2. 자바개발환경 구축하기

2.1 자바 개발도구(JDK)설치하기

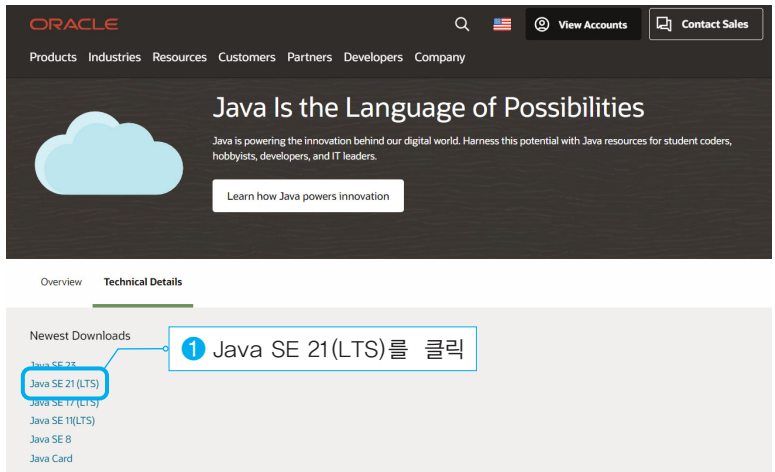
자바로 프로그래밍을 하기위해서는 먼저 JDK(Java Development Kit)를 설치해야 한다. JDK를 설치하면, 자바가상머신(Java Virtual Machine, JVM)과 자바클래스 라이브러리(Java API)외에 자바를 개발하는데 필요한 프로그램들이 설치된다.

JDK 다운로드 받기

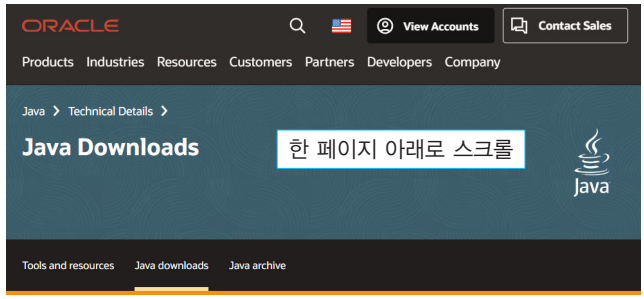
이 책을 학습하기 위해서는 JDK 21 이상의 버전이 필요하며, http://java.sun.com/에서 다운로드 받아서 설치할 것이다. JDK를 설치하는 것만으로는 자바를 학습하기에 불편하기 때문에, 보다 편리한 개발환경을 제공하는 통합 개발 환경(IDE)가 필요한다. 본인에게 익숙한 것을 사용해도 되지만 가능하면 이 책을 학습하는 동안 인텔리제이(IntelliJ IDEA)을 추천한다. 인텔리제이를 설치하는 방법은 JDK를 설치한 다음에 설명할 것이다.

| 참고 | JDK를 설치하는 방법이 변경된 경우 https://github.com/castello/javajungsuk4 에서 JDK21_설치방법.pdf을 확인

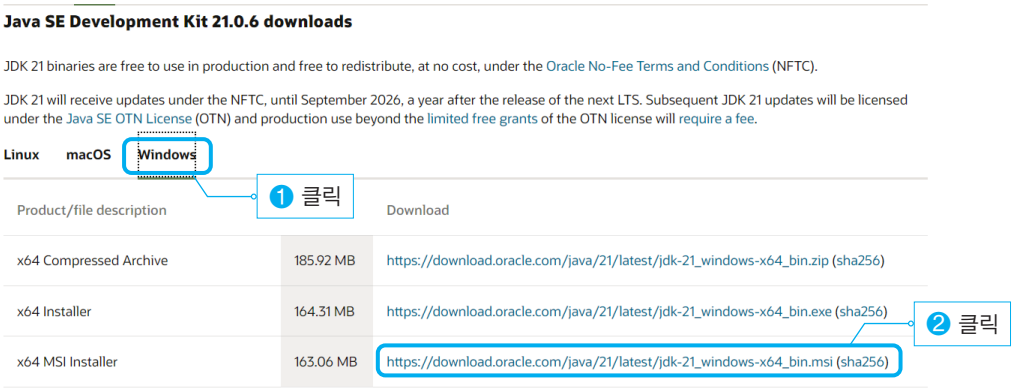
1 브라우저를 열고 http://java.sun.com을 방문하면 아래와 같은 화면이 나온다. 아래의 화면에서 'Java SE 21(LTS)'를 클릭하자.



2 아래의 페이지가 나타나면 아래로 약간 스크롤하자. 두번째 그림처럼 다운로드 받을 JDK의 종류를 선택하는 화면이 나타난다.

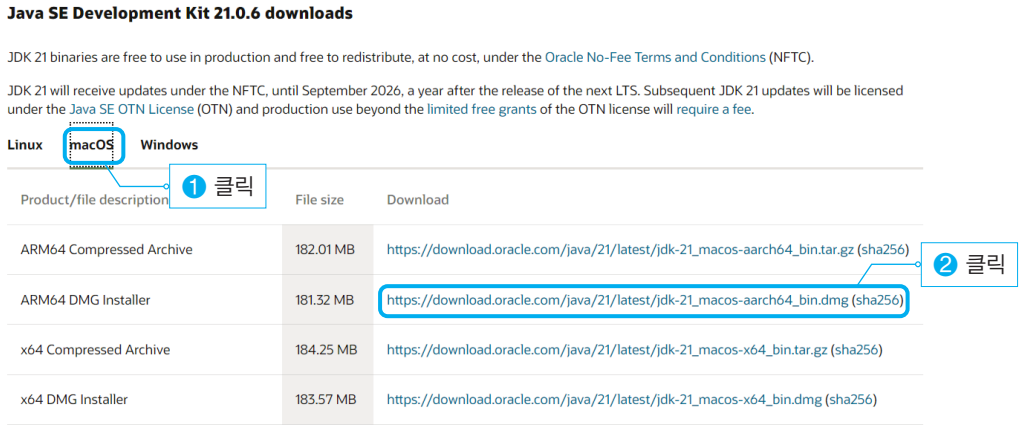


윈도즈 사용자는 아래의 그림과 같이 Windows를 클릭하고, 아래의 세 번째 링크인 'x64 MSI Installer'를 클릭하면, 다운로드가 시작된다.



맥OS 사용자는 아래의 그림에서 macOS를 클릭하고, 두 번째 링크인 'ARM64 DMG Installer'를 클릭하면, 다운로드가 시작된다.

| 참고 | Intel CPU가 장착된 컴퓨터의 경우 네 번째 링크인 '64 DMG Installer'를 클릭해야 한다.

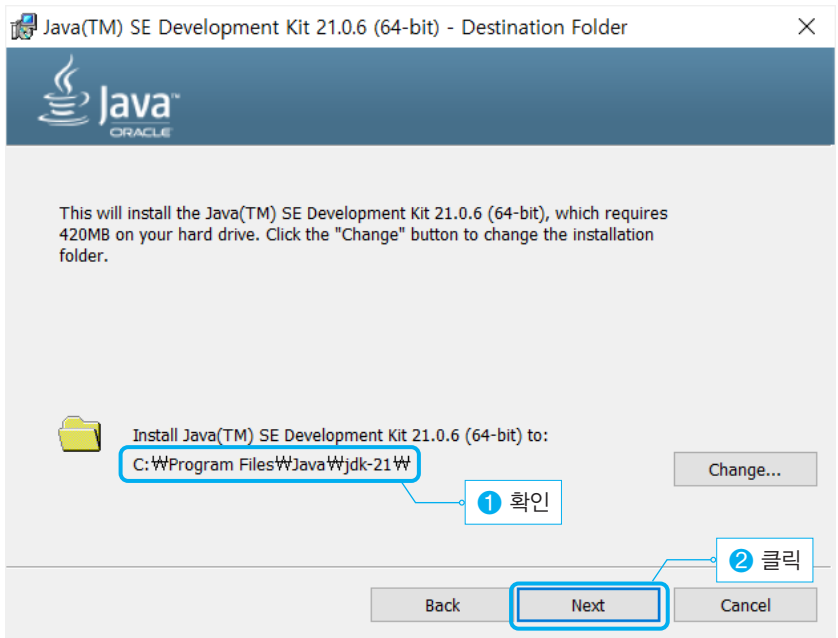


JDK 설치하기 – 윈도우즈

이제 다운로드 받은 JDK를 설치해보자. 윈도우즈에 JDK를 설치하는 방법을 먼저 설명하고 그 다음에 맥OS에 JDK를 설치하는 방법을 설명한다.

1 다운로드 받은 'jdk-21_windows-x64_bin.msi'를 실행하면 다음과 같은 화면을 볼 수 있다. 'Next >'버튼을 클릭하자

2 JDK를 설치할 위치를 묻는 화면인데, 설치될 위치를 변경하려면, 'Change...'버튼을 누르면 된다. 그냥 설치될 위치 'C:₩Program Files₩Java₩jdk-21'만 확인하고, Next버튼을 클릭하자.



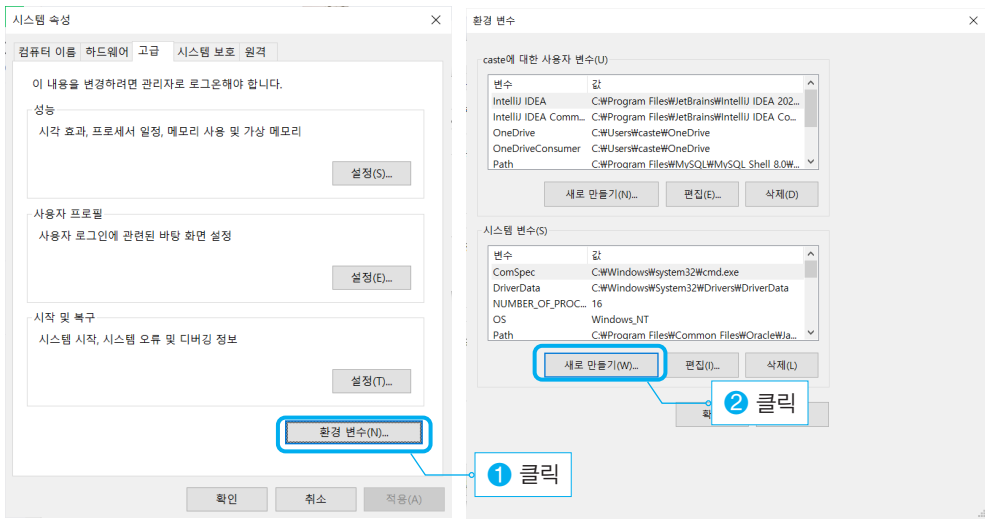
3 아래와 같은 화면이 나타나면서 설치가 시작되는데, 잠시 후 설치가 모두 끝나고 두 번째 화면이 나타난다. 그러면 설치가 잘 끝난 것이다. 'Close'버튼을 누르자.

4 설치가 끝났으나 한가지 설정이 남았다. 제어판을 열고, 검색창에 '환경변수'라고 입력하자.

참고 | 제어판은 윈도우키를 누르고, 찾기에 '제어판'이라고 입력하면 찾을 수 있다.

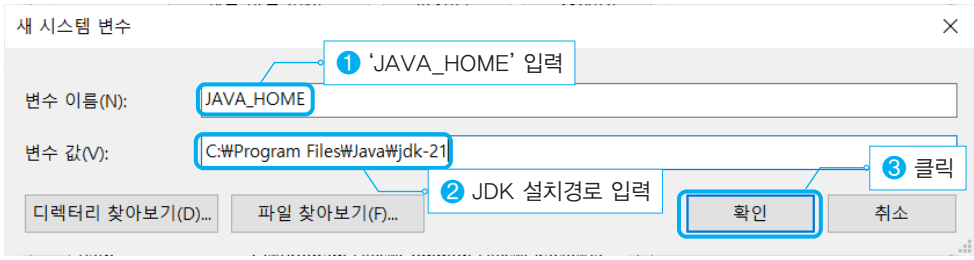
아래의 화면에서 '시스템 환경 변수 편집'을 클릭하면, '시스템 속성' 화면이 나타난다.

5 아래 왼쪽의 화면에서 '환경 변수(N)...'버튼을 클릭하면 오른쪽과 같은 '환경 변수'화면이 나타나는데 여기서 '새로 만들기(W)...'버튼을 누르자.

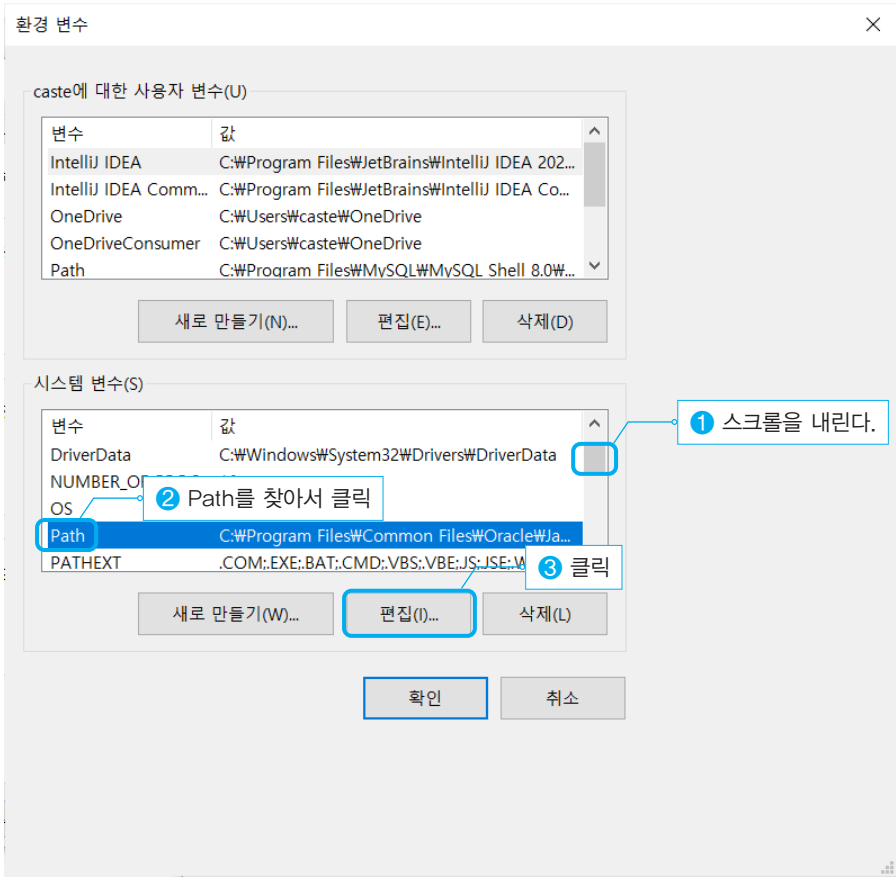


6 아래와 같은 화면이 나타나면, '변수 이름'으로 'JAVA_HOME'을 입력하고 '변수 값'에는 아까 JDK를 설치한 경로인 'C:₩Program Files₩Java₩jdk-21'을 입력하자. 만일 JDK를 다른 곳에 설치했으면, 설치한 경로를 입력해야 한다. 입력한 내용을 다시 한번 확인하고 '확인'을 누르자.

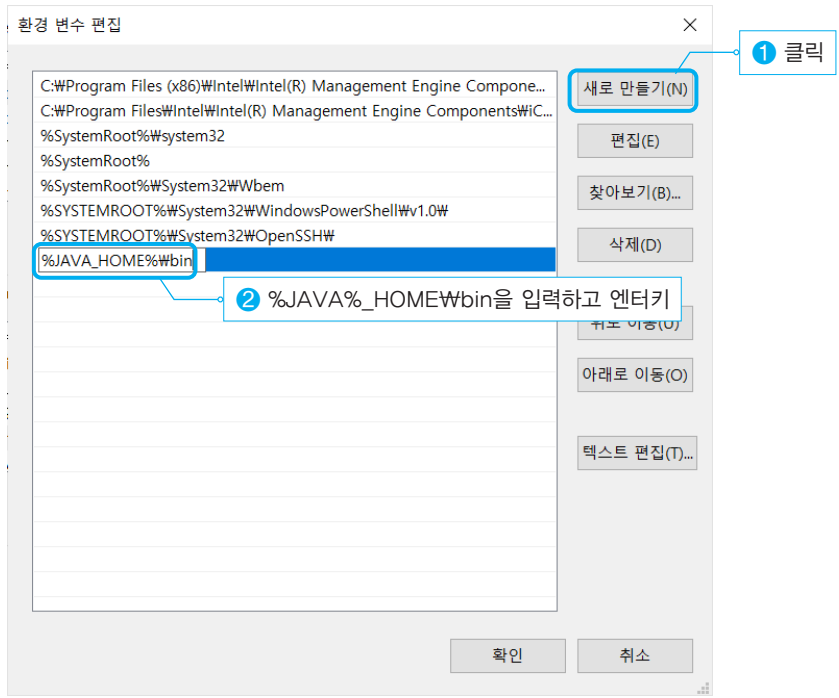
| 참고 | '변수 값'을 직접 입력하기 보다 '디렉터리 찾아보기(D)...'버튼을 눌러서 찾는 것이 확실하다.



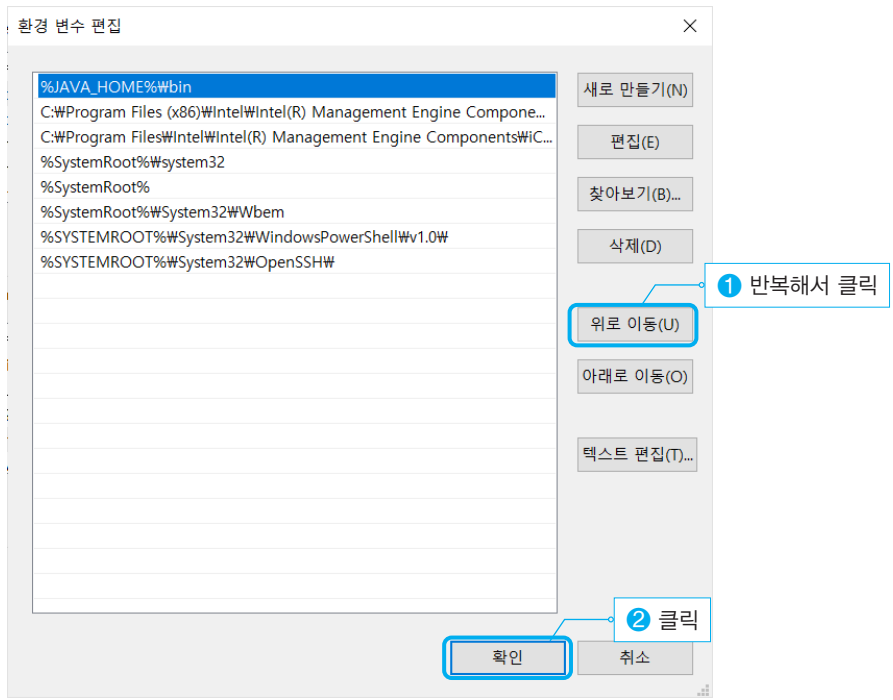
7 아래와 같은 화면이 나타나면, '시스템 변수'목록의 스크롤바를 아래로 내리면 'Path'항목을 찾을 수 있다. 이 항목을 클릭하고, '편집(I)...'버튼을 누르자.



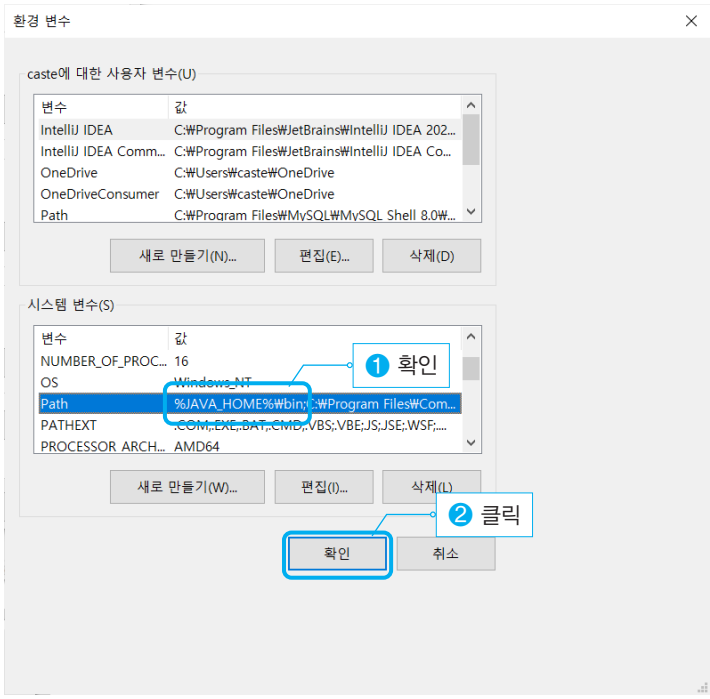
8 아래와 같이 새로운 화면이 열리면 우측 상단의 '새로 만들기(N)'버튼을 누르고, '%JAVA_HOME%₩bin'을 입력하고 Enter키를 누르자.



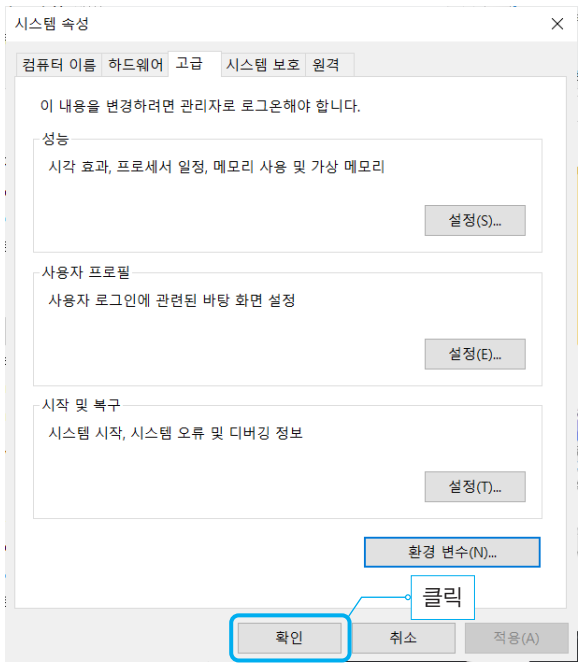
9 우측의 '위로 이동(U)'버튼을 여러번 눌러서 새로 추가한 경로가 맨 위로 올라가게 하고 '확인' 버튼을 눌러서 창을 닫는다.



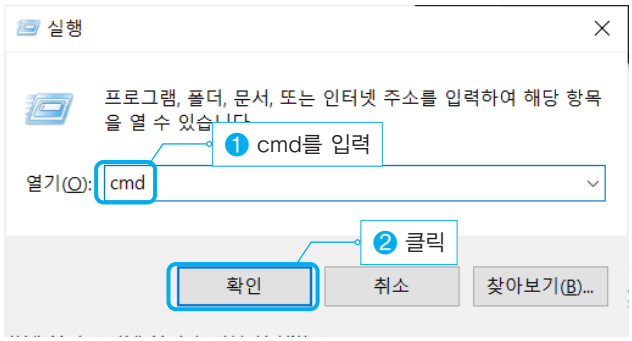
10 아래의 화면에서 전에 입력한 내용이 추가된 것을 확인하고, '확인'버튼을 클릭하자.



11 아래와 같은 화면에서 다시 '확인'버튼을 클릭하자.

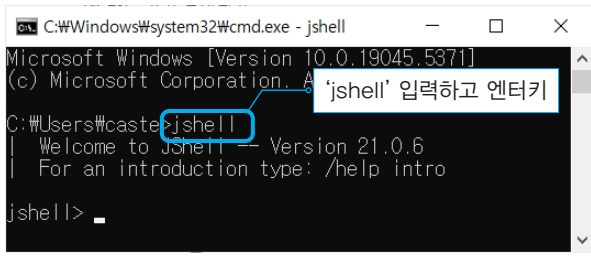


12 이제 설정은 다 끝났고, 설정이 잘되었는지 확인해 보자. '윈도우키+R'을 누르면 아래와 같은 화면이 나오는데, 'cmd'라고 입력하고 '확인'버튼을 누르자.



13 새로운 창이 열리면, 'jshell'이라고 입력하고 엔터키를 누르자. 아래와 같은 결과가 나오면 설정이 잘된 것이니 창을 닫으면 JDK의 설치와 설정이 모두 끝났다.

참고 | 만일 'jshell은 내부 또는 ... 아닙니다.'라는 메시지가 나오면 설정이 잘못된 것이며, 4~11의 과정을 다시 반복하자.

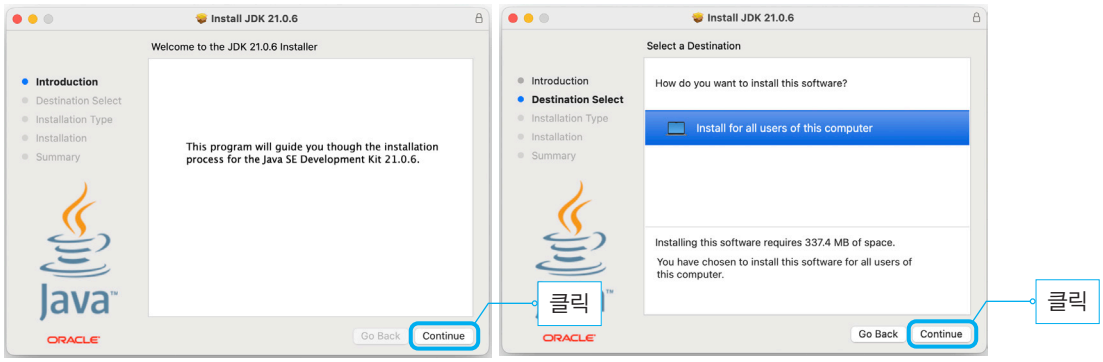


JDK 설치하기 – 맥OS

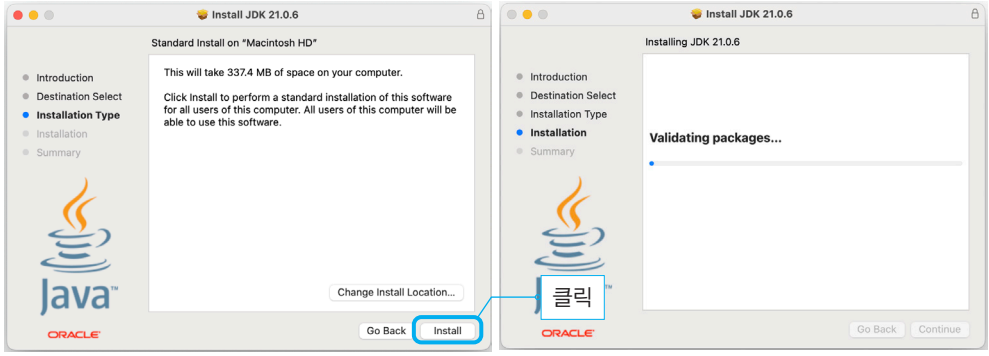
맥OS 사용자를 위한 JDK설치 방법에 대해서 알아보자.

1 앞에서 다운받은 jdk-21_macos-aarch64_bin.dmg파일을 실행하면 다음과 같은 화면이 나온다. 'JDK 21.0.6.pkg'를 더블 클릭하자.

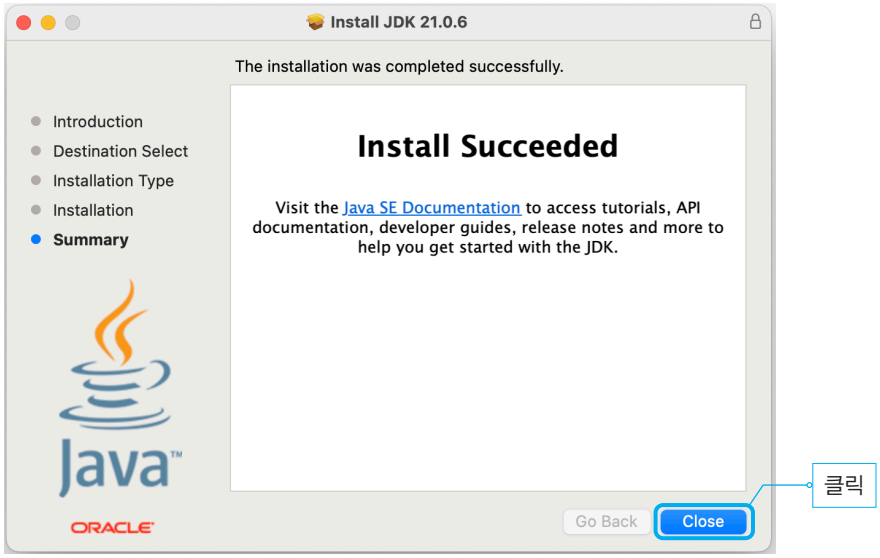
2 아래의 왼쪽 화면에서 'Continue'버튼을 클릭하면, 오른쪽 화면이 나오는데 다시 'Continue' 버튼을 클릭하자.



3 아래의 왼쪽 화면에서 'Install'버튼을 누르면 설치가 시작되면서 오른쪽과 같은 화면이 된다.



4 아래와 같은 화면이 나오면 설치가 잘 끝난 것이다. 'Close'버튼을 누르자.



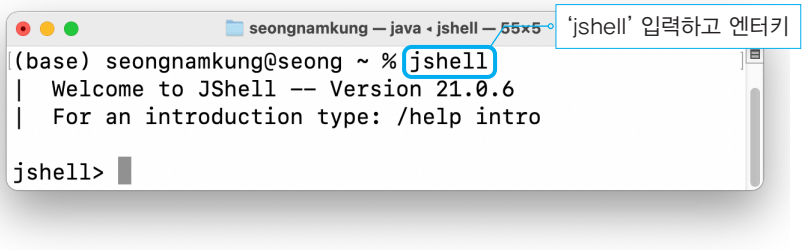
5 이제 설치가 끝났으니 설정을 할 차례이다. 사용자 디렉토리 아래의 '.bash_profile'파일을 열고 아래의 두줄을 마지막에 추가하고 저장하자.

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:${PATH}
```

6 마지막으로 터미널을 열고 아래의 명령을 입력하면 변경한 설정이 반영된다.

```
%source ~/.bash_profile
```

7 변경한 설정이 잘 반영되었는지 확인하기 위해 아래와 같이 'jshell'을 입력하고 엔터키를 누르면 아래와 같은 결과가 나오는지 확인하자.



2.2 인텔리제이(Intellij IDEA) 설치하기

앞으로 통합 개발 도구인 인텔리제이(IntelliJ IDEA)를 설치하고, 간단한 예제를 실행해볼 것이다. 설치 과정이 다소 변경될 수 있으므로 앞으로 설명하는 내용이 실제 화면과 다르면 저자의 깃헙 리포(https://github.com/castello/javajungsuk4)에서 '인텔리제이_설치방법.pdf'에서 최신 설치방법을 확인하자.

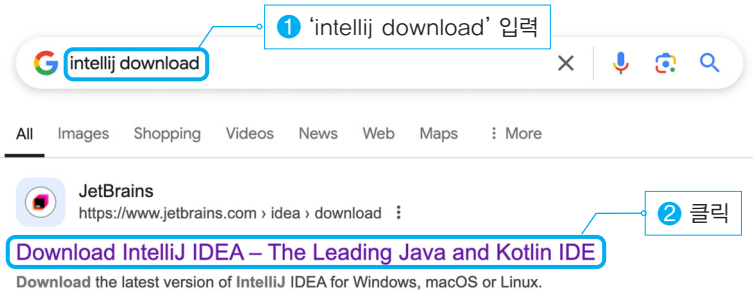
인텔리제이는 두가지 버전이 있는데, 무료 버전으로도 학습에 아무런 지장이 없기 때문에 무료 버전을 설치할 것이다.

- Intellij IDEA Ultimate – 유료, 30일 무료
- **Intellj IDEA Community Edition – 무료**

먼저 윈도우에 설치하는 방법을 설명하고, 그 다음에 MacOS에 설치하는 방법을 설명할 것이다. 본인의 OS에 맞게 설치하면 된다.

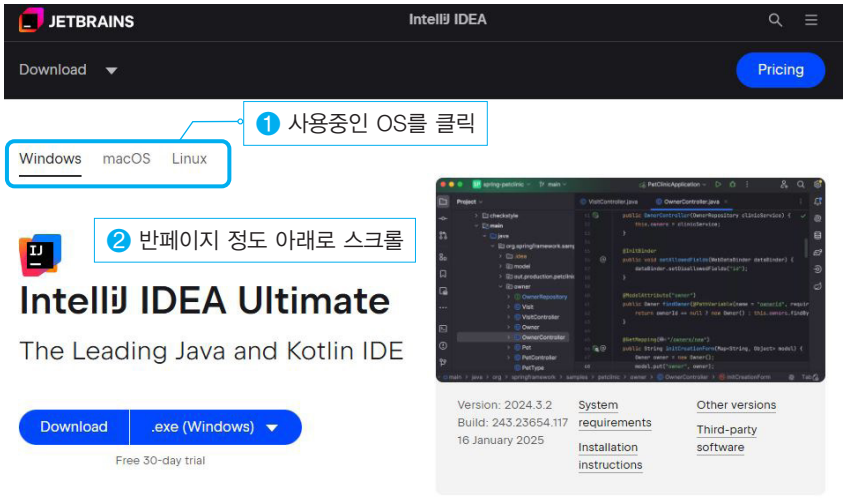
Intellij 다운로드하기

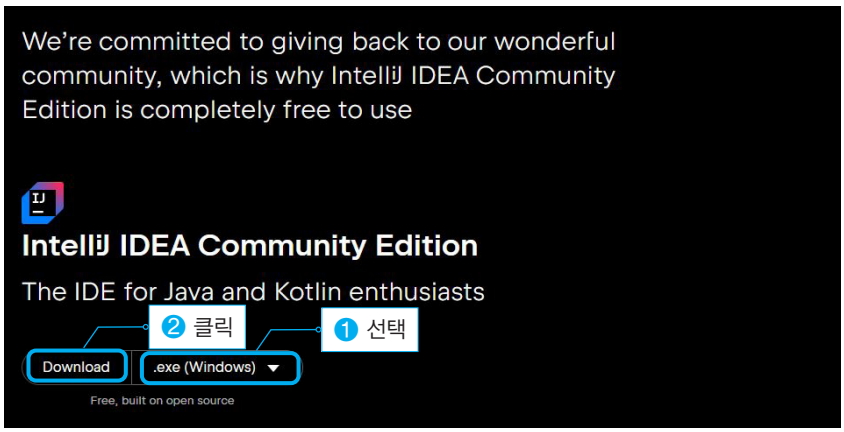
1 브라우저를 열고 intellij download라고 입력하여 검색한 후, 아래의 링크를 클릭



2 아래의 페이지가 나타나면 OS의 종류를 선택해서 클릭하고, 화면을 아래로 반 페이지 정도 스크롤하자. 두번째 그림처럼 Intellj IDEA Community Edition이 나오는데, 여기서 윈도우즈의 경우 .exe(Windows), 맥OS의 경우 .dmg(Apple Sillicon)을 선택하고, 'Download'버튼을 클릭하자.

| 참고 | 맥OS인데 Intel CPU를 사용하는 경우, .dmg(Intel)을 선택하자.





3 아래의 페이지가 나타나면서 다운로드가 자동으로 시작된다. 만일 자동으로 다운로드가 시작되지 않으면, 아래의 수동 다운로드 링크를 클릭하자.

이제 다운로드한 파일을 실행해서 인텔리제이를 설치해 보자. 맥OS의 설치가 간단하므로 먼저 살펴보고, 그 다음에 윈도우에 설치하는 방법을 설명할 것이다.

Intellij 설치하기 – 맥OS

1 전에 다운로드 받은 'idealC-2024.3.2.2-aarch64.dmg'를 더블 클릭하면 다음과 같은 화면이 나타난다. 왼쪽의 'Intellij IDEA CE' 아이콘을 드래그해서 'Applications' 아이콘 위로 겹쳐놓으면 된다. 설치는 이것으로 끝이다.

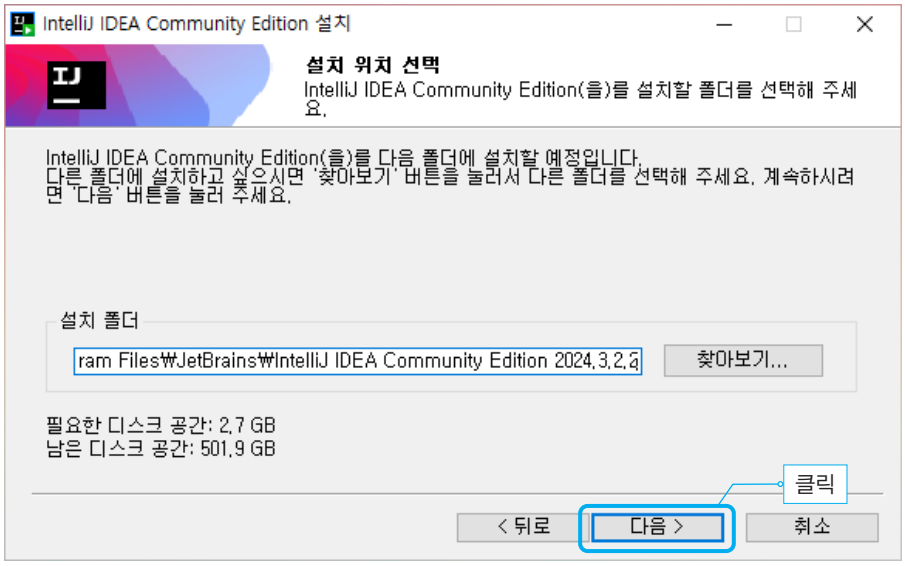
| 참고 | 다운받은 파일의 이름은 새로운 버전이 나오면 달라질 수 있다.

Intellij 설치하기 – 윈도우즈

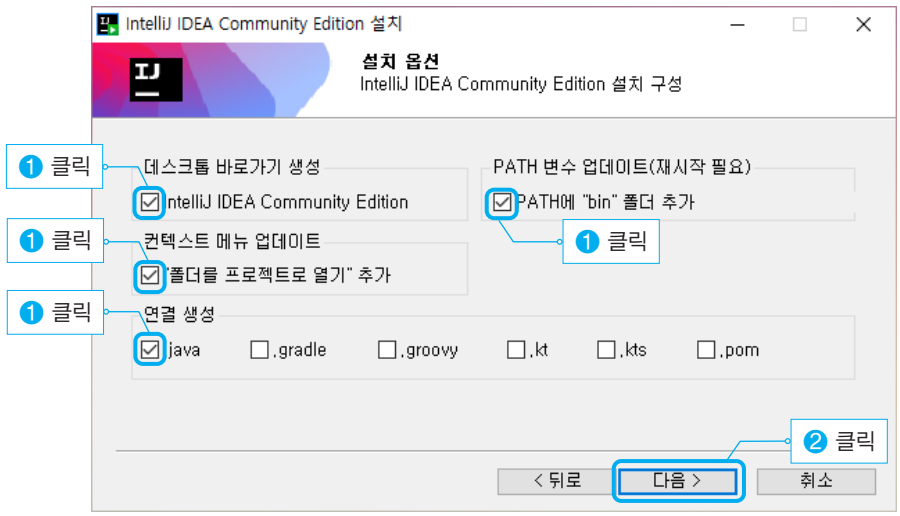
1 전에 다운로드 받은 'idealC-2024.3.2.2.exe'를 더블 클릭하여 실행하면 다음과 같은 화면이 나오는데, '다음 >'버튼을 클릭하자.

| 참고 | 다운받은 파일의 이름은 새로운 버전이 나오면 달라질 수 있다.

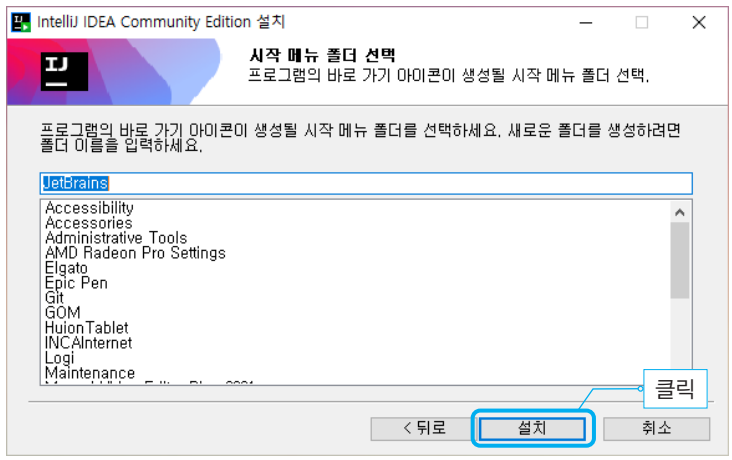
2 인텔리제이를 설치할 위치를 지정하는 화면이 나오는데, 원하는 곳으로 변경해도 되지만 특별한 이유가 없으면 기본을 지정된 위치에 설치하자. 그냥 '다음 >'버튼을 클릭하자.

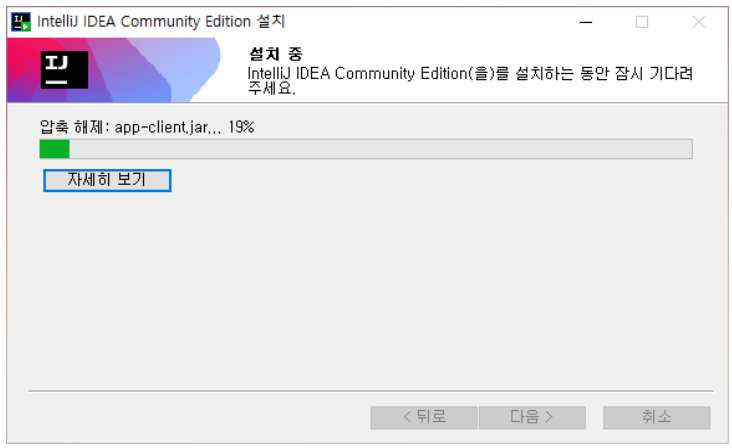


❸ 아래에 표시된 옵션들을 클릭해서 체크하고, '다음 〉'버튼을 누른다.

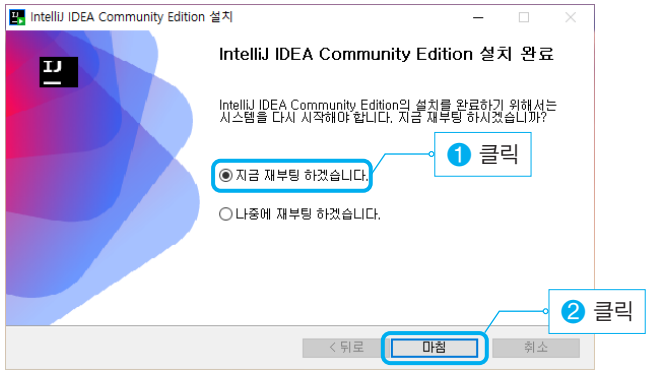


❹ 바로가기 아이콘이 생성될 시작 메뉴 폴더를 선택하는 화면이다. '설치'버튼을 클릭하면 설치가 시작된다.





5 설치가 끝나면 아래와 같은 화면이 나타난다. '지금 재부팅 하겠습니다.'클릭하고, '마침'버튼을 클릭하면 설치가 끝난다.



Intellij 실행하기 – 윈도우즈, 맥OS

1 바탕화면에 새로 생성된 아이콘을 클릭하면, 아래와 같은 화면이 나온다. 언어를 선택하고 '다음'버튼을 클릭하고, 그 다음의 사용자 계약 화면에서 체크하고 '계속'버튼을 클릭하자.

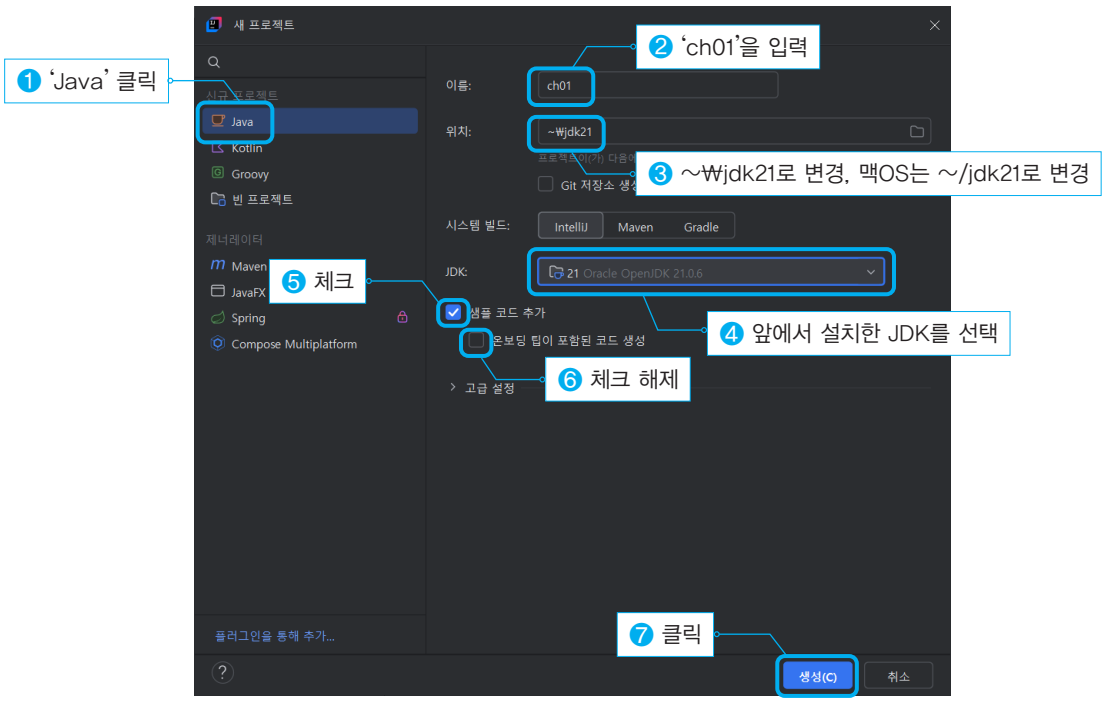
| 참고 | 경우에 따라 아래의 화면이 생략될 수도 있으며,, 언어와 지역은 나중에 변경할 수 있다.

2 아래의 화면에서 좌측의 '프로젝트'를 클릭하고, 중앙의 '새 프로젝트'를 클릭하자.

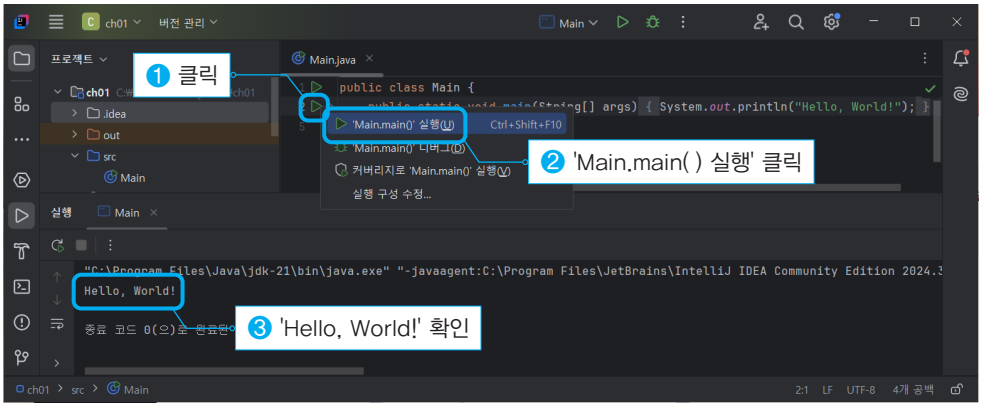
참고 아래의 화면이 나타나지 않으면, 메뉴에서 File 〉 New 〉 Project...를 클릭면 다음 단계의 화면이 나타난다.

참고 좌측의 '프로젝트' 아래의 '사용자 지정'을 클릭하면, 사용 언어와 테마 등의 설정을 변경할 수는 화면이 나온다.

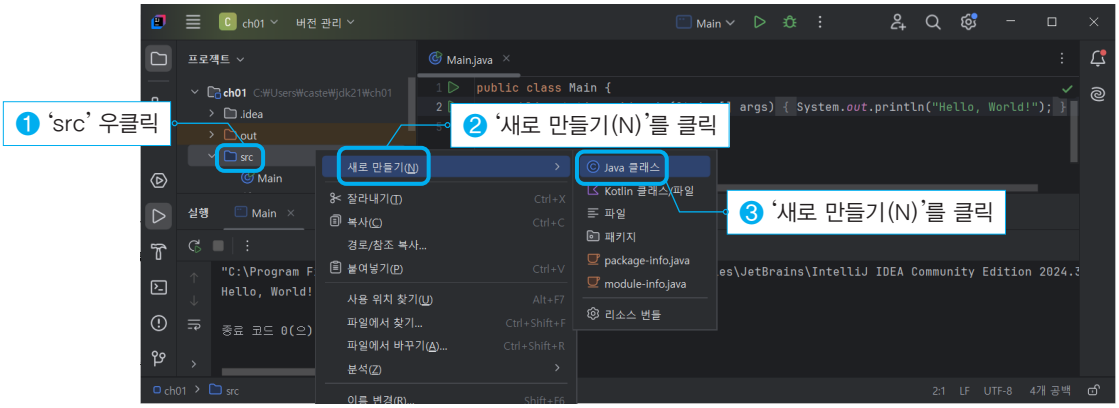
3 새로 생성할 프로젝트의 정보를 입력하는 화면이 나오는데, 프로젝트의 이름을 입력하고 경로를 변경하자. JDK는 전에 설치한 것이 자동으로 나타난다. 앞으로 챕터마다 이름만 다르게 새 프로젝트를 생성하자.



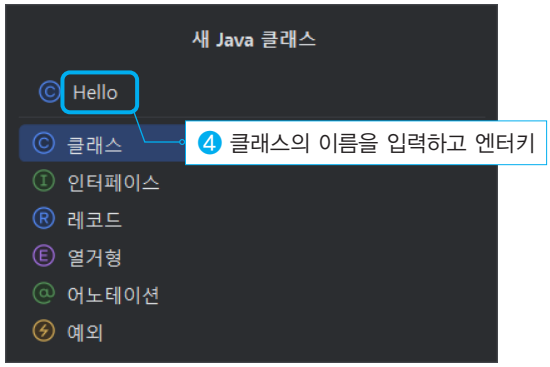
4 새로운 프로젝트가 생성되고 자동으로 Main.java라는 파일이 아래와 같이 생성된다. 녹색 삼각형을 클릭하면 프로그램이 실행되고 그 결과가 화면 하단의 콘솔에 'Hello, World!'가 출력된다. 이제 이걸로 인텔리제이의 설치와 설정이 모두 끝났다.



5 자바는 클래스 단위로 코드를 작성하며, 앞으로 새로운 예제를 작성할 때는 아래와 같이 새로운 클래스를 생성하고 코드를 작성하면 된다.



위의 화면에서 '새로 만들기(N)'을 클릭하면 아래와 같은 화면이 나오는데, 클래스 이름을 적고 엔터키를 누르면 새로운 파일이 생성된다. 생성된 파일에 예제의 내용대로 작성하고 이전 단계와 같이 녹색 삼각형을 눌르서 작성한 예제를 실행하면 된다.



3. 자바로 프로그램작성하기

3.1 Hello.java

자바로 프로그램을 개발하려면 JDK이외에도 편집기가 필요하다. 메모장과 같은 간단한 편집기도 있지만, 처음 자바를 배우는 사람들은 인텔리제이(IntelliJ IDEA)나 이클립스(eclipse)와 같이 다양하고 편리한 기능을 겸비한 고급 개발도구를 사용하는 것이 좋다.

이클립스에 비해 기능은 떨어지지만, 가볍고 간단한 편집기로 비쥬얼 스튜디오 코드(Visual Studio Code)라는 것도 있다.

| 참고 | 비쥬얼 스튜디오 코드는 https://code.visualstudio.com/ 에서 무료로 다운로드받을 수 있다.

▼ 예제 1-1/Hello.java

```
class Hello {
	public static void main(String[] args) {
		System.out.println("Hello, Java."); // 화면에 글자를 출력한다.
	}
}
```

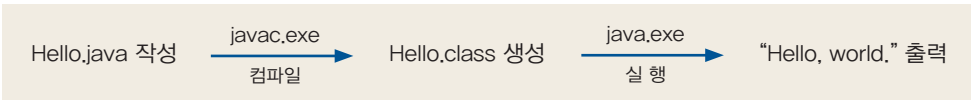
▼ 실행결과

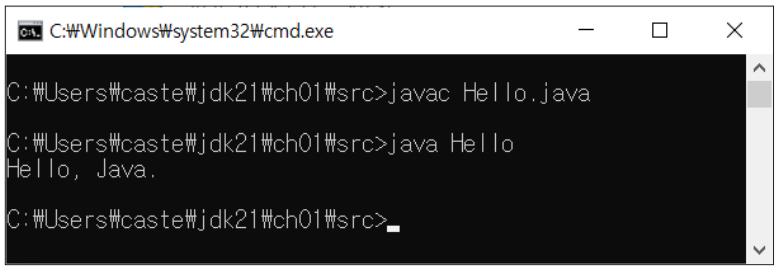
```
Hello, Java.
```

이 예제는 화면에 'Hello, Java.'를 출력하는 아주 간단한 프로그램이다. 이 예제를 통해서 화면에 글자를 출력하려면 어떻게 해야 하는지 쉽게 알 수 있을 것이다.

예제1-1을 편집기를 이용해서 작성한 다음 'Hello.java'로 저장하자. 이 때 클래스의 이름 'Hello'가 대소문자까지 정확히 같아야 한다.

이 예제를 실행하려면, 먼저 자바 컴파일러(javac.exe)를 사용해서 소스파일(Hello.java)로부터 클래스파일(Hello.class)을 생성해야 한다. 그 다음에 자바 인터프리터(java.exe)로 실행한다.





▲ 그림1-3 Hello.java를 컴파일하고 실행한 화면

그림1-3과 같은 결과를 얻었다면 자바로 프로그래밍할 준비가 모두 끝난 것이다. 만일 컴파일 시에 오류가 발생했다면 '3.2 자주 발생하는 에러와 해결방법'을 참고하자.

자바에서 모든 코드는 반드시 클래스 안에 존재해야 하며, 서로 관련된 코드들을 그룹으로 나누어 별도의 클래스를 구성하게 된다. 그리고 이 클래스들이 모여 하나의 Java 애플리케이션을 이룬다.

클래스를 작성하는 방법은 간단하다. 키워드 'class' 다음에 클래스의 이름을 적고, 클래스의 시작과 끝을 의미하는 괄호{} 안에 원하는 코드를 넣으면 된다.

```
class 클래스이름 {
    /*
        모든 코드는 클래스의 블럭{} 내에 작성해야한다. (주석 제외)
    */
}
```

| 참고 | 나중에 배우게 될 package문과 import문은 예외적으로 클래스의 밖에 작성한다.

아래 코드의 'public static void main(String[ ] args)'는 main메서드의 선언부인데, 프로그램을 실행할 때 'java.exe'에 의해 호출될 수 있도록 미리 약속된 부분이므로 항상 똑같이 적어주어야 한다.

| 참고 | '[ ]'은 배열을 의미하는 기호로 배열의 타입(type) 또는 배열의 이름 옆에 붙일 수 있다. 'String[ ] args'는 String타입의 배열 args를 선언한 것이며, 'String args[ ]'와 같이 쓸 수도 있다. 이 둘은 같은 의미이므로 차이가 없다. 자세한 내용은 '5장 배열'에서 배우게 될 것이다.

```
class 클래스이름 {
    public static void main(String[] args)  // main메서드의 선언부
    {
        // 실행될 문장들을 적는다.
    }
}
```

main메서드의 선언부 다음에 나오는 괄호{}는 메서드의 시작과 끝을 의미하며, 이 괄호 사이에 작업할 내용을 작성해 넣으면 된다. Java 애플리케이션은 main메서드의 호출로 시작해서 main메서드의 첫 문장부터 마지막 문장까지 수행을 마치면 종료된다.

모든 클래스가 main메서드를 가지고 있어야 하는 것은 아니지만, 하나의 Java 애플리케이션에는 main메서드를 포함한 클래스가 반드시 하나는 있어야 한다. main메서드는 Java애플리케이션의 시작점이므로 main메서드 없이는 Java 애플리케이션은 실행될 수 없기 때문이다. 작성된 Java애플리케이션을 실행할 때는 'java.exe' 다음에 main메서드를 포함한 클래스의 이름을 적어줘야 한다.

하나의 소스파일에 하나의 클래스만을 정의하는 것이 보통이지만, 하나의 소스파일에 둘 이상의 클래스를 정의하는 것도 가능하다. 이 때 주의해야할 점은 '소스파일의 이름은 public class의 이름과 일치해야 한다.'는 것이다. 만일 소스파일 내에 public class가 없다면, 소스파일의 이름은 소스파일 내의 어떤 클래스의 이름으로 해도 상관없다.

올바른 작성 예	설 명

Hello2.java
`public class Hello2 {}`
`class Hello3 {}`
	public class가 있는 경우, 소스파일의 이름은 반드시 public class의 이름과 일치해야한다.

Hello2.java
`class Hello2 {}`
`class Hello3 {}`
	public class가 하나도 없는 경우, 소스파일의 이름은 'Hello2.java', 'Hello3.java' 둘 다 가능하다.

잘못된 작성 예	설 명

Hello2.java
`public class Hello2 {}`
`public class Hello3 {}`
	하나의 소스파일에 둘 이상의 public class가 존재하면 안 된다. 각 클래스를 별도의 소스파일에 나눠서 저장하던가 아니면 둘 중의 한 클래스에 public을 붙이지 않아야 한다.

Hello3.java
`public class Hello2 {}`
`class Hello3 {}`
	소스파일의 이름이 public class의 이름과 일치하지 않는다. 소스파일의 이름을 'Hello2.java'로 변경해야 맞다.

hello2.java
`public class Hello2 {}`
`class Hello3 {}`
	소스파일의 이름과 public class의 이름이 일치하지 않는다. 대소문자를 구분하므로 대소문자까지 일치해야한다. 그래서, 소스파일의 이름에서 'h'를 'H'로 바꿔야 한다.

▲ 표1-2 소스파일의 작성 예

소스파일(*.java)과 달리 클래스파일(*.class)은 클래스마다 하나씩 만들어지므로 표1-2의 '올바른 작성 예'에 제시된 'Hello2.java'를 컴파일하면 'Hello2.class'와 'Hello3.class' 모두 두 개의 클래스파일이 생성된다.

접근 제어자(access modifier)인 'public'에 대해서는 '7장 객체지향 프로그래밍 II'에서 자세히 배울 것이므로 여기서는 하나의 소스파일에 둘 이상의 클래스를 정의할 때 주의할 점에 대해서만 이해하고 넘어가자.

3.2 자주 발생하는 에러와 해결방법

자바로 프로그래밍을 배워나가면서 많은 수의 크고 작은 에러들을 접하게 될 것이다. 대부분의 에러는 작은 실수에서 비롯된 것들이며, 곧 익숙해져서 쉽게 대응할 수 있게 되지만 처음 배울 때는 작은 실수 하나 때문에 많은 시간을 허비하곤 한다.

그래서 자주 발생하는 기본적인 에러와 해결방법을 간단히 정리하였다. 에러가 발생하였을 때 참고하고, 그 외의 에러는 에러메시지의 일부를 인터넷에서 검색해서 찾아보면 해결책을 얻는데 도움이 될 것이다.

1. cannot find symbol 또는 cannot resolve symbol

지정된 변수나 메서드를 찾을 수 없다는 뜻으로 선언되지 않은 변수나 메서드를 사용하거나, 변수 또는 메서드의 이름을 잘못 사용한 경우에 발생한다. 자바에서는 대소문자 구분을 하기 때문에 철자 뿐 만아니라 대소문자의 일치여부도 꼼꼼하게 확인해야한다.

2. ';' expected

세미콜론';'이 필요한 곳에 없다는 뜻이다. 자바의 모든 문장의 끝에는 ';'을 붙여주어야 하는데 가끔 이를 잊고 실수하기 쉽다.

3. Exception in thread "main" java.lang.NoSuchMethodError: main

'main메서드를 찾을 수 없다.'는 뜻인데 실제로 클래스 내에 main메서드가 존재하지 않거나 메서드의 선언부 'public static void main(String[ ] args)'에 오타가 존재하는 경우에 발생한다.

이 에러의 해결방법은 main메서드가 클래스에 정의되어 있는지 확인하고, 정의되어 있다면 main메서드의 선언부에 오타가 없는지 확인한다. 자바는 대소문자를 구별하므로 대소문자의 일치여부까지 정확히 확인해야한다.

| 참고 | args는 매개변수의 이름이므로 args 대신 argv나 arg와 같이 다른 이름을 사용할 수 있다.

4. Exception in thread "main" java.lang.NoClassDefFoundError: Hello

'Hello라는 클래스를 찾을 수 없다.'는 뜻이다. 클래스 'Hello'의 철자, 특히 대소문자를 확인해보고 이상이 없으면 클래스파일(*.class)이 생성되었는지 확인한다.

예를 들어 'Hello.java'가 정상적으로 컴파일 되었다면 클래스파일 'Hello.class'가 있어야한다. 클래스파일이 존재하는데도 동일한 메시지가 반복해서 나타난다면 클래스패스(classpath)의 설정이 바르게 되었는지 다시 확인해보자.

5. illegal start of expression

직역하면 문장(또는 수식, expression)의 앞부분이 문법에 맞지 않는다는 의미인데, 간단히 말해서 문장에 문법적 오류가 있다는 뜻이다. 괄호'(' 나 '{'를 열고서 닫지 않거나, 수식이나 if문, for문 등에 문법적 오류가 있을 때 또는 public이나 static과 같은 키워드를 잘못 사용한 경우에도 발생한다. 에러가 발생한 곳이 문법적으로 옳은지 확인하라.

6. class, interface, or enum expected

이 메시지의 의미는 '키워드 class나 interface 또는 enum이 없다.'이지만, 보통 괄호 '{' 또는 '}'의 개수가 일치 하지 않는 경우에 발생한다. 열린괄호'{'와 닫힌괄호'}'의 개수가 같은지 확인하자.

마지막으로 한 가지 더 얘기하고 싶은 것은 에러가 발생했을 때, 어떻게 해결할 것인가에 대한 방법이다. 아주 간단하고 당연한 내용이라서 다소 실망스럽게 느껴질지도 모르지만, 막상 실제 에러가 발생했을 때 아래의 순서대로 처리해보면 도움이 될 것이다.

1. 에러 메시지를 잘 읽고 해당 부분의 코드를 살펴본다.
   이상이 없으면 해당 코드의 주위(윗줄과 아래 줄)도 함께 살펴본다.
2. 그래도 이상이 없으면 에러 메시지는 잊어버리고 기본적인 부분을 재확인한다.
   대부분의 에러는 사소한 것인 경우가 많다.
3. 의심이 가는 부분을 주석처리하거나 따로 떼어내서 테스트 한다.

에러 메시지가 실제 에러와는 관계없는 내용일 때도 있지만, 대부분의 경우 에러 메시지만 잘 이해해도 문제가 해결되는 경우가 많으므로 **에러 해결을 위해서 제일 먼저 해야 할 일은 에러 메시지를 잘 읽는 것임을 명심**하자.

3.3 자바프로그램의 실행과정

콘솔에서 아래와 같이 Java 애플리케이션을 실행시켰을 때

```
C:\jdk21\ch01\src>java Hello
                      ↑
                      main(String[] args)
```

내부적인 진행순서는 다음과 같다.

1. 프로그램의 실행에 필요한 클래스(*.class파일)를 로드한다.
2. 클래스파일을 검사한다.(파일형식, 악성코드 체크)
3. 지정된 클래스(Hello)에서 main(String[ ] args)를 호출한다.

main메서드의 첫 줄부터 코드가 실행되기 시작하여 마지막 코드까지 모두 실행되면 프로그램이 종료되고, 프로그램에서 사용했던 자원들은 모두 반환된다.

만일 지정된 클래스에 main메서드가 없다면 다음과 같은 에러 메시지가 나타날 것이다.

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

3.4 주석(comment)

작성하는 프로그램의 크기가 커질수록 프로그램을 이해하고 변경하는 일이 점점 어려워진다. 심지어는 자신이 작성한 프로그램도 '내가 왜 이렇게 작성했지?'라는 의문이 들기도 하는데, 남이 작성한 코드를 이해한다는 것은 정말 쉬운 일이 아니다.

이러한 어려움을 덜기 위해 사용하는 것이 바로 주석이다. 주석을 이용해서 프로그램 코드에 대한 설명을 적절히 덧붙여 놓으면 프로그램을 이해하는 데 많은 도움이 된다.

그 외에도 주석은 프로그램의 작성자, 작성일시, 버전과 그에 따른 변경이력 등의 정보를 제공할 목적으로 사용된다.

주석을 작성하는 방법은 다음과 같이 두 가지 방법이 있다. '/*'와 '*/'사이에 주석을 넣는 방법과 앞에 '//'를 붙이는 방법이 있다.

> **범위 주석** '/*'와 '*/'사이의 내용은 주석으로 간주된다.
> **한 줄 주석** '//'부터 라인 끝까지의 내용은 주석으로 간주된다.

| 참고 | 이 외에도 Java API문서와 같은 형식의 문서를 자동으로 만들 수 있는 주석(/** ~ */)이 있지만 많이 사용되지는 않으므로 자세한 설명은 생략한다. 이 주석은 javadoc.exe에 의해서 html문서로 자동 변환되며, 보다 자세한 내용은 'javadoc'으로 검색하면 찾을 수 있다.

다음은 주석의 몇 가지 사용 예인데 흰색바탕으로 처리된 부분이 주석이다.

```
/*
Date   : 2025. 3. 1
Source : Hello.java
Author : 남궁성
Email  : seong.namkung@gmail.com
*/

class Hello
{
    public static void main(String[] args)  /* 프로그램의 시작 */
    {
        System.out.println("Hello, Java."); // Hello, Java를 출력
    }
}
```

위의 코드는 예제1-1에 주석을 넣은 것인데, 컴파일러는 주석을 무시하고 건너뛰기 때문에 위의 코드를 컴파일한 결과와 예제1-1을 컴파일한 결과는 정확히 일치한다. 따라서 주석이 많다고 해서 프로그램의 성능이 떨어지는 일은 없으니 안심하고 주석을 활용하기 바란다. **코드를 작성하기 전에 미리 주석으로 자신의 생각을 정리하고 검토하는 것은 좋은 습관이다.** 주석을 사용하지 말라는 주장도 있으나, 이 주장은 코드를 대충 작성하고 주석을 달지말고 주석이 필요없을 정도로 코드를 읽기 좋게 잘 작성하라는 뜻이다.

한 가지 주의해야할 점은 문자열을 의미하는 큰따옴표(") 안에 주석이 있을 때는 주석이 아닌 문자열로 인식된다는 것이다. Hello.java를 아래와 같이 변경하여 실행해보면, 주석의 내용도 같이 출력되는 것을 확인할 수 있을 것이다.

```
class Hello
{
	public static void main(String[] args)
	{
		System.out.println("Hello, /* 이것은 주석 아님 */ world.");
		System.out.println("Hello, world. // 이것도 주석 아님");
	}
}
```

3.5 이 책으로 공부하는 방법

2000년에 처음으로 자바강의를 시작했으니까 벌써 25년이 넘는 세월이 흘렀다. 그동안 많은 학생들을 가르치면서 어떻게 하면 더 쉽게 잘 배울 수 있을까에 대한 고민을 끊임없이 해왔고 그에 대한 결실로 책도 쓰게 되었다. 여전히 많은 학생들이 자바를 공부하는데 어려움을 겪고 있고, 공부 방법에 대해 고민하는 글들이 저자가 운영하는 카페에 많이 올라왔다. 카페 회원들과 소통하면서 처음 자바를 배우는 학생들이 어떤 점을 어려워하는지 잘 알게 되었고 그 고민에 대한 나름대로의 해법을 갖게 되었다. 카페에 자바를 공부하는 방법에 대한 글도 여러 번 쓰기도 했는데, 책을 구입하고도 카페에 가입하지 않는 독자들도 있기 때문에 그동안의 공부 방법에 대한 고민을 정리해서 책에 포함시키기로 했다.

누구나 자신만의 공부 방법이 있고, 절대적인 것은 없기 때문에 여기서 제시하는 공부 방법을 참고해서 자신에게 맞는 공부 방법을 완성하기 바란다.

이 책은 크게 3부분, 1장부터 5장까지, 6장부터 9장, 그리고 나머지 부분으로 나눌 수 있다. 처음 프로그래밍 언어를 배우는 사람은 2장부터 5장을 익숙해질 때까지 반복해서 봐야하고 실습도 많이 해야 한다. 눈으로만 이해하지 말고, 반드시 모든 예제를 직접 입력해보고 실행결과를 확인해 보자. **응용이 잘 안된다고 해서 앞부분에만 머물면 안 된다. 지금 단계에서 응용이 안 되는 것은 당연하다.** 기본적인 내용이 익숙해지면, 그 다음 단계인 6장으로 넘어가야 한다.

6장과 7장이 객체지향 개념의 핵심인데, 먼저 6장과 7장에 어떤 내용이 있는지 가볍게 한번 훑고 시작하자. 그 다음엔 6장을 여러 번 반복해서 보자. 6장을 이해하지 못하면 7장은 이해할 수 없기 때문이다. 7장은 좀 어려우므로 저자의 유튜브 채널(https://www.youtube.com/@MasterNKS)의 무료 동영상 강좌를 꼭 볼 것을 권한다.

반복해서 볼 때는 동영상을 1.5배속이나 2배속으로 보면 한 시간에 한번은 볼 수 있을 것이다. 하루에 2시간씩 5일보면, 10번은 볼 수 있다. 10번 봐도 이해가 안가면 10번 더 보자. 객체지향 개념을 이해하는데 30시간도 안 걸린다면 대성공이다.

객체지향 개념을 공부할 때 주의할 점은, 객체지향 개념 자체에 몰두하지 않아야 한다는 것이다. 이것은 완전히 샛길로 빠지는 것으로, 여러분들은 객체지향개념 언어인 '자바'를 배우는 것이지 객체지향 개념을 배우는 것이 아니라는 것을 잊지 말자.

6장과 7장은 반복하면 반복할수록 이해가 깊어지므로, 앞으로도 꾸준히 가볍게 복습하는 것이 좋다. 강의 내용을 요약해서 암기하자. 9장까지가 자바의 가장 기본적인 내용이므로 9장까지 마치고 나면, 2장부터 9장까지 전체적으로 한번 복습하면 좋다.

마지막으로 10장부터 16장까지는 자바의 응용부분이므로 앞부분을 충분히 이해하지 않고는 학습하기 어렵다. 이중에서 11장, 12장과 15장을 제외하고 나머지는 필요할 때 공부해도 좋다.

10장은 어떤 클래스들이 있는지 확인하고 필요할 때 책을 보고 사용할 수 있을 정도로만 공부하면 된다.

11장은 지금까지 배운 것들을 전부 활용하고 자료구조의 원리까지 들어가므로 책 전체에서 가장 어렵다. 처음엔 각 클래스의 특징과 사용법 정도만 확인하고 넘어가야 한다. 어떤 클래스들이 있고, 이 클래스들 통해서 어떤 결과를 얻을 수 있다는 정도면 충분하다. 처음부터 이장의 모든 내용을 이해하려고하면 어렵게만 느껴지고 진도도 안 나갈 것이다.

12장에서는 지네릭스가 중요한데, 예전에는 선택적으로 사용하던 기능이었지만 이제는 지네릭스를 모르고는 이해할 수 없는 코드가 많다. 지네릭스는 깊이 들어가면 어렵기 때문에 처음엔 기본적인 사용법만 익히고, 다른 장들을 공부하면서 막히는 부분을 다시 복습하는 식으로 공부하면 좋다. 애너테이션과 열거형은 경력자들의 요청으로 자세히 썼는데, 프로그래밍을 처음 배우는 사람은 기본적인 것만 이해하고 넘어가도 된다.

13장은 쓰레드에 대한 것인데, 일단 쓰레드가 어떤 것인지에 대한 감을 잡는 정도로만 공부하고, 나중에 필요할 때 자세히 보는 것이 좋다. 자바에서는 쉽게 멀티쓰레드를 구현할 수 있도록 미리 작성된 클래스들을 제공하고 있기 때문에 기본 개념만 알아도 도움이 많이 된다.

14장의 람다와 스트림은 11장, 12장과 관련이 많고 난이도가 높기 때문에 프로그래밍을 처음 배우는 사람들은 건너뛰었다가 필요할 때 추가로 학습해도 좋다.

15장은 입출력에 대한 것인데, 이 책의 후반부에서 꼭 학습해야하는 중요한 부분이다. 다른 장에 비해 실습이 재미있을 것이다.

16장은 컴퓨터간의 통신하는 방법에 대한 내용인데, 채팅 프로그램을 만드는 방법을 배운다. 15장과 관련이 있으므로 15장을 충분히 이해한 다음에 학습해야하며, 16장은 필수적으로 공부하지 않아도 되므로 건너뛰어도 좋다.

그 다음에는 안드로이드나 웹프로그래밍(JSP, Spring)을 공부하면서 하루에 한 챕터씩 꾸준히 복습해야 한다. 누구나 시간이 지나면 잊어버리기 때문에 계속 조금씩이라도 반복해서 봐야 실력이 쌓인다. 새로 배워야 할 것이 많다고 기본을 소홀히 하면 배우는 것보다 잃는 것이 더 많을 것이다. 하루에 10분이라도 반드시 시간을 내어 복습하자.

연습문제에 대하여

그동안 책이 두꺼워서 힘들다는 얘기를 많이 들었다. 그래서 고민 끝에 연습문제(약 200페이지)를 별도의 pdf파일로 제공하기로 결정했다. 연습문제는 저자의 깃헙 리포(https://github.com/castello/javajungsuk4)에서 제공하며, 상업적인 목적이 아니면 누구나 자유롭게 배포해도 된다.

하나의 챕터(chapter, 장)를 공부하고나면 연습문제를 꼭 풀어보자. 그러나 모든 문제를 다 풀어야 하는 것은 아니다. **풀 수 있는 문제들만 풀고, 풀 수 없는 문제들은 넘어갔다가 나중에 복습할 때 다시 풀어보는 것이 좋다.** 아니면 하루에 한 문제씩 틈틈이 고민하는 것도 좋다.

연습문제는 책에 있는 예제들을 응용한 것이 많기 때문에 연습문제를 풀면서 안 풀리는 문제는 해당 챕터를 뒤적거리면서 비슷한 예제가 없는지 찾아봐야 한다. 그러면서 공부가 되는 것이지, 문제만 붙들고 아무리 고민해봐야 문제도 안 풀리고 공부도 되지 않는다.

문제를 풀 때는 항상 종이에 낙서를 하는 방법을 추천한다. 어떻게 풀 것인가를 머리로만 고민하는 것보다 글과 그림으로 시각화하면 문제가 명확해지고 문제해결의 실마리를 쉽게 찾을 수 있기 때문이다.

아무리 고민해도 모르겠는 것은 연습문제 후반부에 있는 답안을 보자. 그래도 이해가 안 되면 카페에 질문을 올리면 저자가 직접 답변해 줄 것이다.

코드초보스터디카페 활용하기

네이버에 운영하고 있는 코드초보스터디는 개설한지 20년이 넘는 장수 카페이다. 회원수는 16만이 넘고 자바관련 카페 중에서 제일 오래되었다. 그동안 많은 스터디와 세미나를 해왔고, 자바 관련 질문이나 고민에 답변해주었다. 특히 본인이 집필한 책에 대한 질문은 거의 본인이 직접 답변해주었고, 다른 회원이 답변해준 내용은 바르게 답변했는지 확인한다.

이 책으로 공부하면서 막히는 것이 있으면, 카페에 질문을 하기 바란다. 가끔 지식인이나 다른 사이트에 이 책에 대한 질문을 하는 것을 볼 수 있는데, 가능하면 코드초보스터디에 질문해주었으면 한다. 다른 카페에 폐를 끼치는 것이기도 하고, 책의 저자로부터 직접 답변을 받을 수 있는데, 실력이 어떤지도 모르는 사람에게 답변을 받을 이유가 없기 때문이다.

카페에는 질문 답변 외에도 많은 자료가 있으므로 카페를 찬찬히 잘 둘러보길 권한다. '초보프로그래머에게'나, '면접후기', '자바소스강좌', 'Java1000제' 같은 게시판은 좋은 글들이 많다.

질문 올리는 방법

책과 관련된 질문은 저자가 빠짐없이 다 확인하고 답변하기 때문에, 답변을 못 받을 까봐 걱정하지 않아도 된다. 다만 질문하기 전에 유사한 질문이 없었는지 검색으로 확인하자.

책의 페이지로만 검색해도 이미 답변된 글들을 찾아서 볼 수 있으므로 답변해줄 때까지 기다리지 않아도 된다.

책과 관련되지 않은 답변은 카페에서 비교적 오래 활동해온 회원들이 해주는 경우가 많은데, 답변을 잘 받으려면, 읽는 사람 입장에서 생각하고 자신의 생각을 잘 정리해서 질문하면 된다. 수려한 문장에 맞춤법까지 완벽해야 좋은 질문이 아니고, 읽는 사람입장에서 답변하기 쉽게 하는 질문이 좋은 질문이다.

급한 마음은 알겠지만, 자신이 올린 질문을 다시 한 번 읽어보고 어떤 작업을 하는 도중에 어떤 문제가 발생했는지를 잘 정리해보자. 그러는 과정에서 스스로 문제가 해결되는 경우도 많다. 에러 메시지가 발생하는 경우는 꼭 같이 올리도록 하고, 소스를 포함시키되 소스가 길다면, 문제가 되는 부분만 따로 떼어서 테스트할 수 있게 올리는 것이 좋다.

앞으로 프로그래밍을 계속할거라면, 카페에서 뿐만 아니라 학교 선배나, 회사 선배, 직장 동료 등 많은 사람에게 질문을 하고 배워야 한다. 실력을 빠르게 향상시키려면, 질문을 잘하는 능력은 필수적이다.

마지막으로 **독자 여러분에게 하고 싶은 당부의 말은 '길을 잃지 말자'는 것과 '남과 비교하지 말자'**라는 것이다. 공부하다가 막힌다고 다른 책을 보고 수학공부하고 그러지 말라는 뜻이다. 공부하다 막히면 카페에 와서 질문을 하기 바란다. 저자 본인은 항상 여러 분의 질문을 20년 넘게 한결같이 같은 곳에서 기다리고 있다. 책을 읽다가 어려움이 있으면, 카페에 와서 질문을 하자. 간단한 것일지라도. 어렵다는 말만 반복하면서 질문 한번 안하는 회원들을 많이 봐왔는데, 질문을 부끄러워하지 않았으면 한다.

그리고, 사람마다 타고난 장점이 다르고 성장하는 속도가 다르다. 남들과 비교하지 말고 어제의 자신과 오늘의 자신을 비교하면서 한발 한발 나아가기 바란다.

Memo