다음과 같다.

```
Student student1 = new Student(); //1. Student 객체 생성
Student student1 = x001; //2. new Student()의 결과로 x001 참조값 반환
student1 = x001; //3. 최종 결과
```

이후에 학생2(`student2`)까지 생성하면 다음과 같이 `Student` 객체(인스턴스)가 메모리에 2개 생성된다. 각각 참조값이 다르므로 서로 구분할 수 있다.



참조값을 확인하고 싶다면 다음과 같이 객체를 담고 있는 변수를 출력해보면 된다.

```
System.out.println(student1);
System.out.println(student2);
```

출력 결과

```
class1.Student@7a81197d
class1.Student@2f2c9b19
```

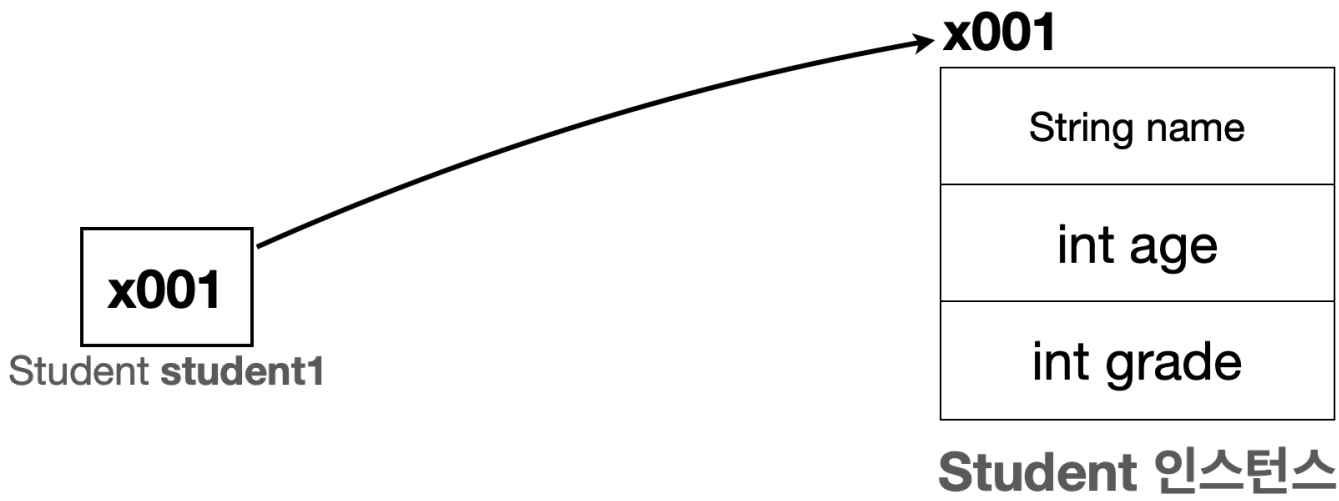
@앞은 패키지 + 클래스 정보를 뜻한다. @뒤에 16진수는 참조값을 뜻한다.

객체 사용

클래스를 통해 생성한 객체를 사용하려면 먼저 메모리에 존재하는 객체에 접근해야 한다. 객체에 접근하려면 `.`(점, dot)을 사용하면 된다.

```
//객체 값 대입
student1.name = "학생1";
student1.age = 15;
student1.grade = 90;

//객체 값 사용
System.out.println("이름:" + student1.name + " 나이:" + student1.age + " 성적:" +
student1.grade);
```

객체 참조 그림



객체에 값 대입

객체가 가지고 있는 멤버 변수(`name`, `age`, `grade`)에 값을 대입하려면 먼저 객체에 접근해야 한다.
객체에 접근하려면 `.`(점, dot) 키워드를 사용하면 된다. 이 키워드는 변수(`student1`)에 들어있는 참조값(`x001`)을 읽어서 메모리에 존재하는 객체에 접근한다.

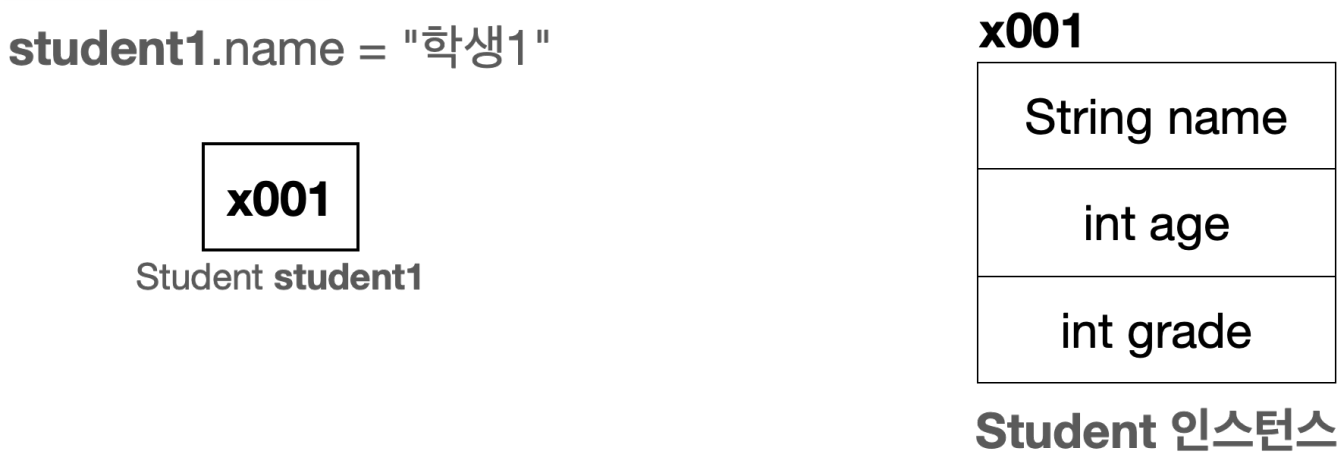
순서를 간단히 풀어보면 다음과 같다.

```
student1.name="학생1" //1. student1 객체의 name 멤버 변수에 값 대입
x001.name="학생1" //2.변수에 있는 참조값을 통해 실제 객체에 접근, 해당 객체의 name 멤버 변수에 값
대입
```

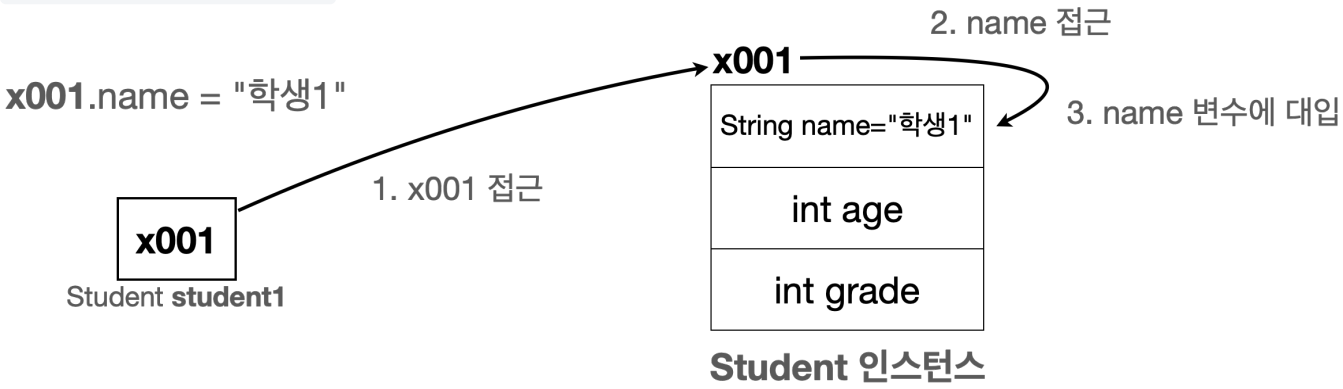
`student1.`(dot)이라고 하면 `student1` 변수가 가지고 있는 참조값을 통해 실제 객체에 접근한다.
`student1`은 `x001`이라는 참조값을 가지고 있으므로 `x001` 위치에 있는 `Student` 객체에 접근한다.

그림으로 자세히 알아보자.

`student1.name="학생1"` **코드 실행 전**



`student1.name="학생1"` **코드 실행 후**



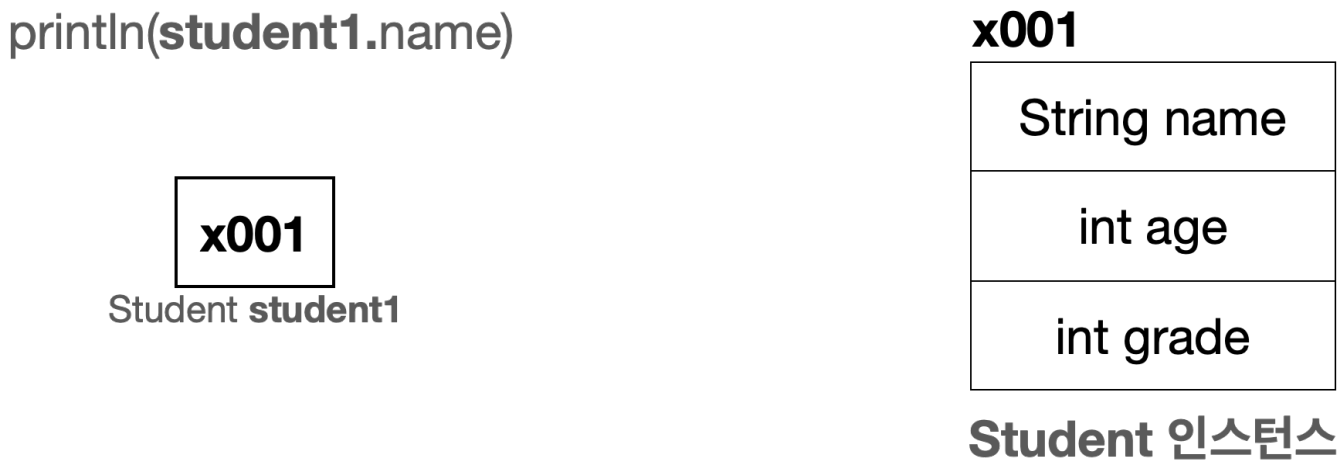
- `student1.name` 코드를 통해 `.`(dot) 키워드가 사용되었다. `student1` 변수가 가지고 있는 참조값을 통해 실제 객체에 접근한다.
- `x001.name = "학생1"` : `x001` 객체가 있는 곳의 `name` 멤버 변수에 "학생1" 데이터가 저장된다.

객체 값 읽기

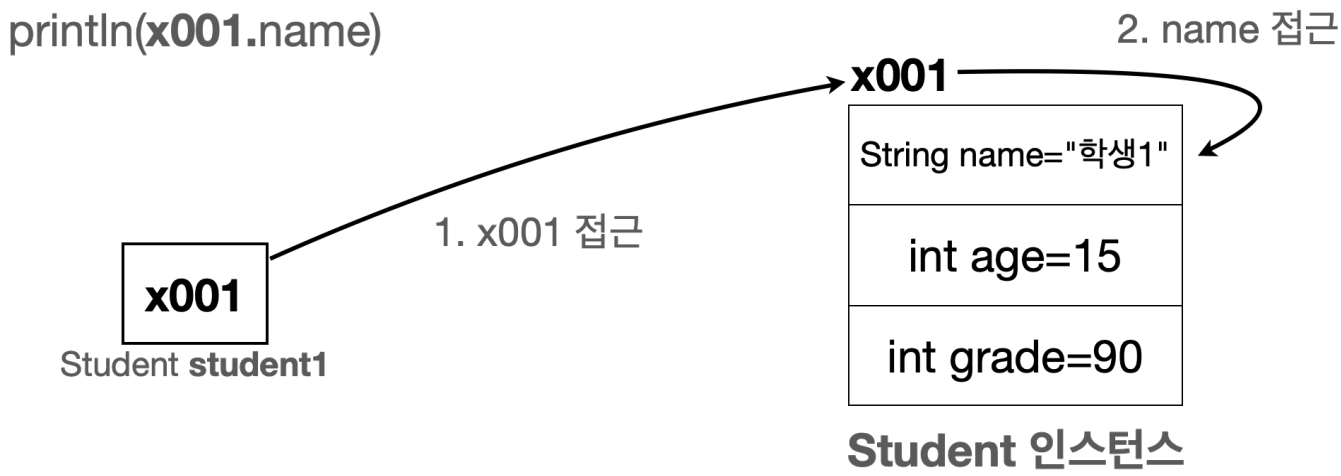
객체의 값을 읽는 것도 앞서 설명한 내용과 같다. `.`(점, dot) 키워드를 통해 참조값을 사용해서 객체에 접근한 다음에 원하는 작업을 하면 된다. 다음 예제를 보자.

```
//1. 객체 값 읽기
System.out.println("이름:" + student1.name);
//2. 변수에 있는 참조값을 통해 실제 객체에 접근하고, name 멤버 변수에 접근한다.
System.out.println("이름:" + x001.name);
//3. 객체의 멤버 변수의 값을 읽어옴
System.out.println("이름:" + "학생1");
```

객체 값 읽기 - 그림1



객체 값 읽기 - 그림2



- `x001`에 있는 `Student` 인스턴스의 `name` 멤버 변수는 "학생1"이라는 값을 가지고 있다. 이 값을 읽어서 사용한다.

클래스, 객체, 인스턴스 정리

클래스 - Class

클래스는 객체를 생성하기 위한 '틀' 또는 '설계도'이다. 클래스는 객체가 가져야 할 속성(변수)과 기능(메서드)를 정의한다. 예를 들어 학생이라는 클래스는 속성으로 `name`, `age`, `grade`를 가진다. 참고로 기능(메서드)은 뒤에서 설명한다. 지금은 속성(변수)에 집중하자

- 틀: 붕어빵 틀을 생각해보자. 붕어빵 틀은 붕어빵이 아니다! 이렇게 생긴 붕어빵이 나왔으면 좋겠다고 만드는 틀일 뿐이다. 실제 먹을 수 있는 것이 아니다. 실제 먹을 수 있는 팥 붕어빵을 객체 또는 인스턴스라 한다.
- 설계도: 자동차 설계도를 생각해보자. 자동차 설계도는 자동차가 아니다! 설계도는 실제 존재하는 것이 아니라 개

념으로만 있는 것이다. 설계도를 통해 생산한 실제 존재하는 흰색 테슬라 모델 Y 자동차를 객체 또는 인스턴스라 한다.

객체 - Object

객체는 클래스에서 정의한 속성과 기능을 가진 실체이다. 객체는 서로 독립적인 상태를 가진다.
예를 들어 위 코드에서 `student1`은 학생1의 속성을 가지는 객체이고, `student2`는 학생2의 속성을 가지는 객체이다. `student1`과 `student2`는 같은 클래스에서 만들어졌지만, 서로 다른 객체이다.

인스턴스 - Instance

인스턴스는 특정 클래스로부터 생성된 객체를 의미한다. 그래서 객체와 인스턴스라는 용어는 자주 혼용된다. 인스턴스는 주로 객체가 어떤 클래스에 속해 있는지 강조할 때 사용한다. 예를 들어서 `student1` 객체는 `Student` 클래스의 인스턴스다. 라고 표현한다.

객체 vs 인스턴스

둘다 클래스에서 나온 실체라는 의미에서 비슷하게 사용되지만, 용어상 인스턴스는 객체보다 좀 더 관계에 초점을 맞춘 단어이다. 보통 `student1`은 `Student`의 객체이다. 라고 말하는 대신 `student1`은 `Student`의 인스턴스이다. 라고 특정 클래스와의 관계를 명확히 할 때 인스턴스라는 용어를 주로 사용한다.
좀 더 쉽게 풀어보자면, 모든 인스턴스는 객체이지만, 우리가 인스턴스라고 부르는 순간은 특정 클래스로부터 그 객체가 생성되었음을 강조하고 싶을 때이다. 예를 들어 `student1`은 객체이지만, 이 객체가 `Student` 클래스로부터 생성되다는 점을 명확히 하기 위해 `student1`을 `Student`의 인스턴스라고 부른다.

하지만 둘다 클래스에서 나온 실체라는 핵심 의미는 같기 때문에 보통 둘을 구분하지 않고 사용한다.

배열 도입 - 시작

클래스와 객체 덕분에 학생 데이터를 구조적으로 이해하기 쉽게 변경할 수 있었다.
마치 실제 학생이 있고, 그 안에 각 학생의 정보가 있는 것 같다. 따라서 사람이 이해하기도 편리하다.
이제 각각의 학생 별로 객체를 생성하고, 해당 객체에 학생의 데이터를 관리하면 된다.

하지만 코드를 보면 아쉬운 부분이 있는데, 바로 학생을 출력하는 부분이다.

```
System.out.println("이름:" + student1.name + " 나이:" + student1.age + ...);
System.out.println("이름:" + student2.name + " 나이:" + student2.age + ...);
```

새로운 학생이 추가될 때 마다 출력하는 부분도 함께 추가해야 한다.

배열을 사용하면 특정 타입을 연속한 데이터 구조로 묶어서 편리하게 관리할 수 있다.
`Student` 클래스를 사용한 변수들도 `Student` 타입이기 때문에 학생도 배열을 사용해서 하나의 데이터 구조로 묶어서 관리할 수 있다.

`Student` 타입을 사용하는 배열을 도입해보자.

ClassStart4

```
package class1;

public class ClassStart4 {
    public static void main(String[] args) {
        Student student1 = new Student();
        student1.name = "학생1";
        student1.age = 15;
        student1.grade = 90;

        Student student2 = new Student();
        student2.name = "학생2";
        student2.age = 16;
        student2.grade = 80;

        Student[] students = new Student[2];
        students[0] = student1;
        students[1] = student2;

        System.out.println("이름:" + students[0].name + " 나이:" + students[0].age
+ " 성적:" + students[0].grade);
        System.out.println("이름:" + students[1].name + " 나이:" + students[1].age
+ " 성적:" + students[1].grade);

    }
}
```

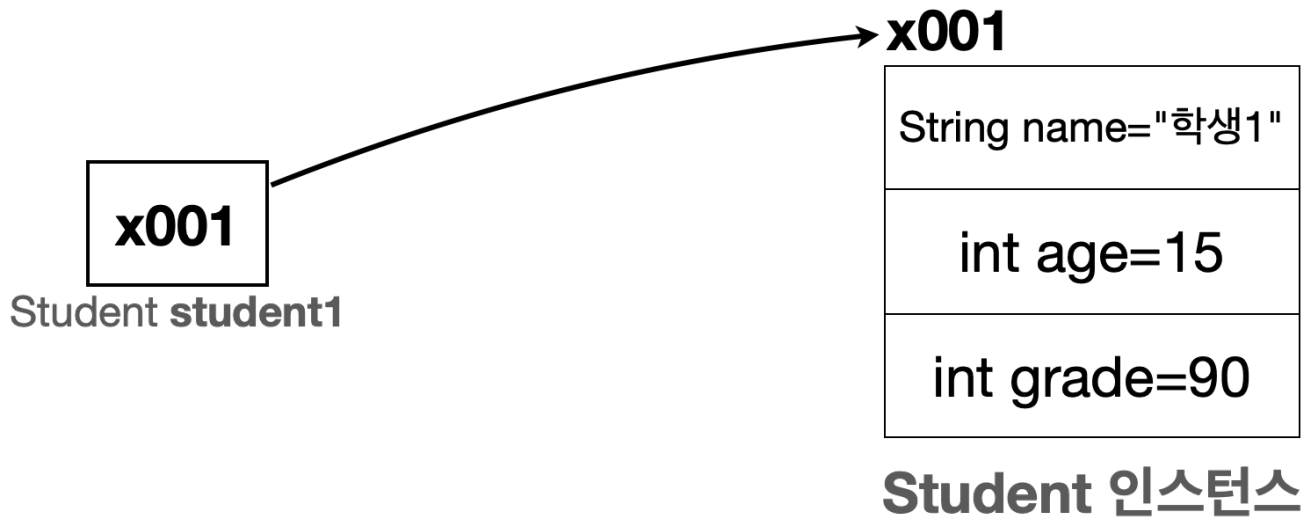
코드를 분석해보자.

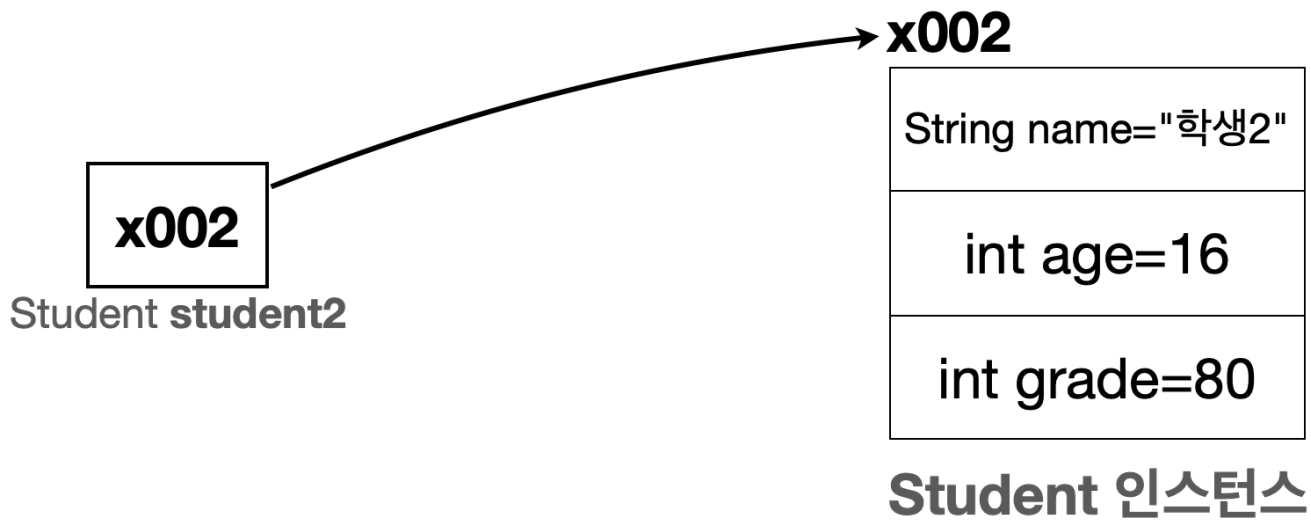
```
Student student1 = new Student();
student1.name = "학생1";
student1.age = 15;
student1.grade = 90;

Student student2 = new Student();
student2.name = "학생2";
```

```
student2.age = 16;
student2.grade = 80;
```

`Student` 클래스를 기반으로 `student1`, `student2` 인스턴스를 생성한다. 그리고 필요한 값을 채워둔다.

인스턴스 생성 그림

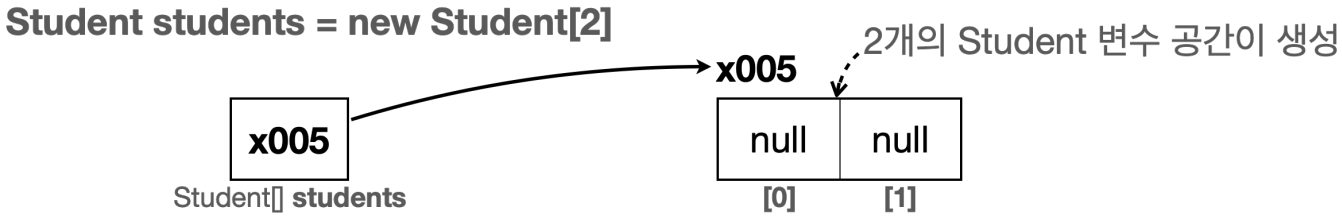




배열에 참조값 대입

이번에는 `Student`를 담을 수 있는 배열을 생성하고, 해당 배열에 `student1`, `student2` 인스턴스를 보관하자.

```
Student[] students = new Student[2];
```



- `Student` 변수를 2개 보관할 수 있는 사이즈 2의 배열을 만든다.
- `Student` 타입의 변수는 `Student` 인스턴스의 참조값을 보관한다. `Student` 배열의 각각의 항목도 `Student` 타입의 변수일 뿐이다. 따라서 `Student` 타입의 참조값을 보관한다.
  - `student1`, `student2` 변수를 생각해보면 `Student` 타입의 참조값을 보관한다.
- 배열에는 아직 참조값을 대입하지 않았기 때문에 참조값이 없다는 의미의 `null` 값으로 초기화 된다.

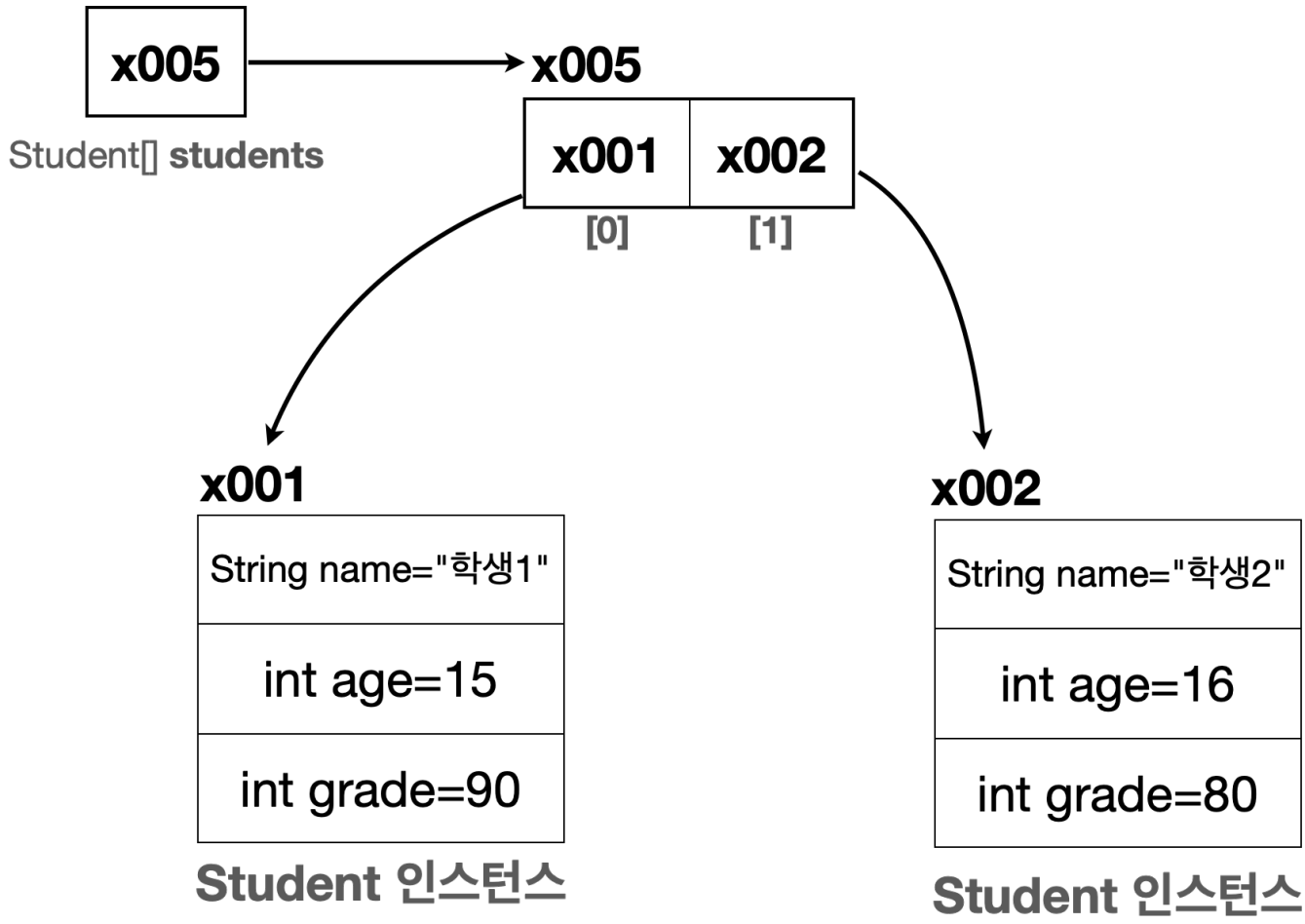
이제 배열에 객체를 보관하자.

```
students[0] = student1;
students[1] = student2;

//자바에서 대입은 항상 변수에 들어 있는 값을 복사한다.
students[0] = x001;
students[1] = x002;
```

자바에서 대입은 항상 변수에 들어 있는 값을 복사한다.

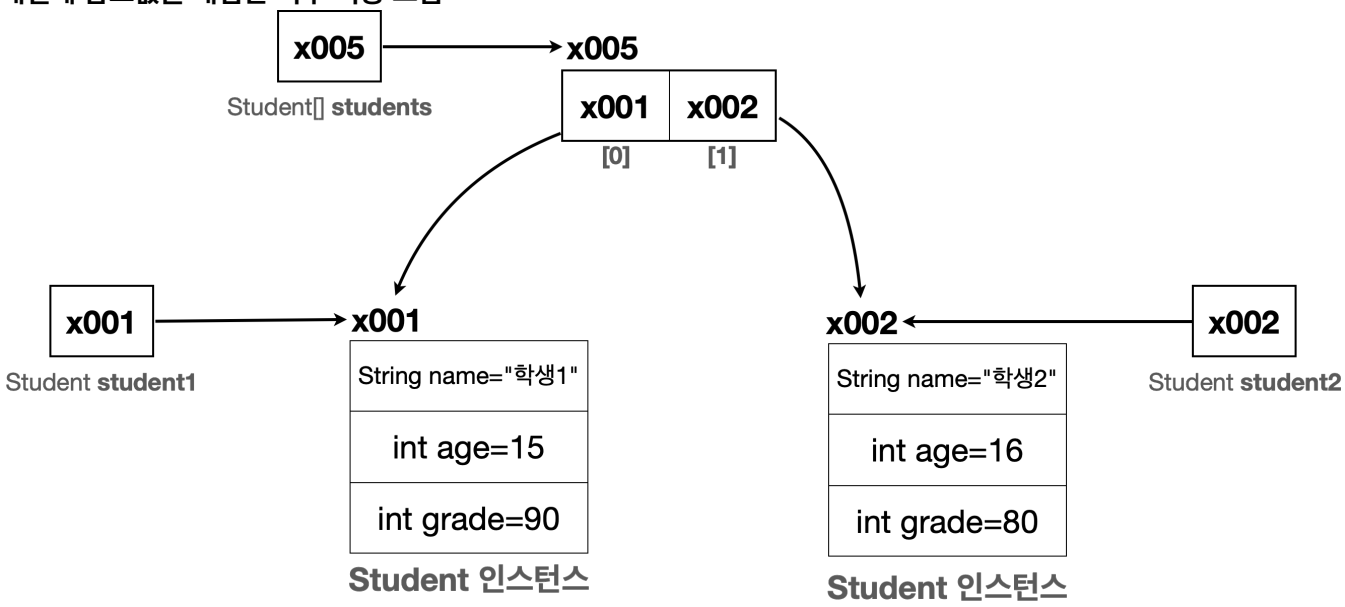
`student1`, `student2`에는 참조값이 보관되어 있다. 따라서 이 참조값이 배열에 저장된다. 또는 `student1`, `student2`에 보관된 참조값을 읽고 복사해서 배열에 대입한다고 표현한다.

배열에 참조값을 대입한 이후 배열 그림



이제 배열은 `x001`, `x002`의 참조값을 가진다. 참조값을 가지고 있기 때문에 `x001`(학생1), `x002`(학생2) `Student` 인스턴스에 모두 접근할 수 있다.

배열에 참조값을 대입한 이후 최종 그림



자바에서 대입은 항상 변수에 들어 있는 값을 복사해서 전달한다.

```
students[0] = student1;
students[1] = student2;
```

```
//자바에서 대입은 항상 변수에 들어 있는 값을 복사한다.
students[0] = x001;
students[1] = x002;
```

자바에서 변수의 대입(=)은 모두 변수에 들어있는 값을 복사해서 전달하는 것이다. 이 경우 오른쪽 변수인 `student1`, `student2`에는 참조값이 들어있다. 그래서 이 값을 복사해서 왼쪽에 있는 배열에 전달한다. 따라서 기존 `student1`, `student2`에 들어있던 참조값은 당연히 그대로 유지된다.

주의!

변수에는 인스턴스 자체가 들어있는 것이 아니다! 인스턴스의 위치를 가리키는 참조값이 들어있을 뿐이다! 따라서 대입(=)시에 인스턴스가 복사되는 것이 아니라 참조값만 복사된다.

배열에 들어있는 객체 사용

배열에 들어있는 객체를 사용하려면 먼저 배열에 접근하고, 그 다음에 객체에 접근하면 된다. 이전에 설명한 그림과 코드를 함께 보면 쉽게 이해가 될 것이다.

학생1 예제

```
System.out.println(students[0].name); //배열 접근 시작
System.out.println(x005[0].name); //[0]를 사용해서 x005 배열의 0번 요소에 접근
System.out.println(x001.name); //.(dot)을 사용해서 참조값으로 객체에 접근
System.out.println("학생1");
```

학생2 예제

```
System.out.println(students[1].name); //배열 접근 시작
System.out.println(x005[1].name); //[1]를 사용해서 x005 배열의 1번 요소에 접근
System.out.println(x002.name); //.(dot)을 사용해서 참조값으로 객체에 접근
System.out.println("학생2");
```

배열 도입 - 리펙토링

배열을 사용한 덕분에 출력에서 다음과 같이 for문을 도입할 수 있게 되었다.

ClassStart5

```
package class1;
```

```
public class ClassStart5 {
    public static void main(String[] args) {
        Student student1 = new Student();
        student1.name = "학생1";
        student1.age = 15;
        student1.grade = 90;

        Student student2 = new Student();
        student2.name = "학생2";
        student2.age = 16;
        student2.grade = 80;

        //배열 선언
        Student[] students = new Student[]{student1, student2};

        //for문 적용
        for (int i = 0; i < students.length; i++) {
            System.out.println("이름:" + students[i].name + " 나이:" +
students[i].age + " 성적:" + students[i].grade);
        }

    }
}
```

배열 선언 최적화

우리가 직접 정의한 `Student` 타입도 일반적인 변수와 동일하게 배열을 생성할 때 포함할 수 있다.

```
Student[] students = new Student[]{student1, student2};
```

생성과 선언을 동시에 하는 경우 다음과 같이 더 최적화 할 수 있다.

```
Student[] students = {student1, student2};
```

for문 최적화

배열을 사용한 덕분에 for문을 사용해서 반복 작업을 깔끔하게 처리할 수 있다.

for문 도입

```
for (int i = 0; i < students.length; i++) {
    System.out.println("이름:" + students[i].name + " 나이:" + students[i].age
+ ...);
```

```
}
```

for문 - 반복 요소를 변수에 담아서 처리하기

```
for (int i = 0; i < students.length; i++) {
    Student s = students[i];
    System.out.println("이름:" + s.name + " 나이:" + s.age + ...);
}
```

`students[i].name`, `students[i].age`처럼 `students[i]`를 자주 접근하는 것이 번거롭다면 반복해서 사용하는 객체를 `Students s`와 같은 변수에 담아두고 사용해도 된다.
물론 이런 경우에는 다음과 같이 향상된 for문을 사용하는 것이 가장 깔끔하다.

향상된 for문(Enhanced For Loop)

```
for (Student s : students) {
    System.out.println("이름:" + s.name + " 나이:" + s.age + " 성적:" + s.grade);
}
```

문제와 풀이

문제를 스스로 풀어보세요.

> **백문이 불여일타!**
> 프로그래밍은 수영이나 자전거 타기와 비슷하다. 아무리 책을 보고 강의 영상을 봐도 직접 물속에 들어가거나 자전거를 타봐야 이해할 수 있다. 프로그래밍도 마찬가지이다. 이해하고 외우는 것도 중요하지만 무엇보다 직접 코딩을 해보는 것이 더욱 중요하다. 읽어서 이해가 잘 안되더라도 직접 코딩을 해보면 자연스럽게 이해가 되는 경우도 많다. 백번 읽는 것 보다 한번 직접 코딩해서 결과를 보는 것이 좋은 개발자로 빠르게 성장할 수 있는 지름길이다.

문제: 영화 리뷰 관리하기1

문제 설명

당신은 영화 리뷰 정보를 관리하려고 한다. 먼저, 영화 리뷰 정보를 담을 수 있는 `MovieReview` 클래스를 만들어보자.

요구 사항

1. `MovieReview` 클래스는 다음과 같은 멤버 변수를 포함해야 한다.
    - 영화 제목 (`title`)
    - 리뷰 내용 (`review`)
2. `MovieReviewMain` 클래스 안에 `main()` 메서드를 포함하여, 영화 리뷰 정보를 선언하고 출력하자.

예시 코드 구조

```
public class MovieReview {
    String title;
    String review;
}
```

```
public class MovieReviewMain {
    public static void main(String[] args) {
        // 영화 리뷰 정보 선언
        // 영화 리뷰 정보 출력
    }
}
```

출력 예시

```
영화 제목: 인셉션, 리뷰: 인생은 무한 루프
영화 제목: 어바웃 타임, 리뷰: 인생 시간 영화!
```

답안 - 영화 리뷰 관리하기1

```
package class1.ex;

public class MovieReview {
    String title;
    String review;
}
```

```
package class1.ex;

public class MovieReviewMain1 {
    public static void main(String[] args) {
        MovieReview inception = new MovieReview();
        inception.title = "인셉션";
        inception.review = "인생은 무한 루프";

        MovieReview aboutTime = new MovieReview();
        aboutTime.title = "어바웃 타임";
```

```
        aboutTime.review = "인생 시간 영화!";

        System.out.println("영화 제목: " + inception.title + ", 리뷰: " +
inception.review);
        System.out.println("영화 제목: " + aboutTime.title + ", 리뷰: " +
aboutTime.review);
    }
}
```

문제: 영화 리뷰 관리하기2

기존 문제에 배열을 도입하고, 영화 리뷰를 배열에 관리하자.

리뷰를 출력할 때 배열과 `for` 문을 사용해서 `System.out.println` 을 한번만 사용하자.

답안 - 영화 리뷰 관리하기2

```
package class1.ex;

public class MovieReviewMain2 {
    public static void main(String[] args) {
        MovieReview[] reviews = new MovieReview[2];

        MovieReview inception = new MovieReview();
        inception.title = "인셉션";
        inception.review = "인생은 무한 루프";
        reviews[0] = inception;

        MovieReview aboutTime = new MovieReview();
        aboutTime.title = "어바웃 타임";
        aboutTime.review = "인생 시간 영화!";
        reviews[1] = aboutTime;

        for (MovieReview review : reviews) {
            System.out.println("영화 제목: " + review.title + ", 리뷰: " +
review.review);
        }
    }
}
```

문제: 상품 주문 시스템 개발

문제 설명

당신은 온라인 상점의 주문 관리 시스템을 만들려고 한다.

먼저, 상품 주문 정보를 담을 수 있는 `ProductOrder` 클래스를 만들어보자.

요구 사항

1. `ProductOrder` 클래스는 다음과 같은 멤버 변수를 포함해야 한다.
   - 상품명 (`productName`)
   - 가격 (`price`)
   - 주문 수량 (`quantity`)
2. `ProductOrderMain` 클래스 안에 `main()` 메서드를 포함하여, 여러 상품의 주문 정보를 배열로 관리하고, 그 정보들을 출력하고, 최종 결제 금액을 계산하여 출력하자.
3. 출력 예시와 같도록 출력하면 된다.

예시 코드 구조

```
public class ProductOrder {
    String productName;
    int price;
    int quantity;
}
```

```
public class ProductOrderMain {
    public static void main(String[] args) {
        // 여러 상품의 주문 정보를 담는 배열 생성
        // 상품 주문 정보를 `ProductOrder` 타입의 변수로 받아 저장
        // 상품 주문 정보와 최종 금액 출력
    }
}
```

출력 예시

```
상품명: 두부, 가격: 2000, 수량: 2
상품명: 김치, 가격: 5000, 수량: 1
상품명: 콜라, 가격: 1500, 수량: 2
총 결제 금액: 12000
```

답안 - 상품 주문 시스템 개발

```
package class1.ex;
```

```java
public class ProductOrder {
    String productName;
    int price;
    int quantity;
}
```

```java
package class1.ex;

public class ProductOrderMain {

    public static void main(String[] args) {
        ProductOrder[] orders = new ProductOrder[3];

        // 첫 번째 상품 주문 정보 입력
        ProductOrder order1 = new ProductOrder();
        order1.productName = "두부";
        order1.price = 2000;
        order1.quantity = 2;
        orders[0] = order1;

        // 두 번째 상품 주문 정보 입력
        ProductOrder order2 = new ProductOrder();
        order2.productName = "김치";
        order2.price = 5000;
        order2.quantity = 1;
        orders[1] = order2;

        // 세 번째 상품 주문 정보 입력
        ProductOrder order3 = new ProductOrder();
        order3.productName = "콜라";
        order3.price = 1500;
        order3.quantity = 2;
        orders[2] = order3;

        int totalAmount = 0;
        for (ProductOrder order : orders) {
            System.out.println("상품명: " + order.productName + ", 가격: " +
order.price + ", 수량: " + order.quantity);
            totalAmount += order.price * order.quantity;
        }

        System.out.println("총 결제 금액: " + totalAmount);
    }
```

```
}
```

정리