1. Hello World

#0.강의/1.자바로드맵/1.자바-입문

- /개발 환경 설정
- /다운로드 소스 코드 실행 방법
- /자바 프로그램 실행
- /주석(comment)
- /자바란?

개발 환경 설정

IDE - 인텔리제이 vs 이클립스

- 자바 프로그램을 개발할 때는 인텔리제이(IntelliJ) 또는 이클립스(Eclipse)라는 툴을 많이 사용한다. 과거에는 이클립스를 많이 사용했지만 최근에는 빠른 속도와 사용의 편의성 때문에 인텔리제이를 주로 사용한다.
- 자바로 개발하는 대부분의 메이저 회사들도 최근에는 인텔리제이를 주로 사용하므로 우리도 인텔리제이로 학습

OS - 윈도우 vs Mac

- 자바로 개발하는 대부분의 메이저 회사들은 Mac 사용
- 윈도우를 사용해도 무방함
- 강의는 Mac을 사용하지만 윈도우 사용자들을 최대한 배려해서 진행
    - 윈도우 화면 스크린샷, 윈도우용 단축키

참고: 자바를 별도로 설치하지 않아도 됩니다. 인텔리제이 안에서 자바 설치도 함께 진행합니다.

인텔리제이(IntelliJ) 설치하기

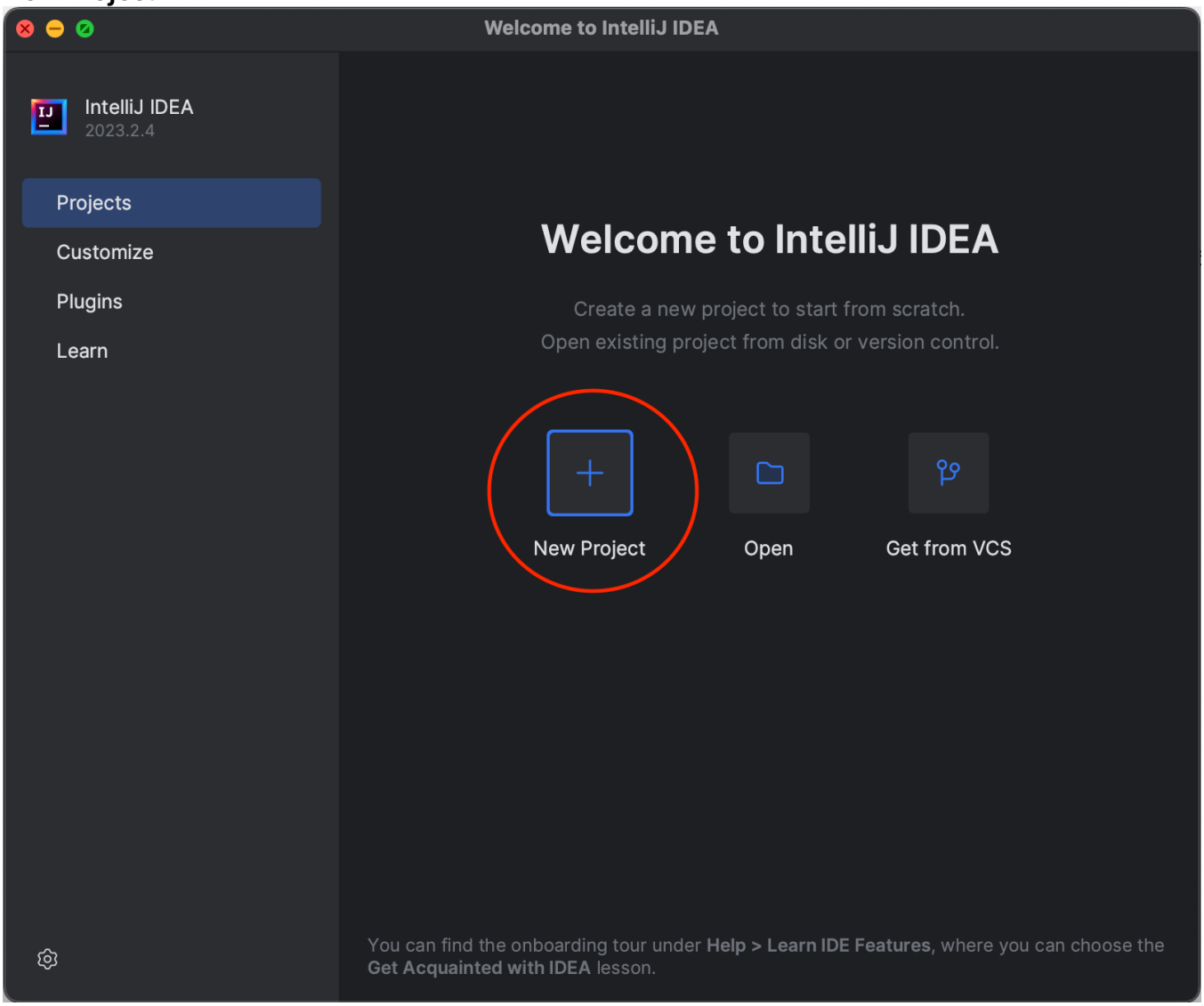
- 다운로드 링크: https://www.jetbrains.com/ko-kr/idea/download
- **IntelliJ IDEA Community Edition (무료 버전)**
    - OS 선택: Windows, macOs, Linux
        - Windows: `.exe` 선택
        - macOs: M1, M2: Apple Silicon 선택, 나머지: Intel 선택

> ~~**참고**: 인텔리제이는 무료 버전인 Community Edition과 유료 버전인 IntelliJ IDEA Ultimate가 있습니다. 제가 진행하는 모든 강의는 무료 버전인 Community Edition으로 충분합니다. 특히 자바 언어를 학습하는 단계에서는 유료 버전과 무료 버전의 차이가 없습니다.~~

> **인텔리제이 무료 버전과 유료 버전이 통합되었습니다.**
> 과거에는 무료 버전과 유로 버전이 분리되어 있었지만, 최근에 하나로 통합되었습니다.
> 통합된 인텔레제이도 자바와 관련된 핵심 기능은 여전히 무료로 제공됩니다. 따라서 다운로드 링크에서 보이는 제품을 다운로드 받으시면 됩니다.

인텔리제이 실행하기

New Project



- New Project를 선택해서 새로운 프로젝트를 만들자

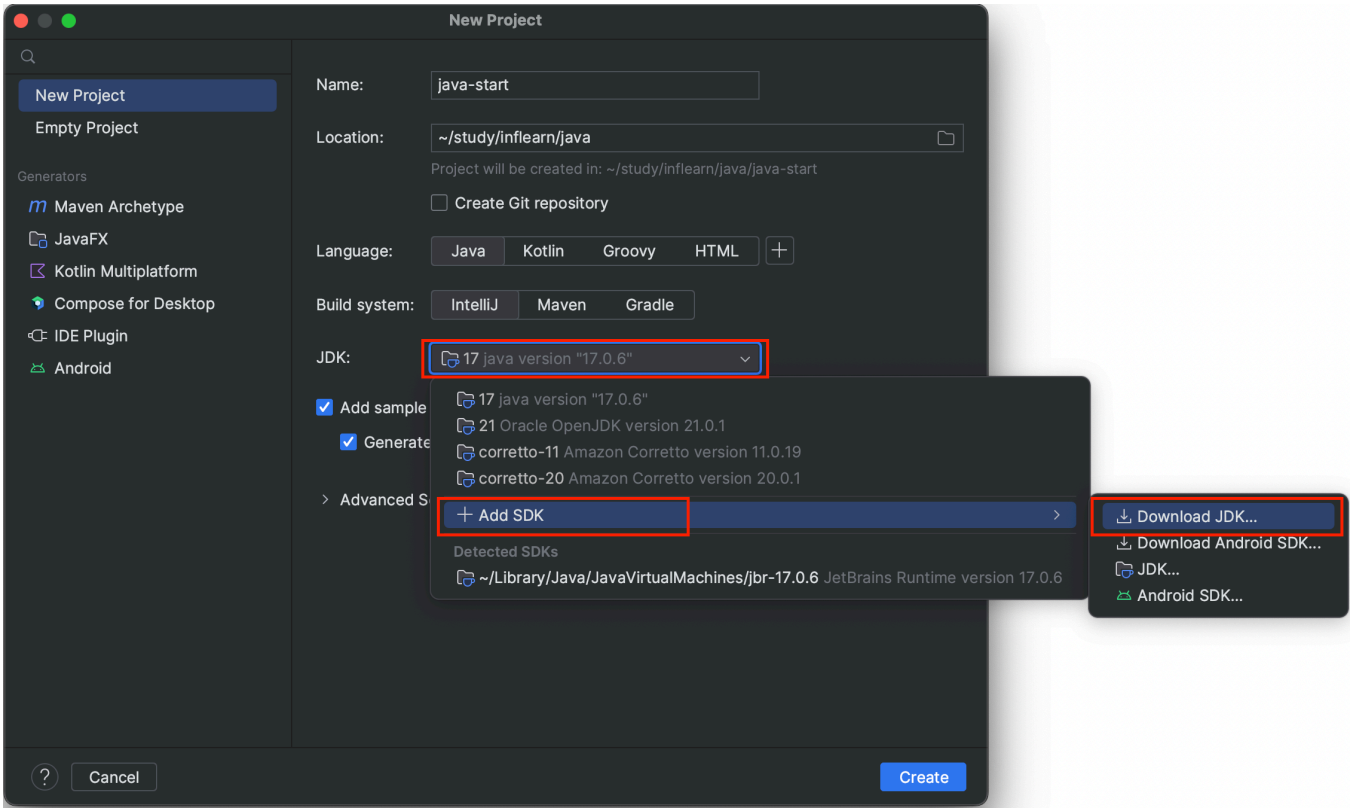
New Project 화면

- Name: java-start
- Location: 프로젝트 위치, 임의 선택
- Create Git repository 선택하지 않음
- Language: Java
- Build system: IntelliJ

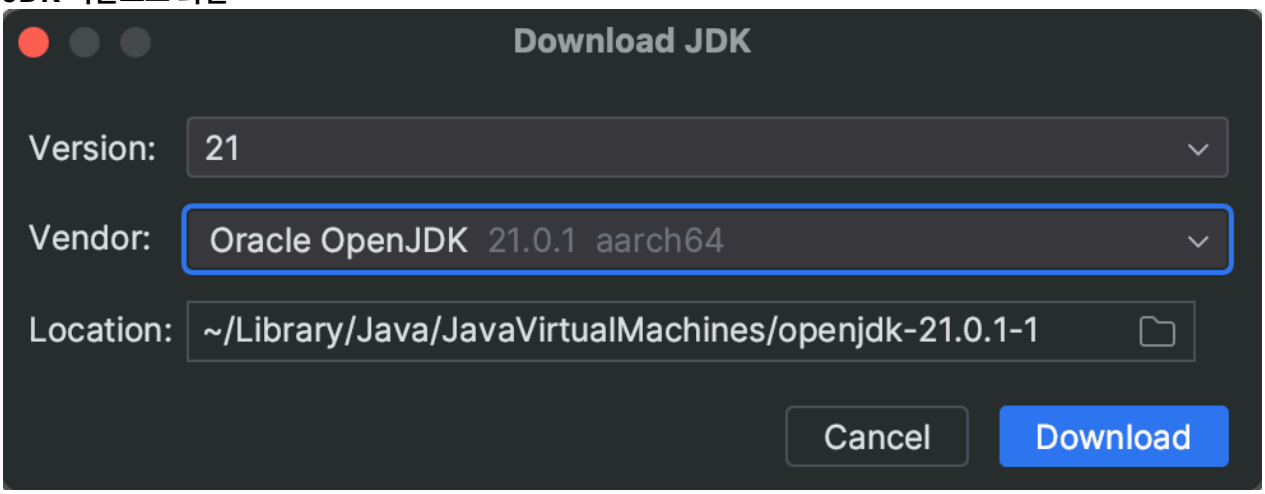
- JDK: 자바 버전 17 이상
- `Add sample code` 선택

JDK 다운로드 화면 이동 방법

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



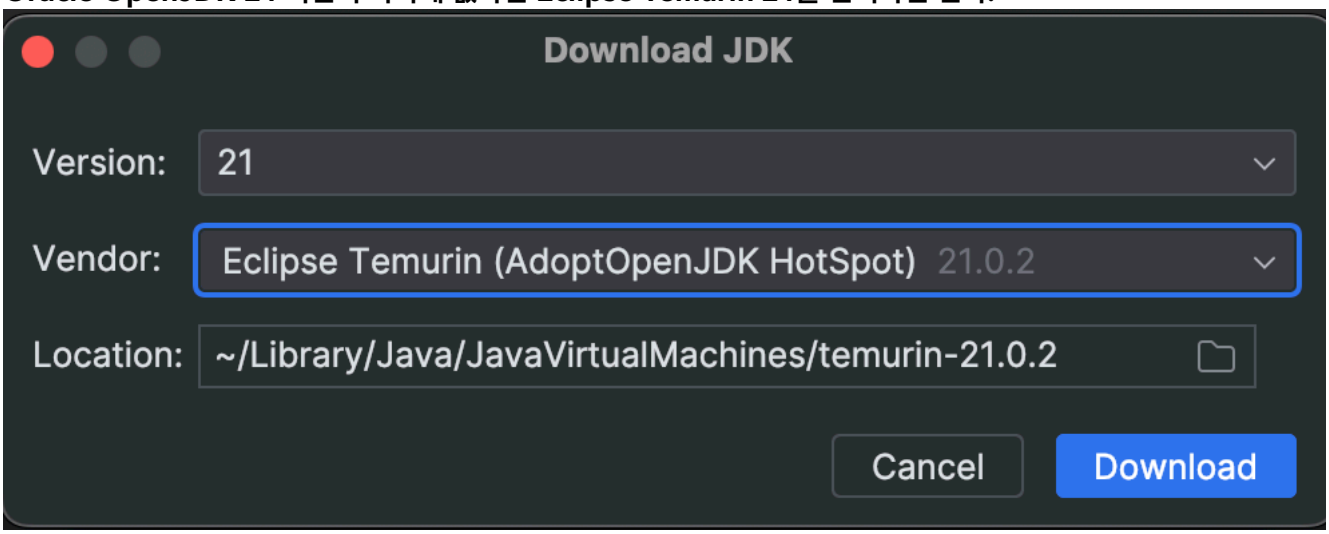
JDK 다운로드 화면



- 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

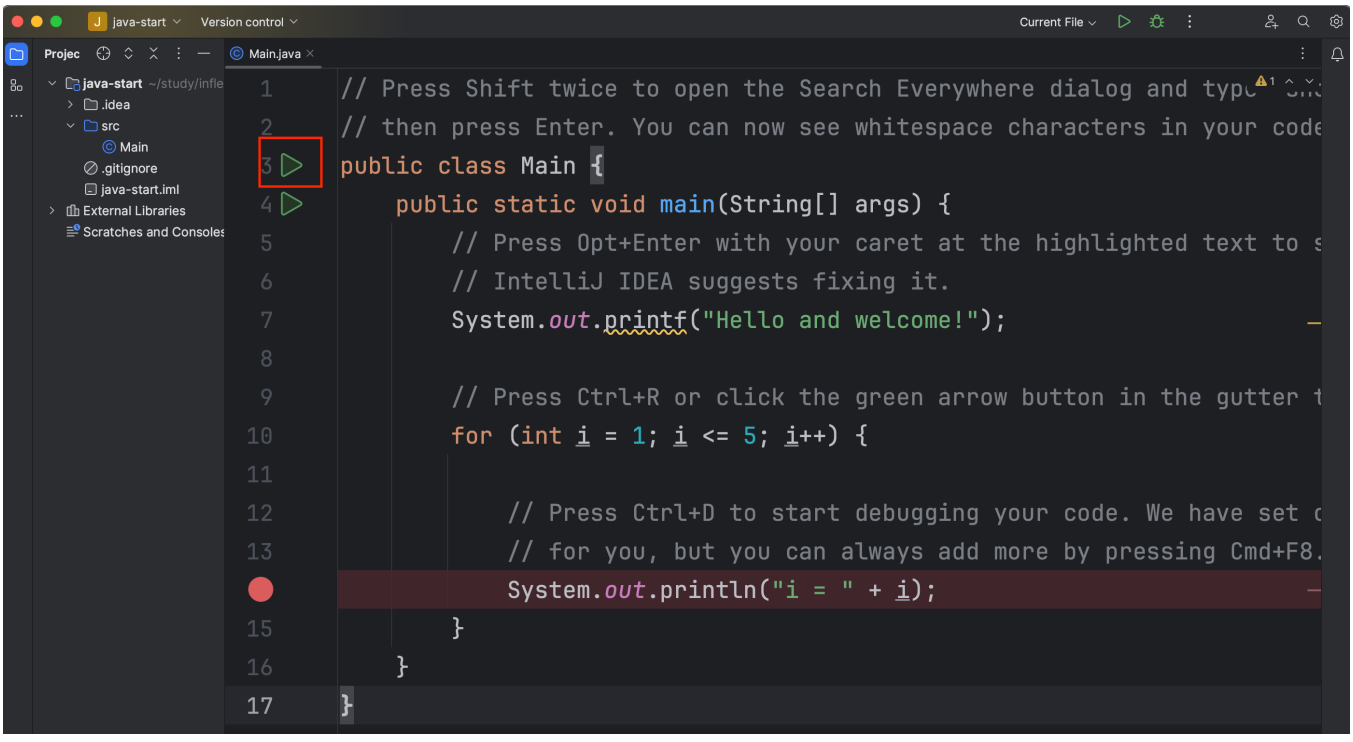
Oracle OpenJDK 21 버전이 목록에 없다면 Eclipse Temurin 21을 선택하면 된다.



`Download` 버튼을 통해서 다운로드 JDK를 다운로드 받는다.

다운로드가 완료 되고 이전 화면으로 돌아가면 `Create` 버튼 선택하자. 그러면 다음 IntelliJ 메인 화면으로 넘어간다.

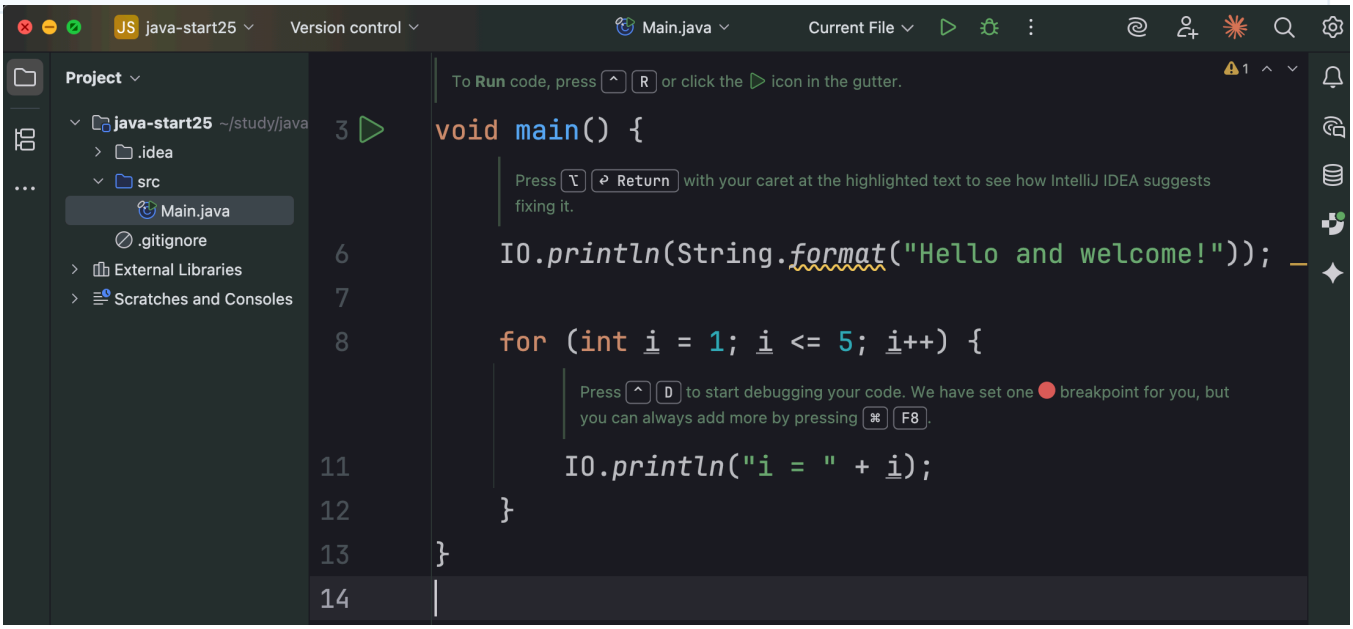
IntelliJ 메인 화면



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

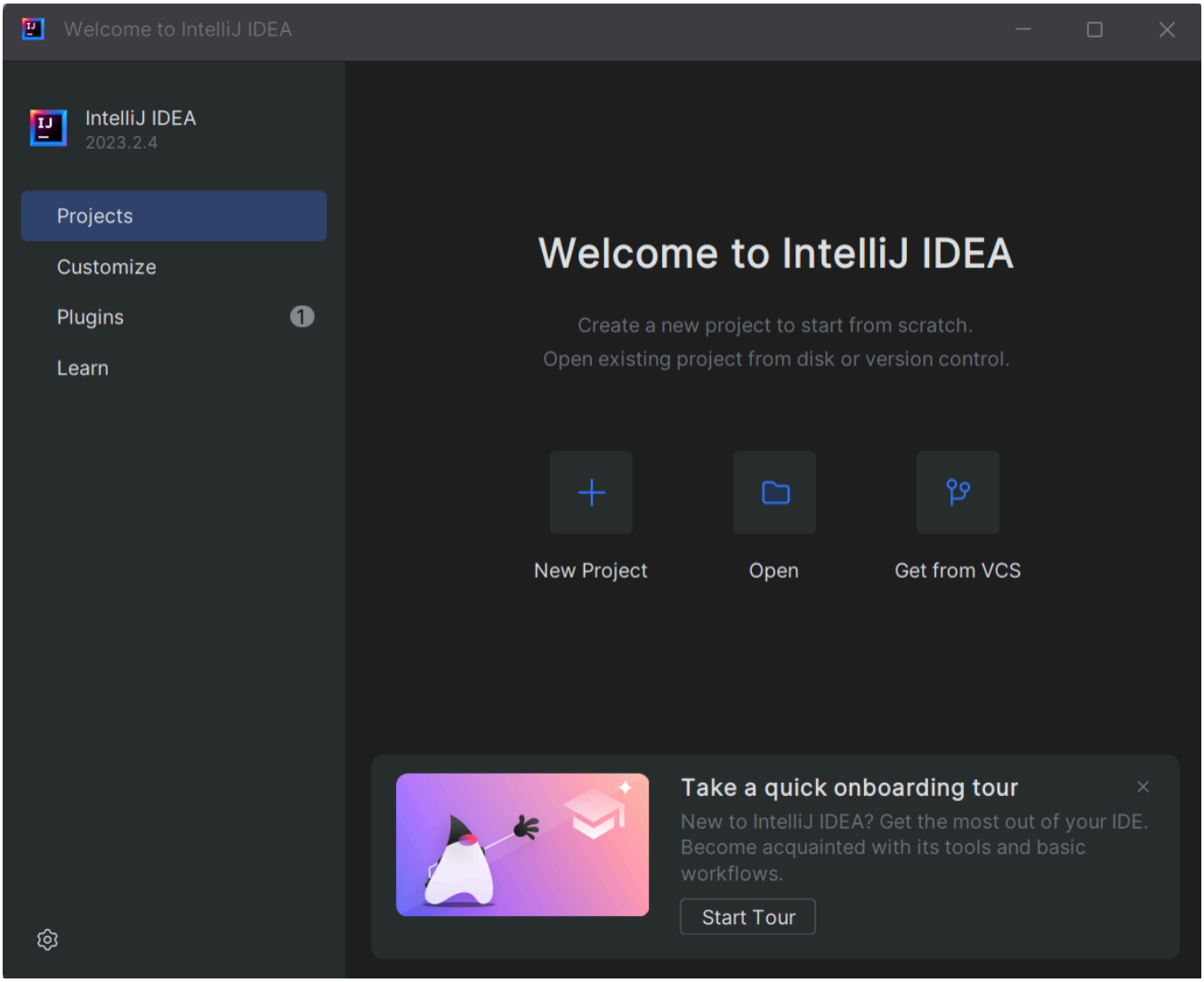
> 자바 25 이상 사용시 차이

자바 25부터는 사용자의 편의를 위해서 **IO.println()** 기능을 제공한다.
자바 25 버전 이상을 사용하면 **System.out.println()** 대신에 다음 사진과 같이 **IO.print()**을 사용하도록 기본 코드가 제공된다.

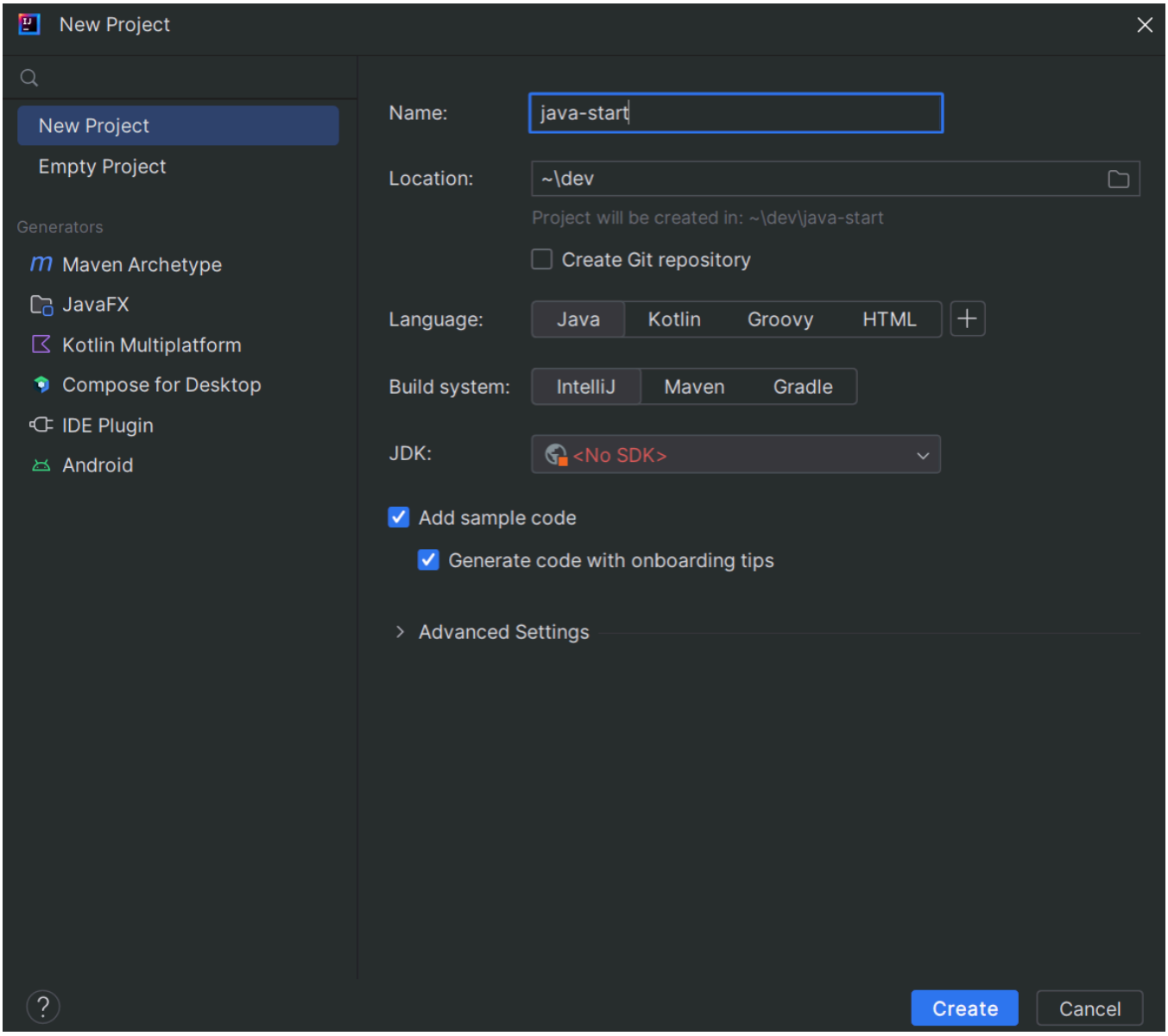


윈도우 사용자 추가 설명서

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

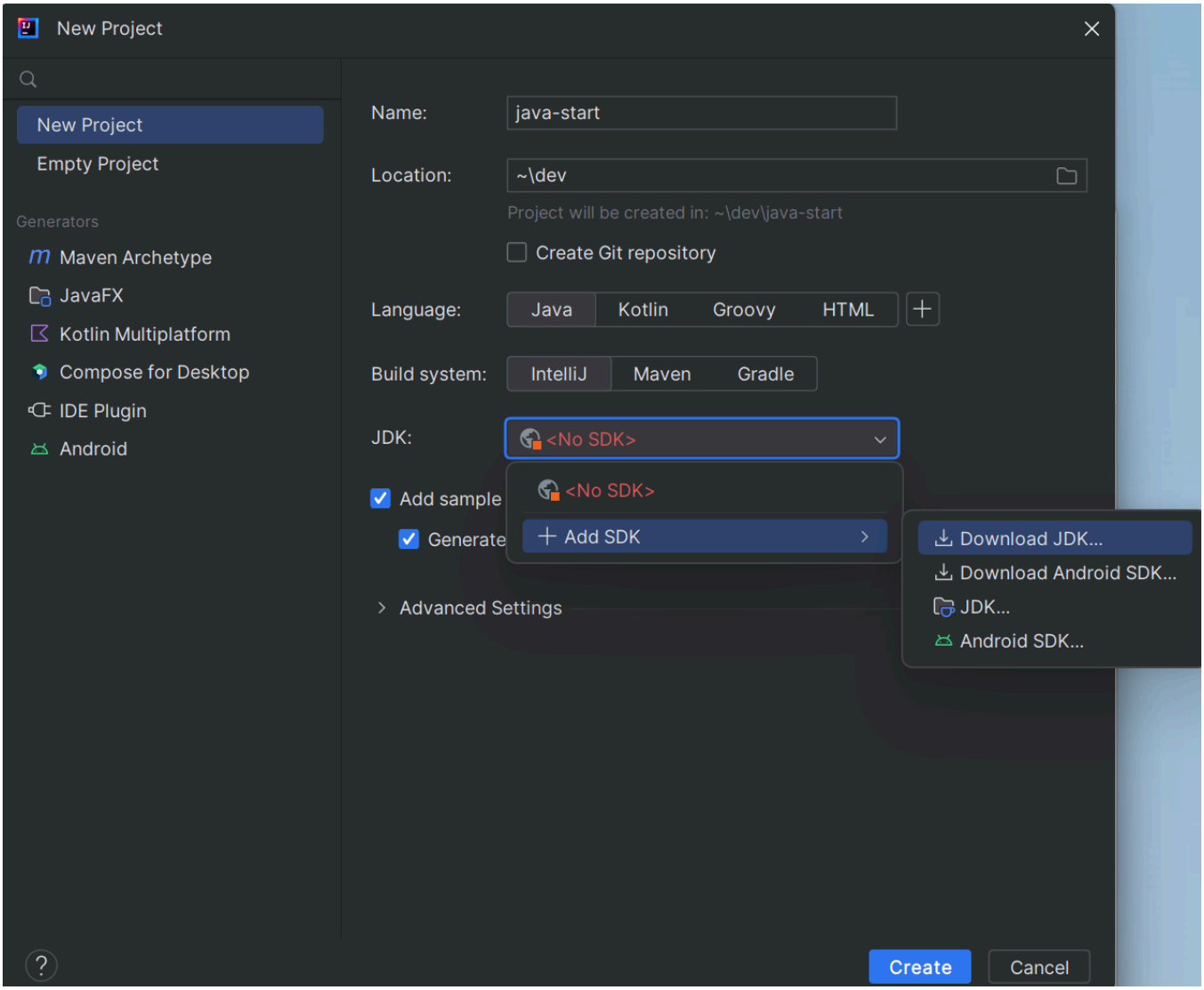


- 프로그램 시작 화면
- New Project 선택

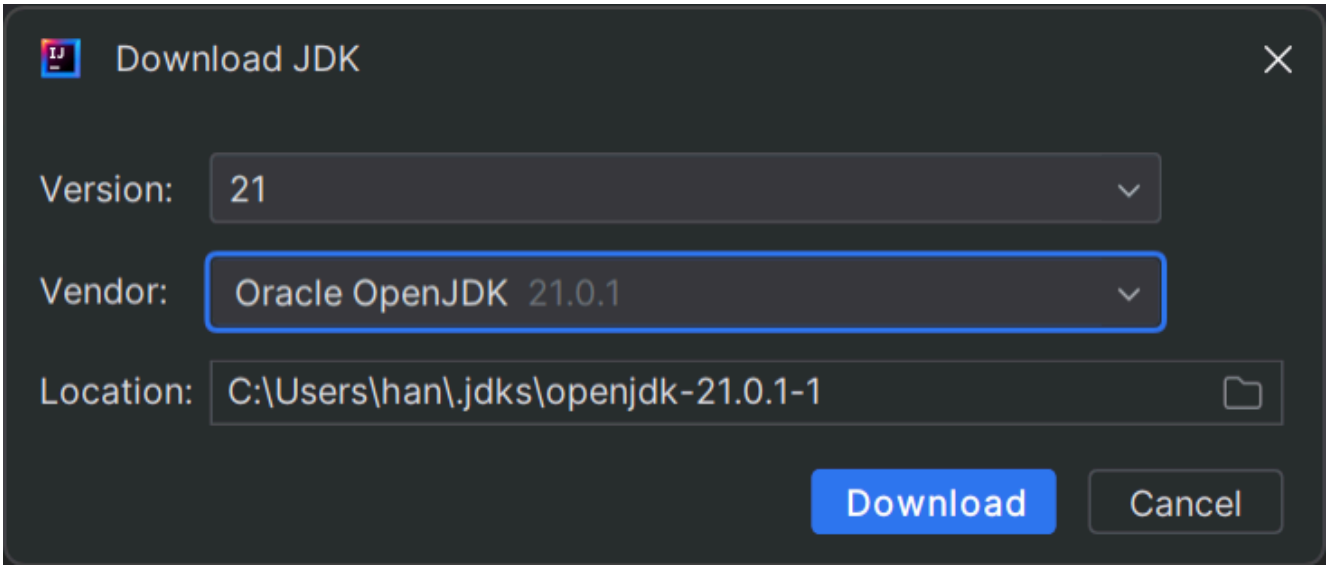


New Project 화면

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



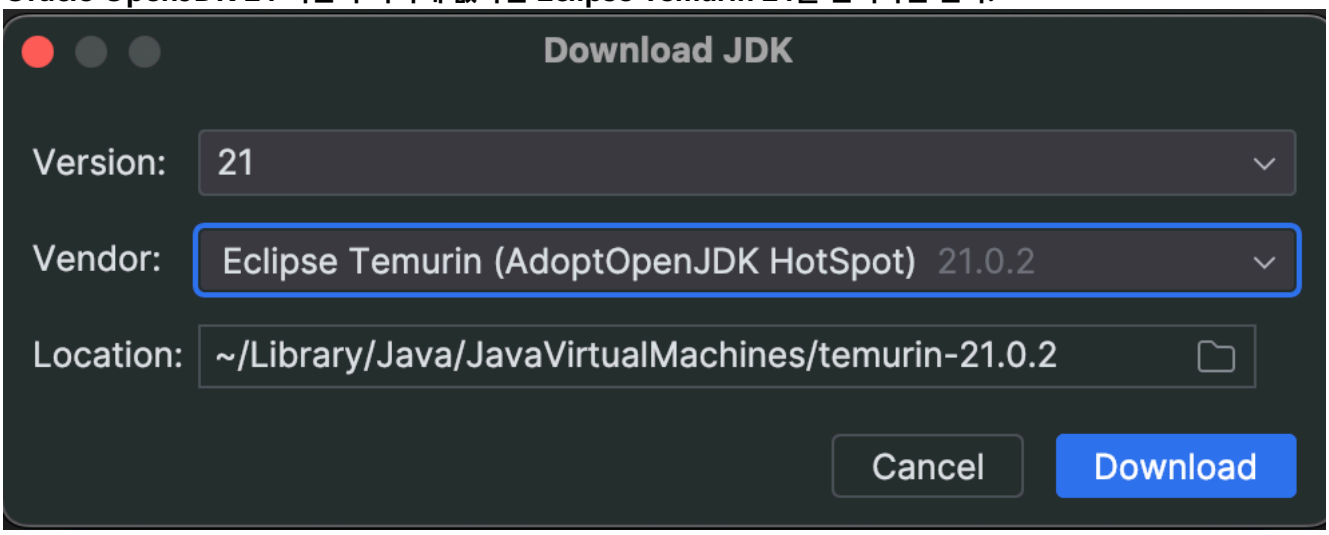
JDK 설치는 Mac과 동일하다.

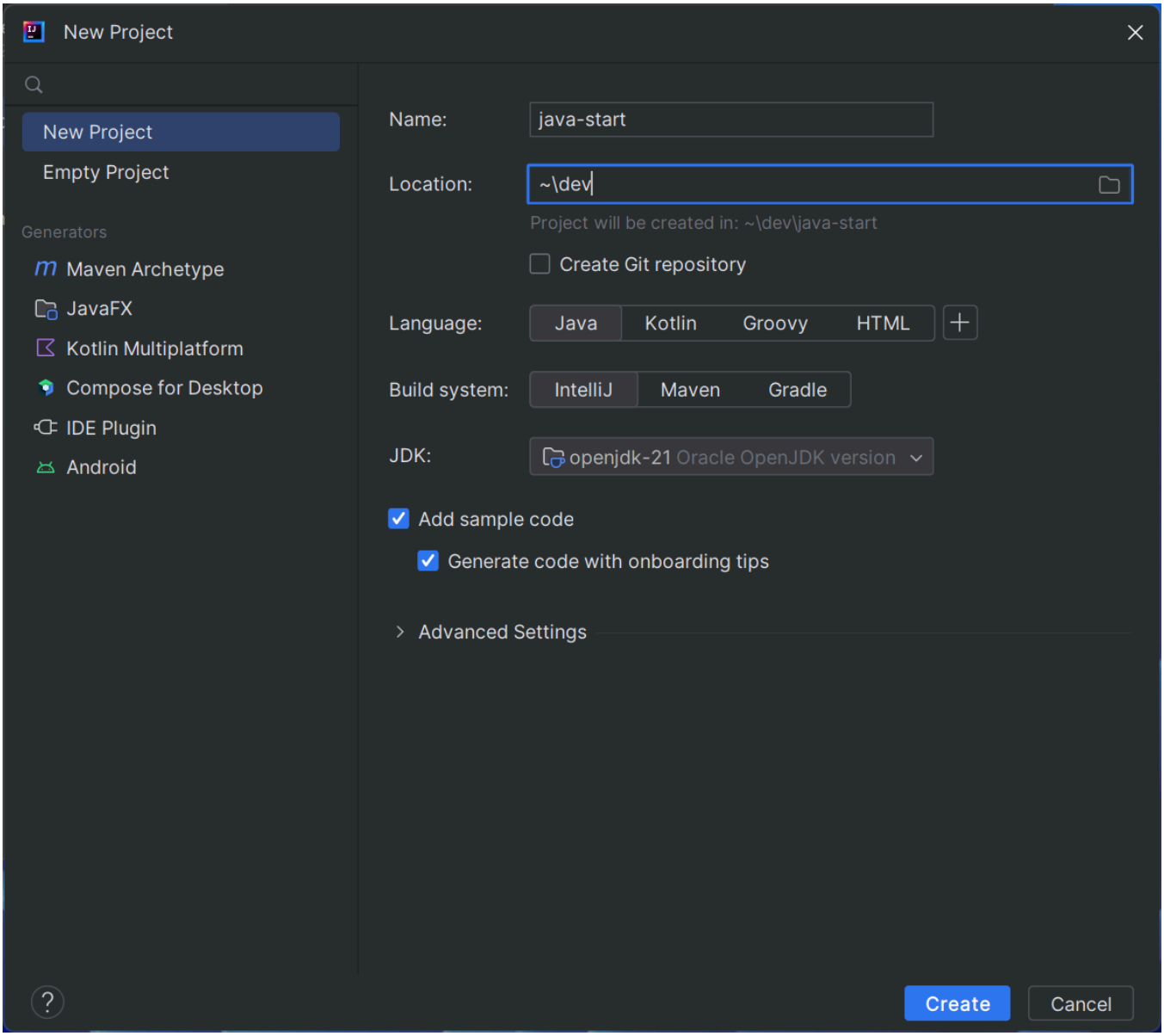


- Version: 21
- Vendor: Oracle OpenJDK
- Location은 가급적 변경하지 말자.

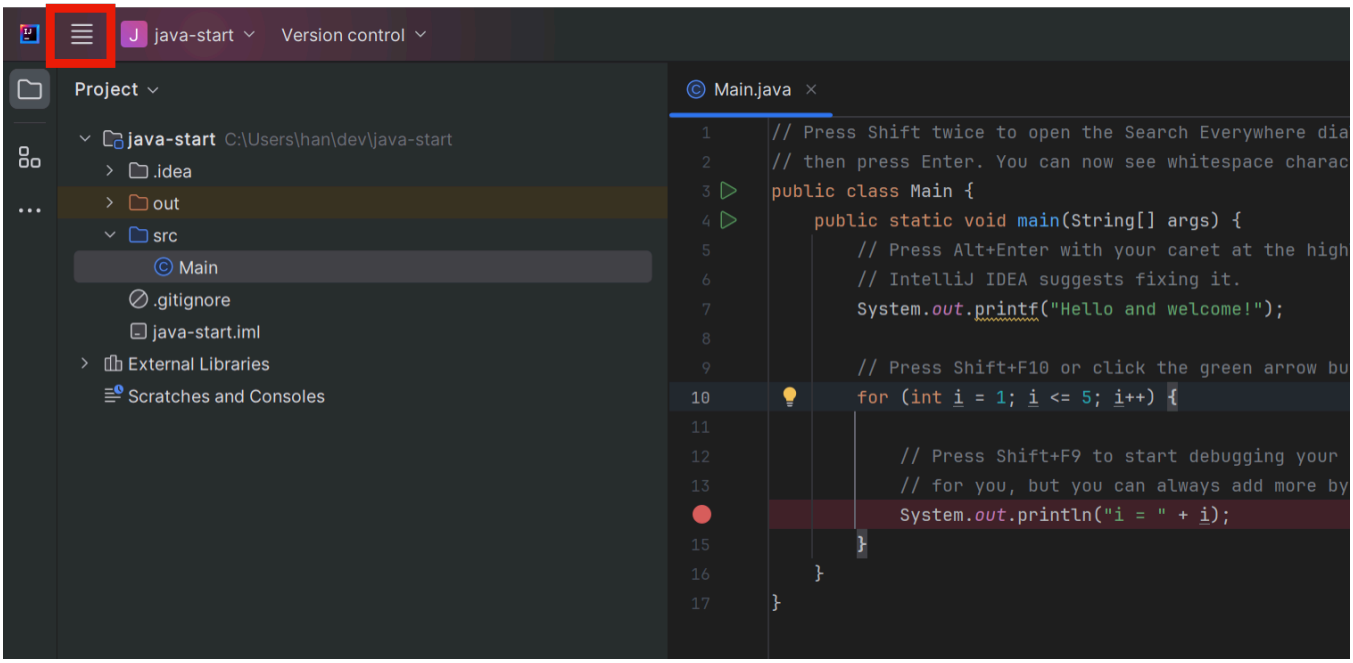
주의 - 변경 사항

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

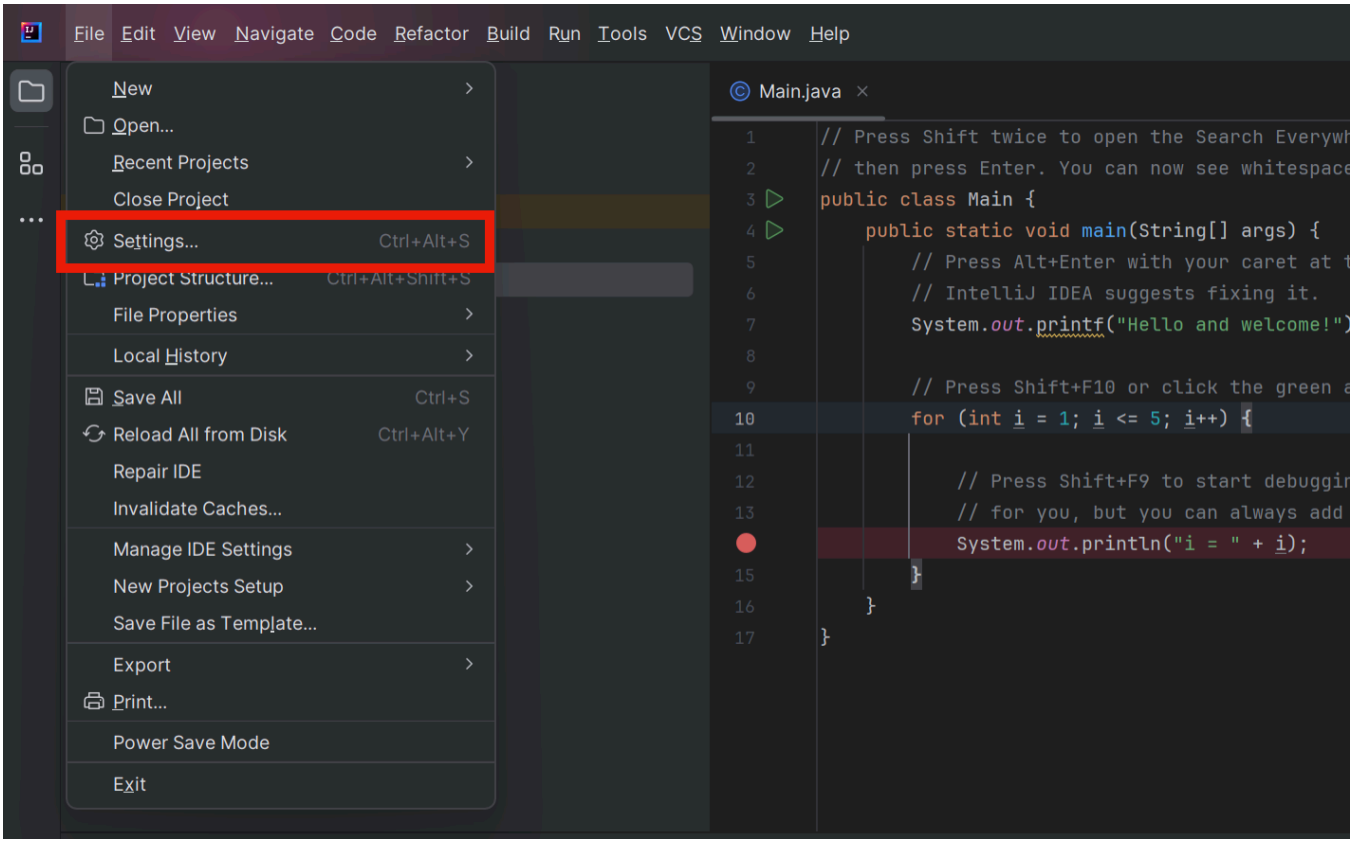




- New Project 완료 화면



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



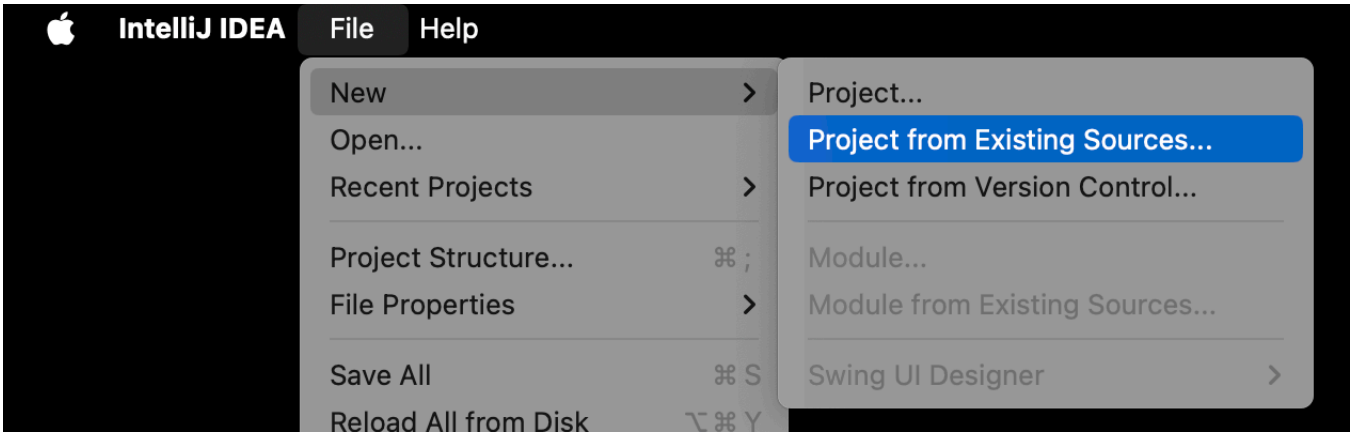
- Mac과 다르게 `Settings...` 메뉴가 `File`에 있다. 이 부분이 Mac과 다르므로 유의하자.

한글 언어팩 → 영어로 변경

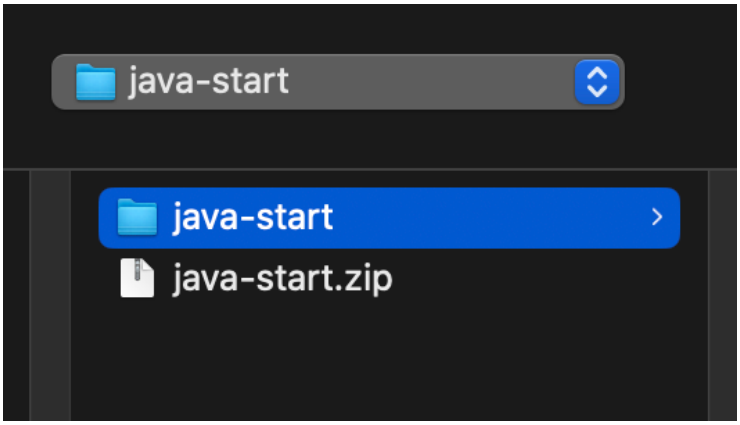
- IntelliJ는 가급적 한글 버전 대신, 영문 버전을 사용하자. 개발하면서 필요한 기능들을 검색하게 되는데, 영문으로 된 자료가 많다. 이번 강의도 영문을 기준으로 진행한다.
- 만약 한글로 나온다면 다음과 같이 영문으로 변경하자.
- **Mac**: IntelliJ IDEA(메뉴) → Settings... → Plugins → Installed
- **윈도우**: File → Settings... → Plugins → Installed
  - Korean Language Pack 체크 해제
  - OK 선택후 IntelliJ 다시 시작

다운로드 소스 코드 실행 방법

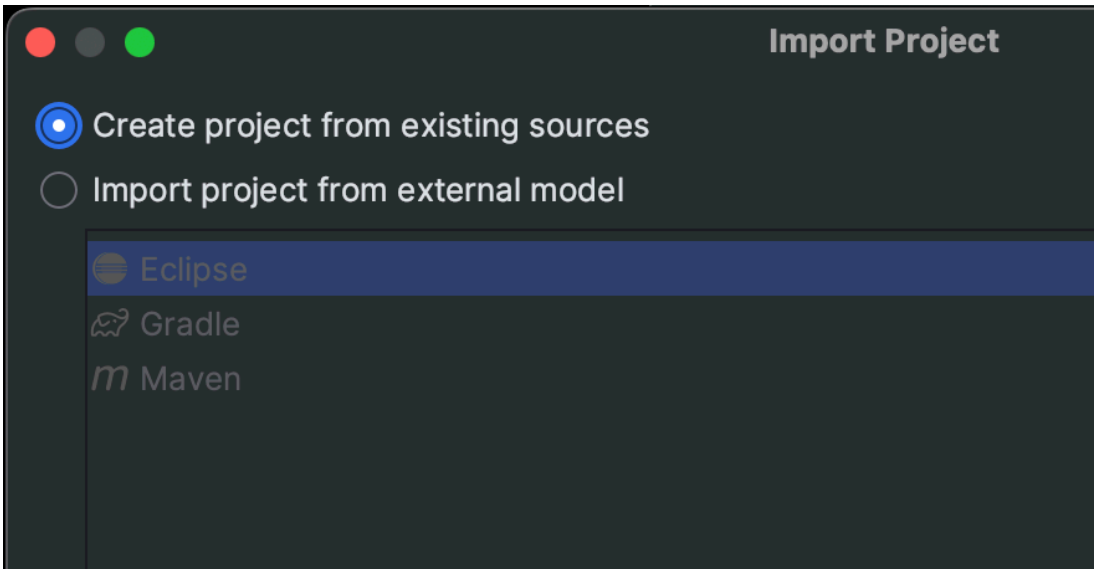
영상 참고



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

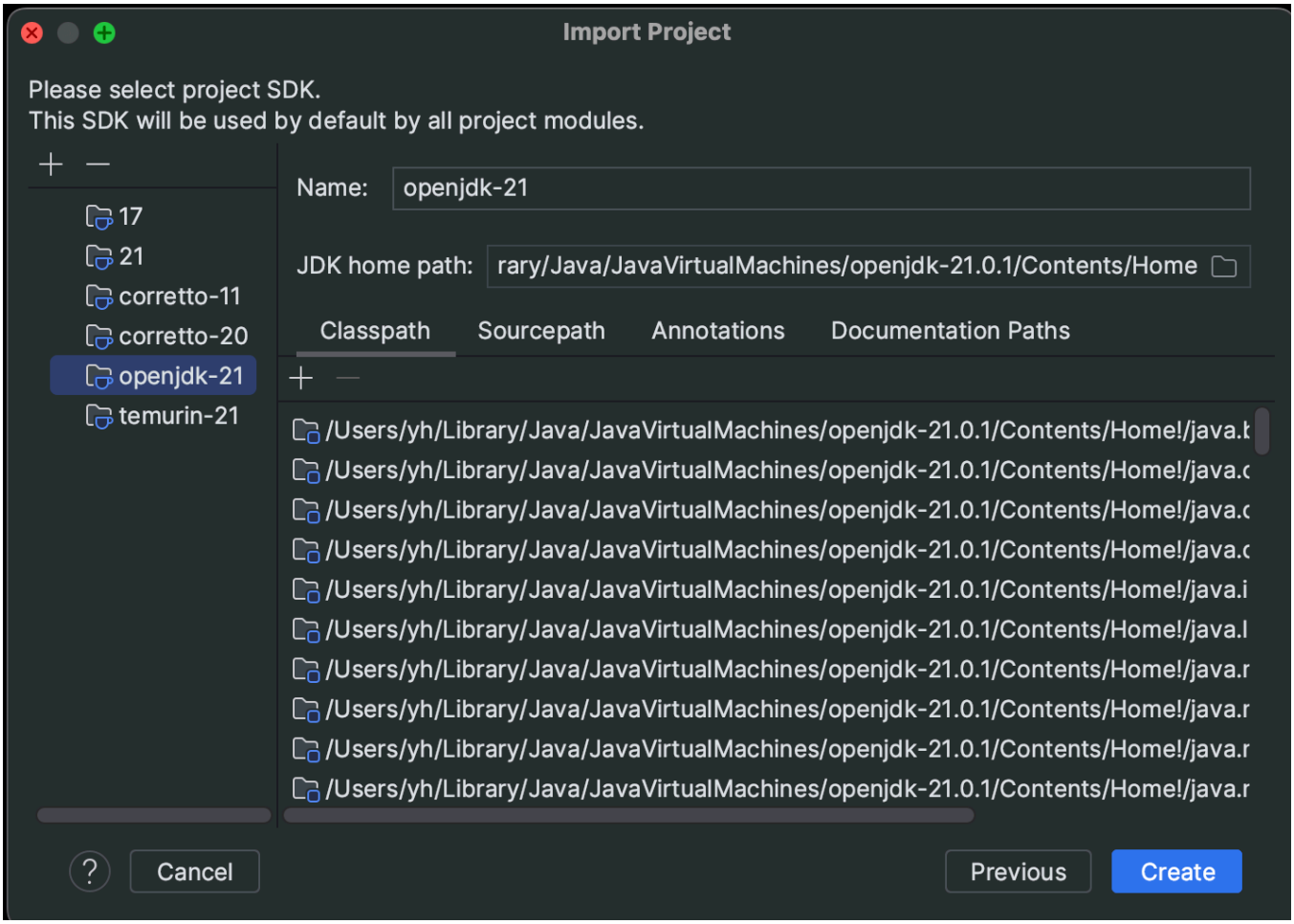


압축을 푼 프로젝트 폴더 선택

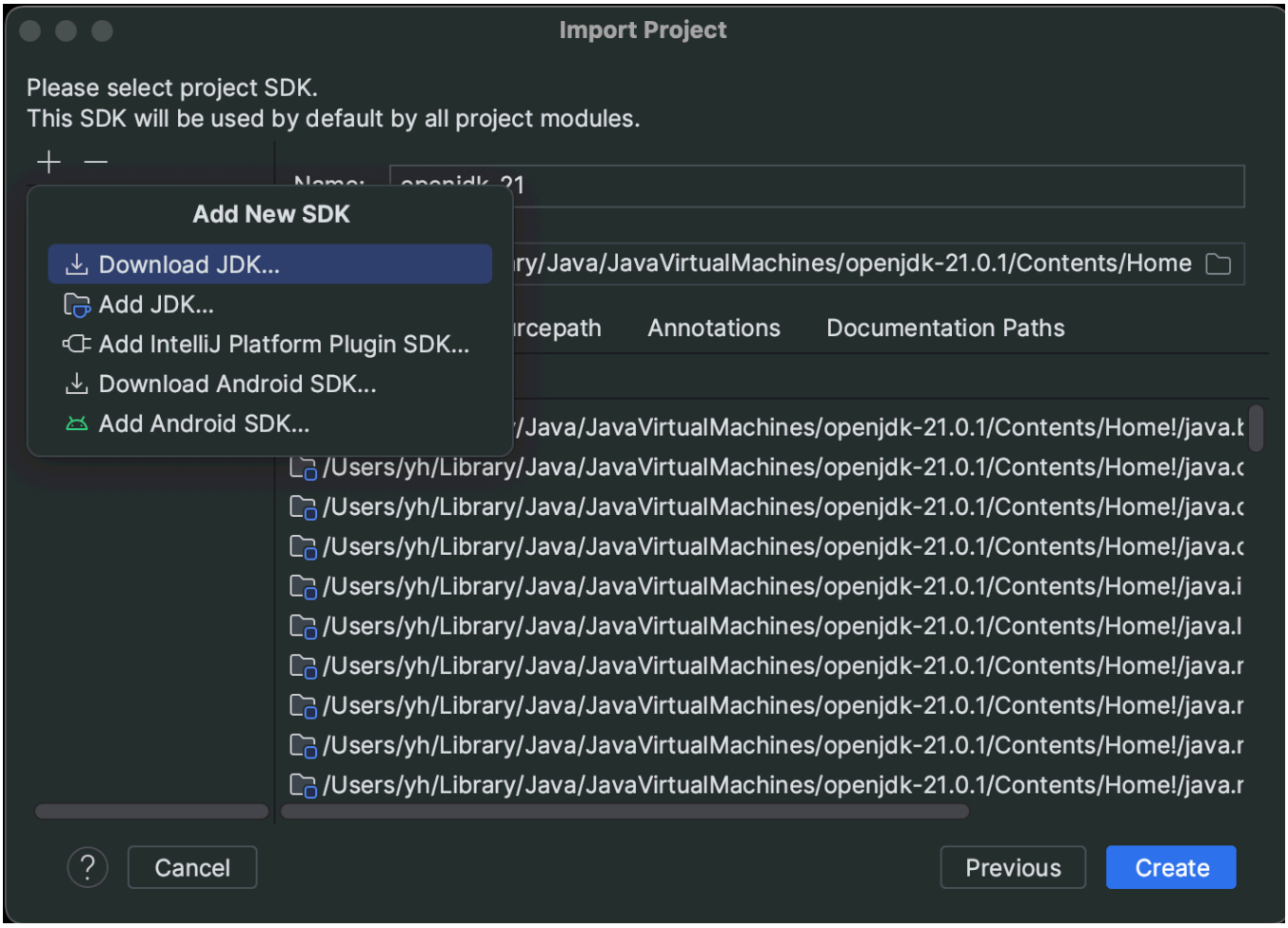


`Create project from existing sources` 선택

이후 계속 `Next` 선택



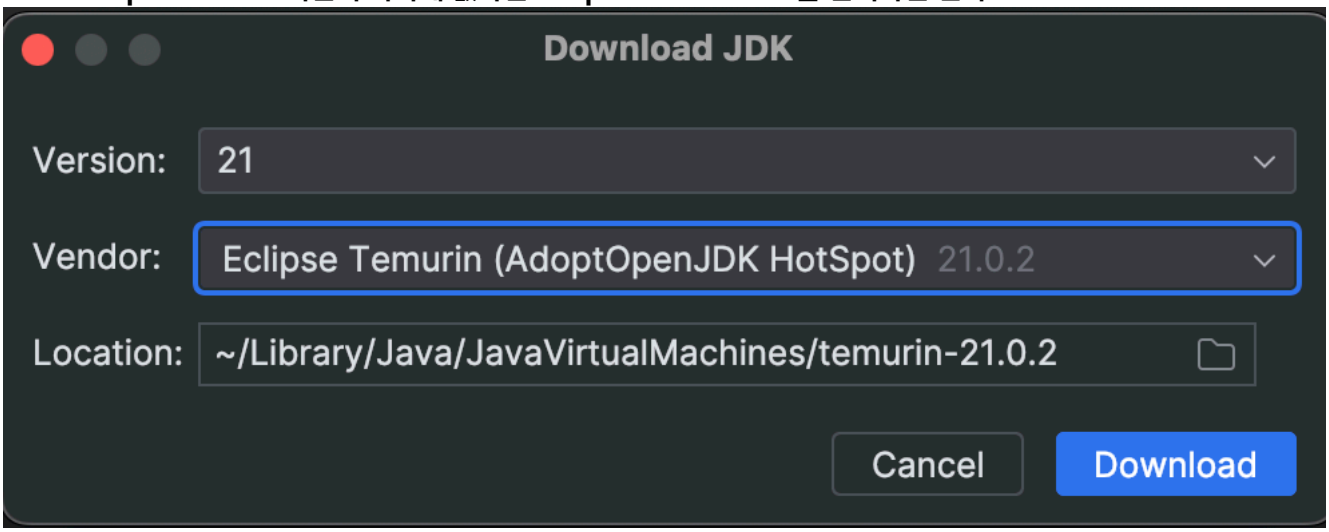
openjdk-21 선택



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

주의 - 변경 사항

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



이후 `Create` 버튼 선택

자바 프로그램 실행

HelloJava

```
public class HelloJava {

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

영상을 따라서 해당 코드를 만들고 실행해보자.

> **주의!**
> 자바 언어는 대소문자를 구분한다. 대소문자가 다르면 오류가 발생할 수 있다.

실행 결과

```
hello java
```

코드를 분석해보자. 지금 단계에서는 이 코드의 모든 내용을 이해할 수 없다. 앞으로 차근차근 하나씩 알아가보자

`public class HelloJava`

- `HelloJava`를 클래스라 한다. 클래스(class)의 개념을 학습해야 이해할 수 있다. 클래스는 뒤에서 학습한다.
- 지금은 단순히 `HelloJava.java`라는 파일을 만들었다고 이해하면 된다.
- 파일명과 클래스 이름이 같아야 한다.
- `{}` 블록을 사용해서 클래스의 시작과 끝을 나타낸다.

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

- `main` 메서드라 한다. 함수, 메서드의 개념을 학습해야 이해할 수 있다. 함수, 메서드는 뒤에서 학습한다.
- 자바는 `main(String[] args)` 메서드를 찾아서 프로그램을 시작한다.
- 지금은 단순히 `main`은 프로그램의 시작점이라고 이해하면 된다.
- `{}` 블록을 사용해서 메서드의 시작과 끝을 나타낸다.

`System.out.println("hello java");`

- `System.out.println()` : 값을 콘솔에 출력하는 기능이다.
- `"hello java"` : 자바는 문자열을 사용할 때 `"` (쌍따옴표)를 사용한다. 쌍따옴표 사이에 원하는 문자열을 감싸면 된다.
- `;` : 자바는 세미콜론으로 문장을 구분한다. 문장이 끝나면 세미콜론을 필수로 넣어주어야 한다.

참고: 괄호

- 소괄호 `()`
- 중괄호 `{}`
- 대괄호 `[]`

실행 과정

1. `HelloJava` 프로그램을 실행한다.
2. 자바는 시작점인 `main()` 메서드를 실행한다.
3. `System.out.println("hello java")` 을 만나고, 문자열 `hello java` 을 출력한다.
4. `main()` 메서드의 `{}` 블록이 끝나면 프로그램은 종료된다.

블록(block) 예시

```
public class HelloJava { //HelloJava 클래스의 범위 시작

    public static void main(String[] args) { //main() 메서드의 범위 시작
        System.out.println("hello java");
    } //main() 메서드의 범위 끝

} //HelloJava 클래스의 범위 끝
```

- 블록(`{}`)이 시작되고 끝날 때 마다 들여쓰기가 적용되어 있는 것을 확인할 수 있다. 이것은 코드를 쉽게 구분하고 이해하도록 도와주는 좋은 관례이다. 블록이 중첩될 때 마다 들여쓰기의 깊이가 추가된다.
- 들여쓰기는 보통 스페이스 4번을 사용한다. 참고로 IntelliJ IDE를 사용하면 키보드 `Tab` 을 한번 누르면 자동으로 스페이스 4번을 적용한다.
- 참고로 들여쓰기를 하지 않아도 프로그램은 작동한다. 하지만 코드를 읽기에 좋지 않다.

추가 예제

프로그램 코드에 익숙해지도록 다음 코드를 작성하고 실행해보자.

HelloJava2

```
public class HelloJava2 {

    public static void main(String[] args) {
        System.out.println("hello java1");
        System.out.println("hello java2");
        System.out.println("hello java3");
    }
}
```

실행 결과

```
hello java1
hello java2
hello java3
```

프로그램은 `main()`을 시작으로 위에서 아래로 한 줄 씩 실행된다.

주석(comment)

소스 코드가 복잡하다면 소스 코드에 대한 이해를 돕기 위해 설명을 적어두고 싶을 수 있다.

또는 특정 코드를 지우지 않고, 잠시 실행을 막아두고 싶을 때도 있다.

이럴 때 주석을 사용하면 된다. 자바는 주석이 있는 곳을 무시한다.

주석의 종류

- 한 줄 주석 (single line comment)
  - `//` 기호로 시작한다. 이 기호 이후의 모든 텍스트는 주석으로 처리된다.
- 여러 줄 주석(multi line comment)
  - `/*`로 시작하고 `*/`로 끝난다. 이 사이의 모든 텍스트는 주석으로 처리된다.

CommentJava

```
public class CommentJava {

    /*
```

```
    주석을 설명하는 부분입니다.
    */
    public static void main(String[] args) {
        System.out.println("hello java1"); //hello java1을 출력합니다. (한 줄 주석 - 부분 적용)
        //System.out.println("hello java2"); 한 줄 주석 - 라인 전체 적용

        /* 여러 줄 주석
        System.out.println("hello java3");
        System.out.println("hello java4");
         */
    }
}
```

실행 결과

```
hello java1
```

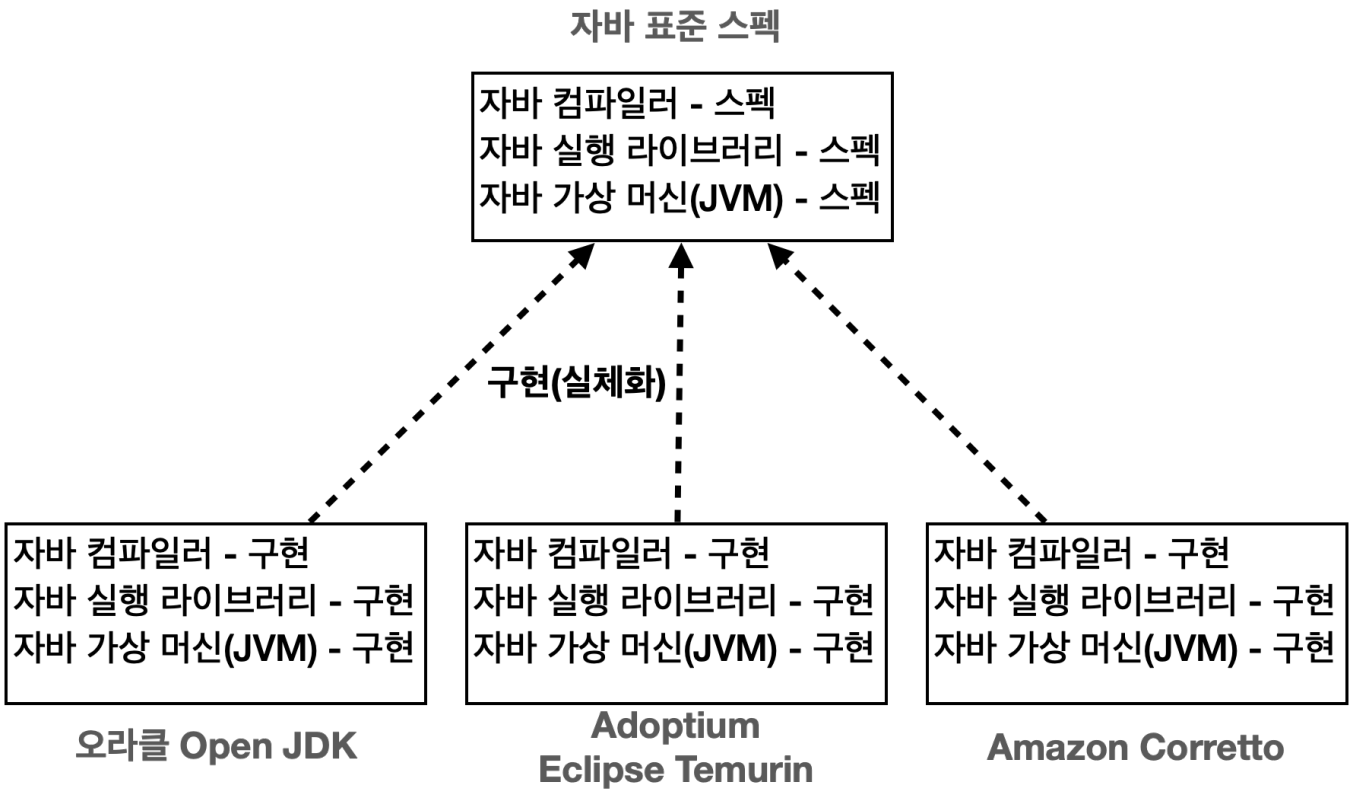
주석으로 처리한 코드가 실행되지 않은 것을 확인할 수 있다.

주석은 쉽게 이야기해서 자바 프로그램이 읽지 않고 무시하는 부분이다. 사람이 읽기 위해서 사용한다.

자바란?

자바 표준 스펙

자바 표준 스펙과 구현

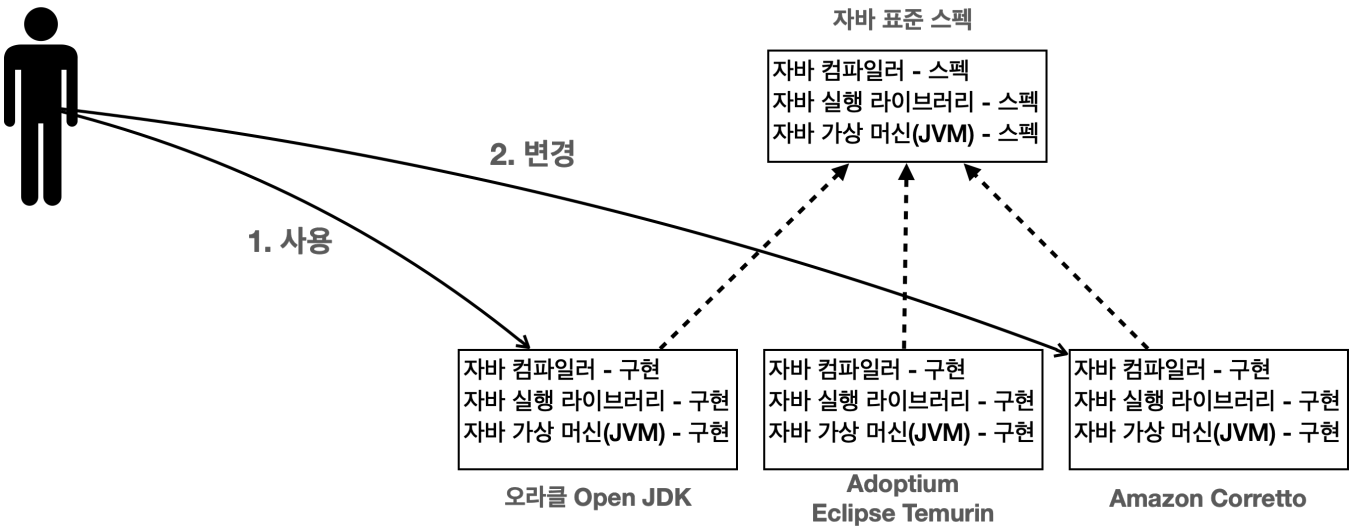


자바는 **표준 스펙**과 **구현**으로 나눌 수 있다.

- 자바 표준 스펙
  - 자바는 이렇게 만들어야 한다는 설계도이며, 문서이다.
  - 이 표준 스펙을 기반으로 여러 회사에서 실제 작동하는 자바를 만든다.
  - 자바 표준 스펙은 자바 커뮤니티 프로세스(JCP)를 통해 관리된다.
- 다양한 자바 구현
  - 여러 회사에서 자바 표준 스펙에 맞추어 실제 작동하는 자바 프로그램을 개발한다.
  - 각각 장단점이 있다. 예를 들어 Amazon Corretto는 AWS에 최적화 되어 있다.
  - 각 회사들은 대부분 윈도우, MAC, 리눅스 같이 다양한 OS에서 작동하는 버전의 자바도 함께 제공한다.

> 참고: 다양한 자바 구현에 대해서는 다음 사이트를 참고하자.
>
> https://whichjdk.com/ko

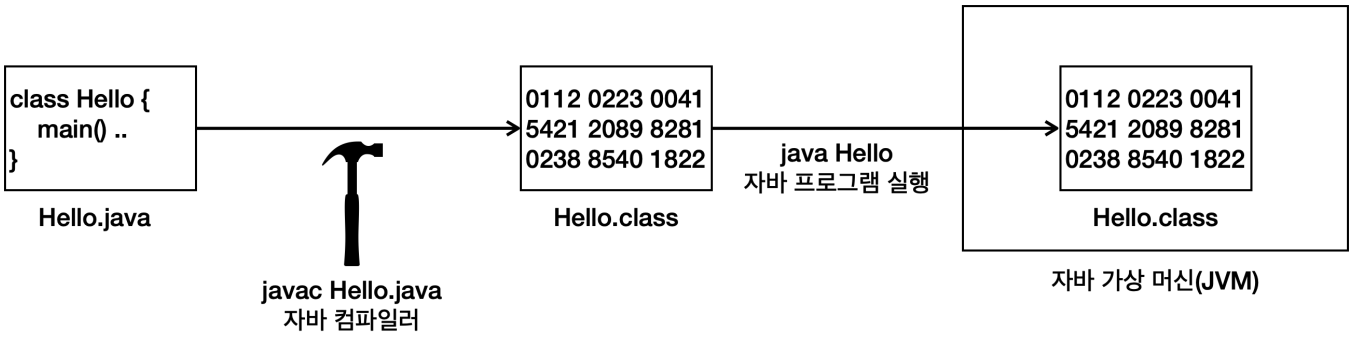
변경의 용이



- 자바 구현들은 모두 표준 스펙에 맞도록 개발되어 있다. 따라서 오라클 Open JDK를 사용하다가 Amazon Corretto 자바로 변경해도 대부분 문제 없이 동작한다.

참고: 학습 단계에서는 어떤 자바를 사용하든 크게 상관이 없다.

컴파일과 실행

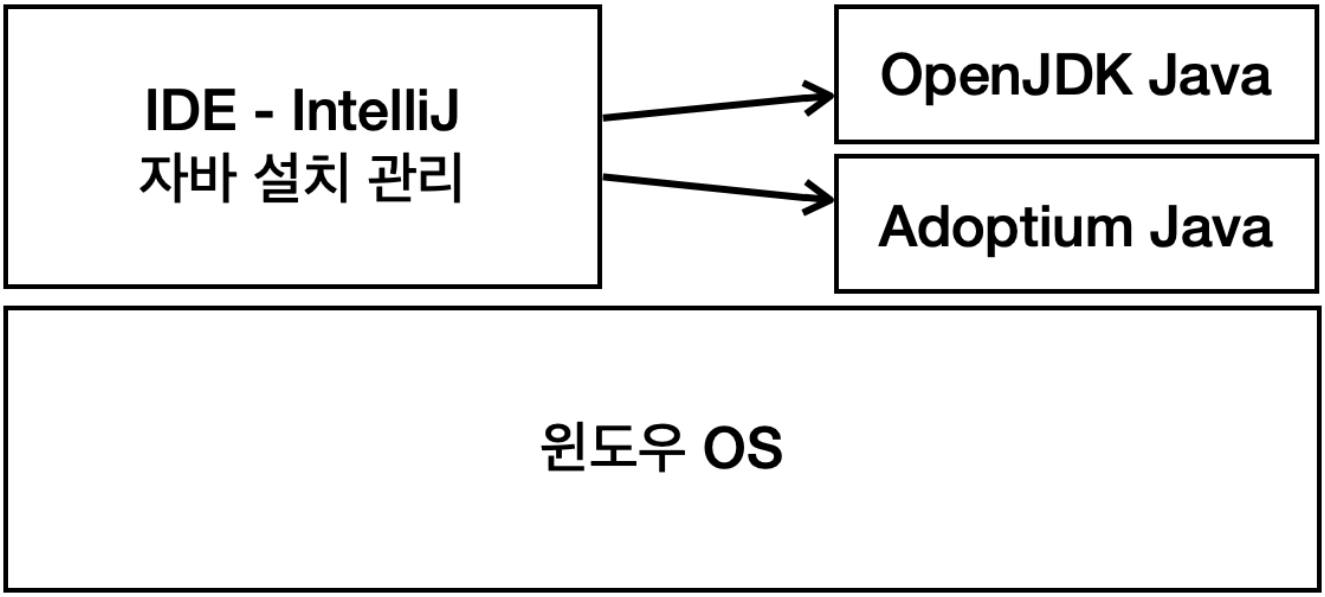


자바 프로그램은 컴파일과 실행 단계를 거친다.

- `Hello.java`와 같은 자바 소스 코드를 개발자가 작성한다.
- 자바 컴파일러를 사용해서 소스 코드를 컴파일 한다.
  - 자바가 제공하는 `javac`라는 프로그램을 사용한다.
  - `.java` ➜ `.class` 파일이 생성된다.
  - 자바 소스 코드를 바이트코드로 변환하며 자바 가상 머신에서 더 빠르게 실행될 수 있게 최적화하고 문법 오류도 검출한다.
- 자바 프로그램을 실행한다.
  - 자바가 제공하는 `java`라는 프로그램을 사용한다.
  - 자바 가상 머신(JVM)이 실행되면서 프로그램이 작동한다.

IDE와 자바

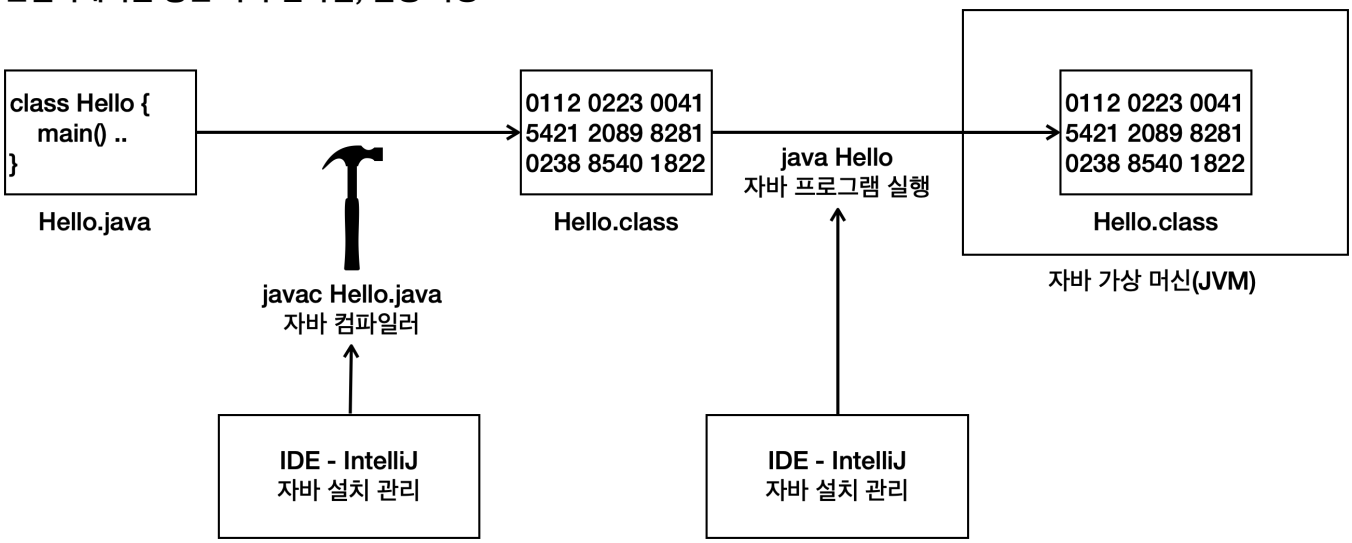
인텔리제이를 통한 자바 설치 관리



- 인텔리제이는 내부에 자바를 편리하게 설치하고 관리할 수 있는 기능을 제공한다.
- 이 기능을 사용하면 인텔리제이를 통해 자바를 편리하게 다운로드 받고 실행할 수 있다.

> 참고: 자바를 OS에 직접 설치해도 되지만, 처음 프로그래밍을 시작하는 사람에게 이 과정은 매우 번거롭다. 자바를 직접 설치하는 경우 환경 설정이 복잡하다. 그래서 자바를 설치하다가 잘 안되어서 시작도 하기 전에 포기하는 경우가 많다. 자바 언어를 배우는 단계라면 인텔리제이를 통해 자바를 설치하는 정도면 충분하다. 자바를 직접 설치하고 실행하는 내용은 별도로 다룬다.

인텔리제이를 통한 자바 컴파일, 실행 과정

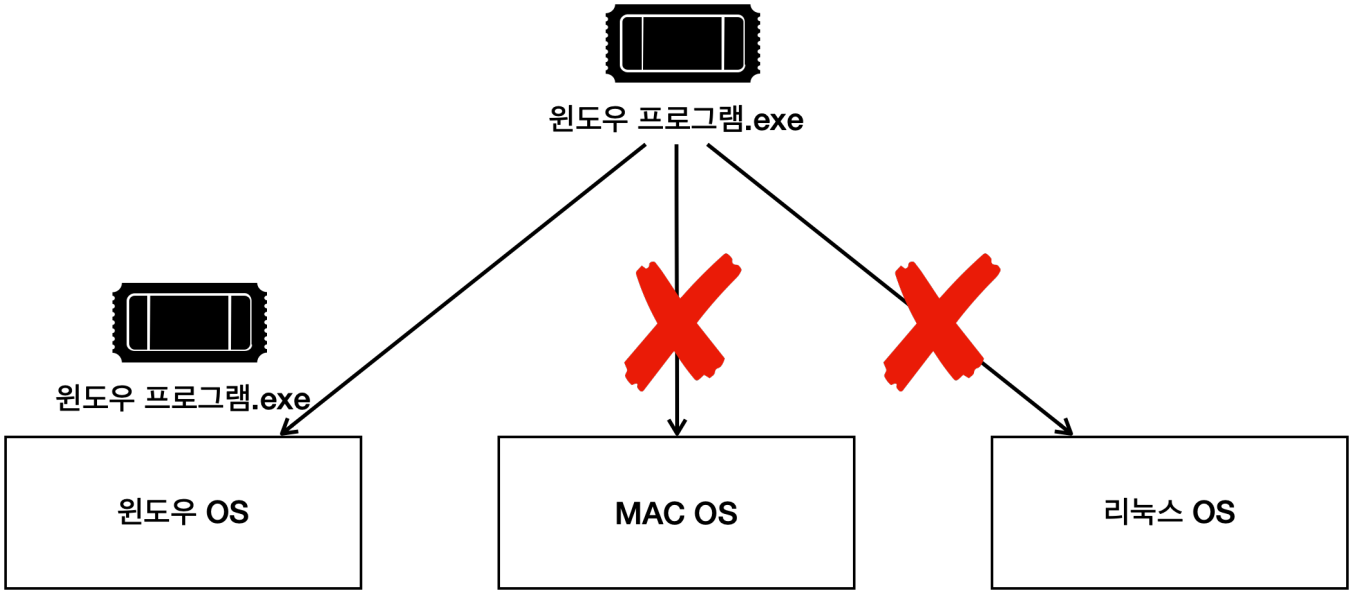


- **컴파일**
  - 자바 코드를 컴파일 하려면 `javac` 라는 프로그램을 직접 사용해야 하는데, 인텔리제이는 자바 코드를 실행할 때 이 과정을 자동으로 처리해준다.

        - 예) `javac Hello.java`
    - 인텔리제이 화면에서 프로젝트에 있는 `out` 폴더에 가보면 컴파일된 `.class` 파일이 있는 것을 확인할 수 있다.
- **실행**
    - 자바를 실행하려면 `java`라는 프로그램을 사용해야 한다. 이때 컴파일된 `.class` 파일을 지정해주면 된다.
    - 예) `java Hello`, 참고로 확장자는 제외한다.
- 인텔리제이에서 자바 코드를 실행하면 컴파일과 실행을 모두 한번에 처리한다.
- 인텔리제이 덕분에 매우 편리하게 자바 프로그램을 개발하고, 학습할 수 있다.

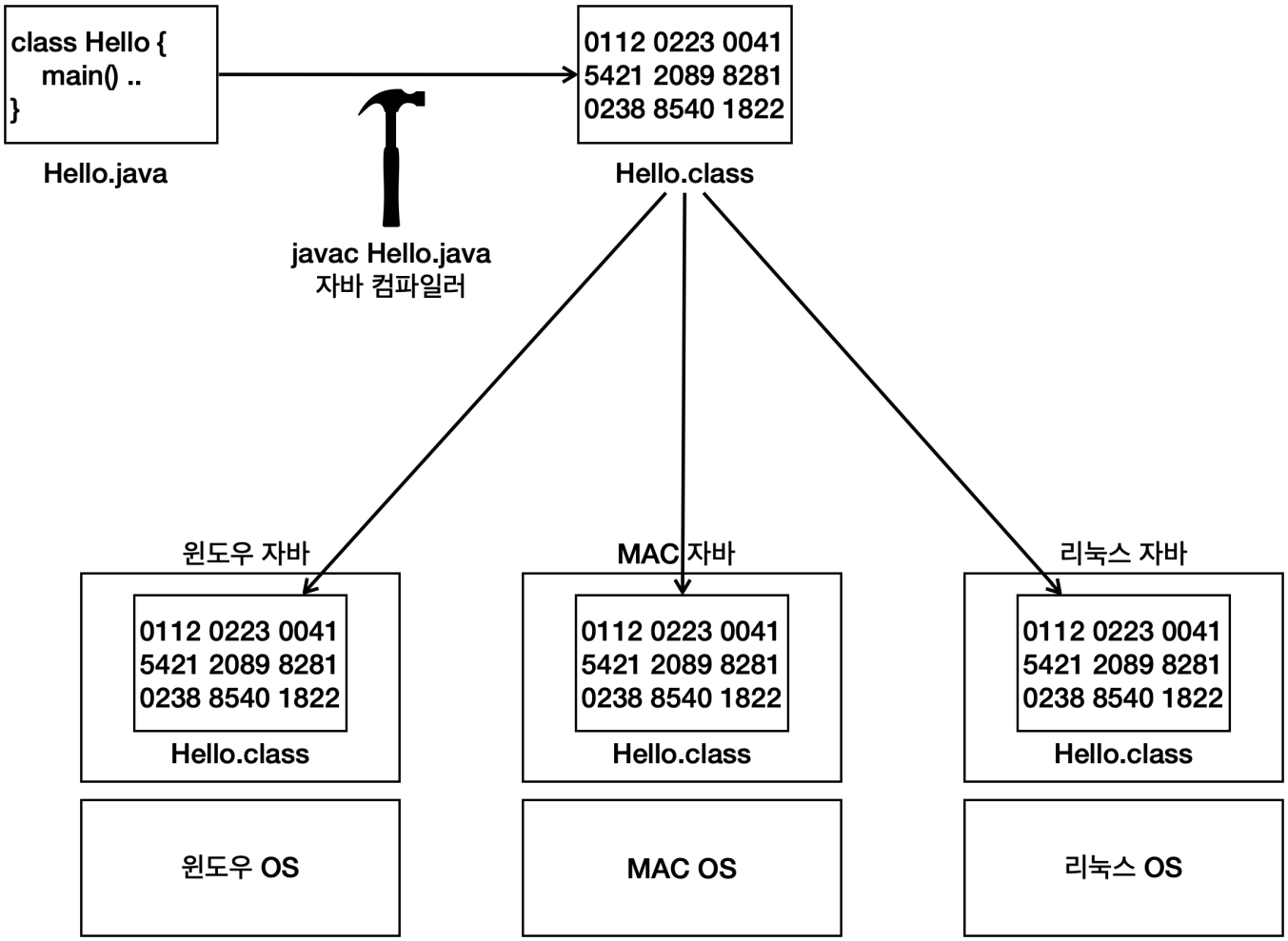
자바와 운영체제 독립성

일반적인 프로그램



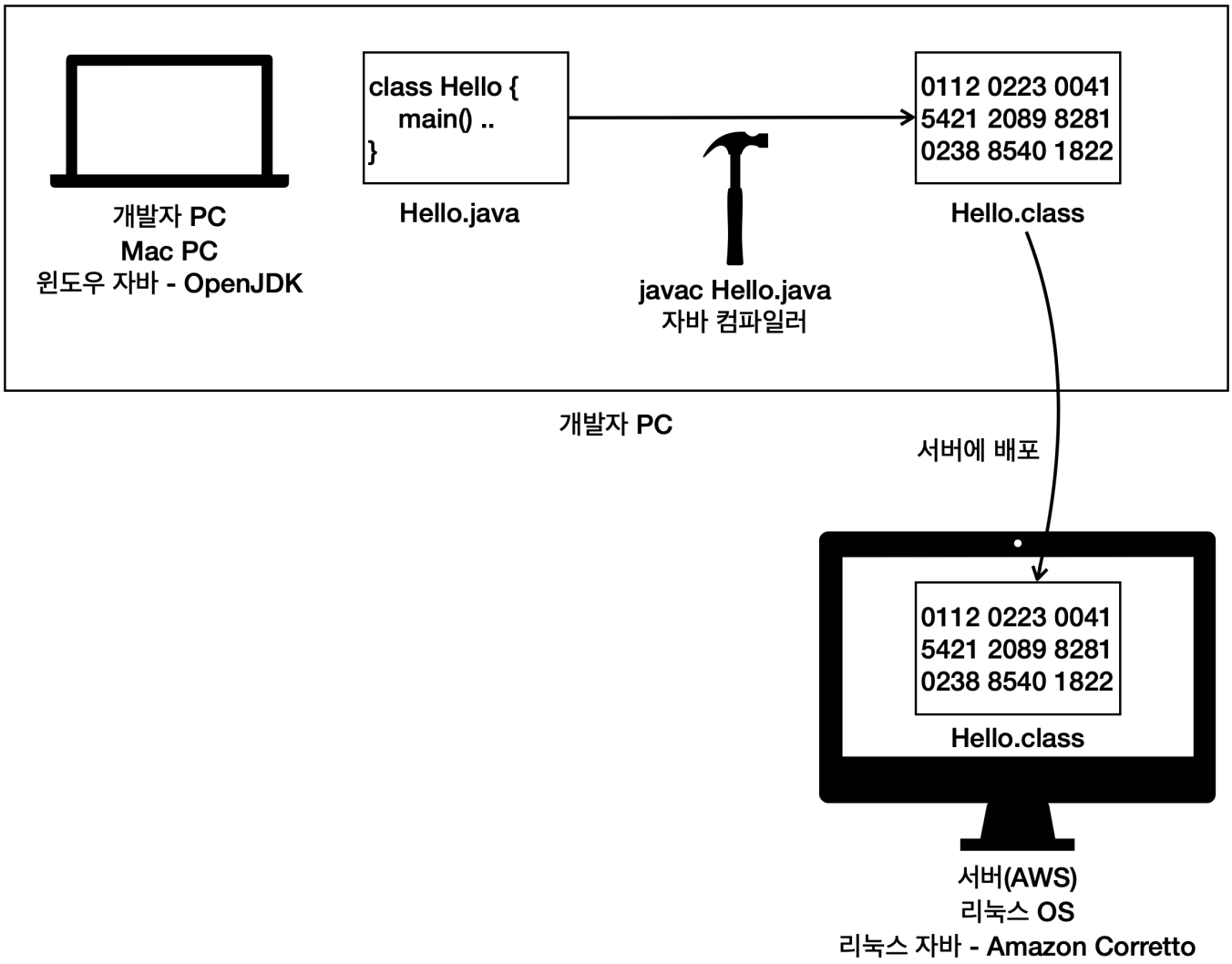
- 일반적인 프로그램은 다른 운영체제에서 실행할 수 없다.
- 예를 들어서 윈도우 프로그램은 MAC이나 리눅스에서 작동하지 않는다.
- 왜냐하면 윈도우 프로그램은 윈도우 OS가 사용하는 명령어들로 구성되어 있기 때문이다. 해당 명령어는 다른 OS와는 호환되지 않는다.

자바 프로그램



- 자바 프로그램은 자바가 설치된 모든 OS에서 실행할 수 있다.
- 자바 개발자는 특정 OS에 맞추어 개발을 하지 않아도 된다. 자바 개발자는 자바에 맞추어 개발하면 된다. OS 호환성 문제는 자바가 해결한다. `Hello.class`와 같이 컴파일된 자바 파일은 모든 자바 환경에서 실행할 수 있다.
- 윈도우 자바는 윈도우 OS가 사용하는 명령어들로 구성되어 있다. MAC이나 리눅스 자바도 본인의 OS가 사용하는 명령어들로 구성되어 있다. 개발자는 각 OS에 맞도록 자바를 설치하기만 하면 된다.

자바 개발과 운영 환경



- 개발할 때 자바와 서버에서 실행할 때 다른 자바를 사용할 수 있다.
- 개발자들은 개발의 편의를 위해서 윈도우나 MAC OS를 주로 사용한다.
- 서버는 주로 리눅스를 사용한다. 만약 AWS를 사용한다면 Amazon Corretto 자바를 AWS 리눅스 서버에 설치하면 된다.
- 자바의 운영체제 독립성 덕분에 각각의 환경에 맞추어 자바를 설치하는 것이 가능하다.