

Java의 정석

4판 Java 21

남궁성지음

도우출판

Foreword

개정 4판을 펴내며...

첫 번째 자바 책을 출간한지 22년이 넘었네요. 이 오랜 시간동안 늘 같은 자리에서 독자 분들의 질문에 답변하며 유튜브에 무료 강의, 스터디, 세미나 등 많은 사람이 자바를 쉽게 잘 배울 수 있도록 노력해왔습니다. 저자가 본인 스스로 굳이 이런 글을 쓴다는 것이 내키지 않으나 인터넷에 독자를 현혹하는 정보가 많아서 이 자리를 빌어 저자의 허심탄회한 생각을 한번 읽어주길 바랄 뿐입니다.

세상의 변화만큼이나 자바도 빠르게 성장해왔는데 몇년이 지나도 이 변화를 온전히 담아주는 책이 나오지 않아서 제대로 된 책을 써보자 다짐하고 4판의 집필을 시작하게 되었습니다. 반년이면 되겠지 싶었으나 어느새 1년이 훌쩍 넘어서야 세상에 나오게 되었습니다.

금전적인 면만 생각하면 대충 구색만 갖춰서 새로운 책으로 출판할 수 있었겠지만, 적어도 누구 하나는 그러지 않아야 하니까요. 누군가 책을 잘 집필해주면 저도 이제 이 무거운 짐을 내려놓고 홀가분해질텐데 이점이 늘 아쉽습니다. 그래도 저는 오랫동안 독자 여러분의 사랑을 많이 받아왔으니 그 보답을 한다는 마음으로 아직까지 버티고 있습니다.

최근 AI의 급격한 발달로 지식을 보다 쉽게 얻을 수 있게 되었습니다. 누구나 AI를 사용하지만, 사람의 능력이 약간만 차이 나도 AI가 만들어내는 결과물의 격차가 큼니다. 그래서 사람이 습득해야 하는 지식의 양과 질이 더 중요해지고 있습니다.

실무에서는 점점 높은 수준의 실력을 요구하고 있으나 독자들은 점점 쉬운 책을 원하고 그에 맞는 책이 많이 팔리고 있습니다. 정말로 쉬운 것도 아니고 그저 쉬워보이는 책이나 강좌로는 실력이 늘기 어렵습니다. 그에 비해 이 책은 다소 어렵게 느껴질 수도 있으나 중요한 내용을 빠짐없이 담았고, 대신 저자가 직접 답변해주고 강의 영상도 무료로 제공하여 혼자서도 학습할 수 있도록 돕고 있습니다.

이를 잘 활용하면 보다 효율적으로 높은 실력을 갖추실 수 있습니다. 저는 독자 여러분을 항상 묵묵히 응원하며, 늘 같은 자리에서 여러분의 질문을 기다리고 있습니다.

2025년 6월 14일

저자 남궁성

seong.namkung@gmail.com

Contents

Chapter 01

자바를 시작하기 전에

1. 자바(Java programming language)	002
1.1 자바란?	002
1.2 자바의 역사	003
1.3 자바언어의 특징	006
1.4 JVM(Java Virtual Machine)	008
2. 자바개발환경 구축하기	010
2.1 자바 개발도구(JDK)설치하기	010
2.2 인텔리제이(IntelliJ IDEA) 설치하기	020
3. 자바로 프로그램 작성하기	028
3.1 Hello.java	028
3.2 자주 발생하는 에러와 해결방법	031
3.3 자바 프로그램의 실행과정	032
3.4 주석(comment)	033
3.5 이 책으로 공부하는 방법	034

Chapter 02

변수(variable)

1. 변수(variable)	040
1.1 변수(variable)란?	040
1.2 변수의 선언과 초기화	040
1.3 변수의 명명규칙	045
2. 변수의 타입	047
2.1 기본형(primitive type)	048
2.2 상수와 리터럴(constant & literal)	050
2.3 형식화된 출력 – printf()	058
2.4 화면에서 입력받기 – Scanner	062
3. 진법	064
3.1 10진법과 2진법	064
3.2 비트(bit)와 바이트(byte)	065
3.3 8진법과 16진법	066
3.4 정수의 진법 변환	068
3.5 실수의 진법 변환	070
3.6 음수의 2진 표현 – 2의 보수법	072
4. 기본형(primitive type)	076
4.1 논리형 – boolean	076
4.2 문자형 – char	076
4.3 정수형 – byte, short, int, long	083
4.4 실수형 – float, double	089
5. 형변환	095
5.1 형변환(캐스팅, casting)이란?	095
5.2 형변환 방법	095
5.3 정수형 간의 형변환	096
5.4 실수형 간의 형변환	098

5.5 정수형과 실수형 간의 형변환	101
5.6 자동 형변환	103

Chapter 03 연산자(operator)

1. 연산자(operator)	108
1.1 연산자와 피연산자	108
1.2 식(式)과 대입 연산자	108
1.3 연산자의 종류	108
1.4 연산자의 우선순위와 결합규칙	110
1.5 산술 변환(usual arithmetic conversion)	113
2. 단항 연산자	115
2.1 증감 연산자 ++, --	115
2.2 부호 연산자 +, -	118
3. 산술 연산자	119
3.1 사칙 연산자 +, -, *, /	119
3.2 나머지 연산자 %	130
4. 비교 연산자	131
4.1 대소비교 연산자 <, >, <=, >=	131
4.2 등가비교 연산자 ==, !=	131
5. 논리 연산자	136
5.1 논리 연산자 &&, , !	136
5.2 비트 연산자 &, , ^, ~, <<, >>	143
6. 그 외의 연산자	152
6.1 조건 연산자 ?:	152
6.2 대입 연산자 =, op=	154

Chapter 04 조건문과 반복문

1. 조건문 – if, switch	158
1.1 if문	158
1.2 if-else문	162
1.3 if-else if문	163
1.4 중첩 if문	166
1.5 switch문	168
2. 반복문 – for, while, do-while	180
2.1 for문	180
2.2 while문	191
2.3 do-while문	197
2.4 break문	199
2.5 continue문	200
2.6 이름 붙은 반복문	202

Contents

Chapter 05 배열(array)

1. 배열(array)	206
1.1 배열(array)이란?	206
1.2 배열의 선언과 생성	206
1.3 배열의 길이와 인덱스	208
1.4 배열의 초기화	213
1.5 배열의 복사	216
1.6 배열의 활용	220
2. String배열	230
2.1 String배열의 선언과 생성	230
2.2 String배열의 초기화	230
2.3 char배열과 String클래스	233
2.4 커맨드 라인을 통해 입력받기	236
3. 다차원 배열	238
3.1 2차원 배열의 선언과 인덱스	238
3.2 2차원 배열의 초기화	238
3.3 가변 배열	243
3.4 다차원 배열의 활용	244

Chapter 06 객체지향 프로그래밍 I

1. 객체지향언어	254
1.1 객체지향언어의 역사	254
1.2 객체지향언어	254
2. 클래스와 객체	255
2.1 클래스와 객체의 정의와 용도	255
2.2 객체와 인스턴스	256
2.3 객체의 구성요소 - 속성과 기능	257
2.4 인스턴스의 생성과 사용	258
2.5 객체 배열	264
2.6 클래스의 또 다른 정의	266
3. 변수와 메서드	270
3.1 선언위치에 따른 변수의 종류	270
3.2 클래스 변수와 인스턴스 변수	271
3.3 메서드	273
3.4 메서드의 선언과 구현	276
3.5 메서드의 호출	278
3.6 return문	282
3.7 JVM의 메모리 구조	285
3.8 기본형 매개변수와 참조형 매개변수	288
3.9 참조형 반환타입	292
3.10 재귀호출(recursive call)	294
3.11 클래스 메서드(static메서드)와 인스턴스 메서드	301
3.12 클래스 멤버와 인스턴스 멤버 간의 참조와 호출	304

4. 오버로딩(overloading)	307
4.1 오버로딩이란?	307
4.2 오버로딩의 조건	307
4.3 오버로딩의 예	307
4.4 오버로딩의 장점	309
4.5 가변인자(varargs)와 오버로딩	311
5. 생성자(constructor)	315
5.1 생성자란?	315
5.2 기본 생성자(default constructor)	316
5.3 매개변수가 있는 생성자	318
5.4 생성자에서 다른 생성자 호출하기 – this(), this	319
5.5 생성자를 이용한 인스턴스의 복사	322
6. 변수의 초기화	324
6.1 변수의 초기화	324
6.2 명시적 초기화(explicit initialization)	325
6.3 초기화 블록(initialization block)	326
6.4 멤버변수의 초기화 시기와 순서	328

Chapter 07

객체지향 프로그래밍 II

1. 상속(inheritance)	334
1.1 상속의 정의와 장점	334
1.2 클래스간의 관계 – 포함 관계	340
1.3 클래스간의 관계 결정하기	341
1.4 단일상속(single inheritance)	347
1.5 Object클래스 – 모든 클래스의 조상	349
2. 오버라이딩(overriding)	351
2.1 오버라이딩이란?	351
2.2 오버라이딩의 조건	352
2.3 오버로딩 vs. 오버라이딩	353
2.4 super	354
2.5 super() – 조상 클래스의 생성자	356
3. package와 import	360
3.1 패키지(package)	360
3.2 패키지의 선언	361
3.3 import문	364
3.4 import문의 선언	364
3.5 static import문	366
4. 제어자(modifier)	368
4.1 제어자란?	368
4.2 static – 클래스의, 공통적인	368
4.3 final – 마지막의, 변경될 수 없는	369
4.4 abstract – 추상의, 미완성의	371
4.5 접근 제어자(access modifier)	372
4.6 제어자(modifier)의 조합	377

Contents

5. 다형성(polymorphism)	378
5.1 다형성이란?	378
5.2 참조변수의 형변환	380
5.3 instanceof 연산자	386
5.4 참조변수와 인스턴스의 연결	394
5.5 매개변수의 다형성	397
5.6 여러 종류의 객체를 배열로 다루기	400
6. 추상 클래스(abstract class)	405
6.1 추상 클래스란?	405
6.2 추상 메서드(abstract method)	405
6.3 추상 클래스의 작성	407
7. 인터페이스(interface)	411
7.1 인터페이스란?	411
7.2 인터페이스의 작성	411
7.3 인터페이스의 상속	412
7.4 인터페이스의 구현	412
7.5 인터페이스를 이용한 다중 상속	415
7.6 인터페이스를 이용한 다형성	417
7.7 인터페이스의 장점	420
7.8 인터페이스의 이해	426
7.9 디폴트 메서드, static메서드, private메서드	430
8. 내부 클래스(inner class)	434
8.1 내부 클래스란?	434
8.2 내부 클래스의 종류와 특징	435
8.3 내부 클래스의 선언	435
8.4 내부 클래스의 제어자와 접근성	436
8.5 익명 클래스(anonymous class)	441

Chapter 08 예외 처리(exception handling)

1. 예외 처리(exception handling)	444
1.1 프로그램 오류	444
1.2 예외 클래스의 계층구조	445
1.3 예외 처리하기 - try-catch문	446
1.4 try-catch문에서의 흐름	449
1.5 예외의 발생과 catch블럭	450
1.6 예외 발생시키기	454
1.7 메서드에 예외 선언하기	457
1.8 finally블럭	464
1.9 자동 자원 반환 - try-with-resources문	466
1.10 사용자정의 예외 만들기	469
1.11 예외 되던지기(exception re-throwing)	472
1.12 연결된 예외(chained exception)	474

Chapter 09

java.lang패키지와 유용한 클래스

1. java.lang패키지	480
1.1 Object클래스	480
1.2 String클래스	494
1.3 StringBuffer와 StringBuilder	508
1.4 Math클래스	514
1.5 래퍼(wrapper) 클래스	521
2. 유용한 클래스	526
2.1 java.util.Objects	526
2.2 java.util.Random	530
2.3 정규식(regular expression) – java.util.regex	535
2.4 java.util.Scanner	540
2.5 java.util.StringTokenizer	543
2.6 java.math.BigInteger	548
2.7 java.math.BigDecimal	551

Chapter 10

날짜와 시간 & 형식화

1. 날짜와 시간	558
1.1 Calendar와 Date	558
2. 형식화 클래스	570
2.1 DecimalFormat	570
2.2 SimpleDateFormat	574
2.3 ChoiceFormat	578
2.4 MessageFormat	579
3. java.time패키지	582
3.1 java.time패키지의 핵심 클래스	582
3.2 LocalDate와 LocalTime	585
3.3 Instant	590
3.4 LocalDateTime과 ZonedDateTime	591
3.5 TemporalAdjusters	595
3.6 Period와 Duration	597
3.7 파싱과 포맷	602

Chapter 11

컬렉션 프레임워크

1. 컬렉션 프레임워크(collections framework)	608
1.1 컬렉션 프레임워크의 핵심 인터페이스	608
1.2 ArrayList	615
1.3 LinkedList	626
1.4 Stack과 Queue	634

Contents

1.5	Iterator, ListIterator, Enumeration	644
1.6	Arrays	654
1.7	Comparator와 Comparable	658
1.8	HashSet	661
1.9	TreeSet	668
1.10	HashMap과 Hashtable	674
1.11	TreeMap	684
1.12	Properties	688
1.13	Collections	694
1.14	컬렉션 클래스 정리 & 요약	699

Chapter 12

모던 자바 기능(new Java features)

1. 지네릭스(generics)	702
1.1 지네릭스란?	702
1.2 지네릭 클래스의 선언	703
1.3 지네릭 클래스의 객체 생성과 사용	706
1.4 제한된 지네릭 클래스	709
1.5 와일드 카드	711
1.6 지네릭 메서드	717
1.7 지네릭 타입의 형변환	720
1.8 지네릭 타입의 제거	722
2. 열거형(enums)	724
2.1 열거형이란?	724
2.2 열거형의 정의와 사용	725
2.3 열거형에 멤버 추가하기	728
2.4 열거형의 이해	731
3. 애너테이션(annotation)	735
3.1 애너테이션이란?	735
3.2 표준 애너테이션	736
3.3 메타 애너테이션	744
3.4 애너테이션 타입 정의하기	748
4. 레코드(record)	754
4.1 레코드란?	754
4.2 레코드의 특징	755
4.3 레코드의 중첩	760
4.4 지네릭 레코드	762
4.5 레코드와 애너테이션	764
5. 실드 클래스(sealed class)	766
5.1 실드 클래스란?	766
5.2 실드 클래스의 제약 조건	767
5.3 실드 클래스와 switch식	769
6. 모듈(module)	774
6.1 모듈이란?	774
6.2 모듈 설명자 - module-info.java	776

6.3	이름없는 모듈과 java.base모듈	779
6.4	전이적 의존성과 순환 의존성	786
6.5	모듈의 컴파일과 실행	788
6.6	자동 모듈	793

Chapter 13

쓰레드(thread)

1. 쓰레드	796
1.1 프로세스와 쓰레드?	796
1.2 쓰레드의 구현과 실행	798
1.3 start()와 run()	802
1.4 싱글쓰레드와 멀티쓰레드	806
1.5 쓰레드의 우선순위	812
1.6 쓰레드 그룹(thread group)	815
1.7 데몬 쓰레드(daemon thread)	818
1.8 쓰레드의 실행제어	822
2. 쓰레드의 동기화	841
2.1 synchronized를 이용한 동기화	841
2.2 wait()과 notify()	845
2.3 Lock과 Condition을 이용한 동기화	853
2.4 volatile	860
2.5 fork & join 프레임워크	862
3. 가상 쓰레드(virtual thread)	867
3.1 가상 쓰레드란?	867
3.2 가상 쓰레드의 생성과 사용	868
3.3 가상 쓰레드의 특징	869
3.4 플랫폼 쓰레드와 가상 쓰레드	871
3.5 가상 쓰레드의 상태	878
3.6 가상 쓰레드 작성시 주의사항	884
3.7 Continuation과 StackChunk	885
4. Executor와 ExecutorService	889
4.1 Executor	889
4.2 ThreadFactory	890
4.3 ExecutorService	892
4.4 쓰레드 풀(thread pool)	898
4.5 Future	903
4.6 CompletableFuture	913

Chapter 14

람다와 스트림

1. 람다식(lambda Expression)	928
1.1 람다식이란?	928
1.2 람다식 작성하기	929
1.3 함수형 인터페이스(Functional Interface)	931
1.4 java.util.function 패키지	936

Contents

1.5 Function의 합성과 Predicate의 결합	942
1.6 메서드 참조	946
2. 스트림(stream)	948
2.1 스트림이란?	948
2.2 스트림 만들기	953
2.3 스트림의 중간연산	958
2.4 Optional<T>와 OptionalInt	971
2.5 스트림의 최종연산	976
2.6 collect()	980
2.7 Collector구현하기	997
2.8 스트림의 변환	1000

Chapter 15

입출력(I/O)

1. 자바에서의 입출력	1004
1.1 입출력이란?	1004
1.2 스트림(stream)	1004
1.3 바이트 기반 스트림 – InputStream, OutputStream	1005
1.4 보조 스트림	1007
1.5 문자 기반 스트림 – Reader, Writer	1008
2. 바이트 기반 스트림	1010
2.1 InputStream과 OutputStream	1010
2.2 ByteArrayInputStream과 ByteArrayOutputStream	1012
2.3 FileInputStream과 FileOutputStream	1016
3. 바이트 기반의 보조 스트림	1019
3.1 FilterInputStream과 FilterOutputStream	1019
3.2 BufferedInputStream과 BufferedOutputStream	1020
3.3 DataInputStream과 DataOutputStream	1023
3.4 SequenceInputStream	1029
3.5 PrintStream	1031
4. 문자 기반 스트림	1035
4.1 Reader와 Writer	1035
4.2 FileReader와 FileWriter	1037
4.3 PipedReader와 PipedWriter	1039
4.4 StringReader와 StringWriter	1041
5. 문자 기반의 보조 스트림	1042
5.1 BufferedReader와 BufferedWriter	1042
5.2 InputStreamReader와 OutputStreamWriter	1043
6. 표준 입출력과 File	1045
6.1 표준 입출력 – System.in, System.out, System.err	1045
6.2 표준 입출력의 대상변경 – setIn(), setOut(), setErr()	1047
6.3 RandomAccessFile	1049
6.4 File	1053
7. 직렬화(serialization)	1072
7.1 직렬화란?	1072

7.2	ObjectInputStream과 ObjectOutputStream	1073
7.3	직렬화 가능한 클래스 만들기 – Serializable, transient	1075
7.4	직렬화 가능한 클래스의 버전관리	1081

Chapter 16

네트워킹(networking)

1. 네트워킹(networking)	1084
1.1 클라이언트/서버(client/server)	1084
1.2 IP주소(IP address)	1086
1.3 InetAddress	1087
1.4 URL과 URI	1089
1.5 URLConnection	1092
2. 소켓 프로그래밍	1097
2.1 TCP와 UDP	1097
2.2 TCP소켓 프로그래밍	1098
2.3 UDP소켓 프로그래밍	1116

Memo

Java

Programming Language

Chapter 01

자바를 시작하기 전에

getting started with Java

1. 자바(Java Programming Language)

1.1 자바란?

자바는 썬 마이크로시스템즈(Sun Microsystems, 이하 썬)에서 개발하여 1996년 1월에 공식적으로 발표한 객체지향 프로그래밍 언어이며, 추후에 함수형 프로그래밍 기능이 추가되었다. 전세계적으로 가장 많이 사용되는 프로그래밍 언어 중의 하나로 약 30년동안 꾸준히 발전해왔으며 쓰이지 않는 곳을 찾기가 힘들 정도로 폭넓은 분야에서 사용된다.

좁은 의미에서의 자바는 단순히 프로그래밍 언어지만, 넓은 의미에서 자바는 프로그래밍 언어 뿐만 아니라 관련된 여러 소프트웨어와 명세(specification)를 포함한다.

자바는 특정 플랫폼에 종속되지 않는 소프트웨어를 개발하고 배포하는데 필요한 모든 것을 제공한다. 그 덕분에 임베디드 시스템, 모바일 기기, 기업용 서버, 게임, 빅데이터, 인공지능에 이르기까지 아주 널리 쓰이고 있다.

자바는 SE(Standard Edition), ME(Micro Edition), EE(Enterprise Edition) 등 여러가지 종류가 있으며 대부분의 경우 자바는 'Java SE'를 의미한다.

참고 | Java EE는 2017년에 Eclipse재단으로 이전되면서 Jakarta EE로 이름이 변경되었다.



널리 사용되는 만큼 신중한 기능 개선으로 발전이 더디다는 평을 받아왔지만, 2017년 이후로 업데이트가 빨라져서 6개월마다 새로운 버전이 출시된다. 앞으로 최소한 몇년간은 자바는 기업 환경에서 가장 많이 사용되는 프로그래밍 언어의 지위를 내어주지 않을 것으로 보인다.

모던 프로그래밍 언어에서 가장 핵심적인 것은 객체지향 개념과 함수형 개념인데, 자바로 작성된 객체지향 개념과 관련된 좋은 자료가 많기 때문에 객체지향 개념을 배우기에는 자바만한 언어가 없다. 빅데이터에서 많이 쓰이는 함수형 프로그래밍 언어인 스칼라(Scala)도 자바에서 발전된 것으로 자바를 잘 배워두면 여러 언어로 쉽게 확장해 나갈 수 있다.

1.2 자바의 역사

자바의 역사는 1991년에 썬의 엔지니어들에 의해서 고안된 오크(Oak)라는 언어에서부터 시작되었다. 원래 목표는 가전제품에 탑재될 소프트웨어를 만드는 것이었는데, 객체지향 언어인 C++을 확장하려다가 부족함을 느껴서 C++의 단점을 보완한 새로운 언어인 Oak를 개발하기로 결정하였다.

인터넷이 등장하자 운영체제에 독립적인 Oak가 이에 적합하다고 판단하여 개발 방향을 인터넷에 맞게 바꾸면서 이름을 자바(Java)로 변경하고, 1996년 1월에 자바의 정식 버전을 발표했다. 그 당시만 해도 자바로 작성된 애플릿(Applet)은 웹페이지에 사운드와 애니메이션 등의 멀티미디어적인 요소들을 제공할 수 있는 유일한 방법이었기 때문에 많은 인기를 얻고 단 기간에 많은 사용자층을 확보할 수 있었다.

이후에 애플릿이 매크로 미디어사의 플래시(flash)에게 밀려서 자바의 인기가 줄어들었으나 1990년 말에 웹의 폭발적인 성장으로 자바의 인기가 다시 급상승하였다. 대규모의 서버 애플리케이션의 개발에 자바가 사용됨으로써 실무에서 탄탄한 입지를 확보하게 되었다. 그러다가 2008년에 모바일 운영체제인 안드로이드에 자바가 사용되면서 자바의 활용 분야가 더욱 확대되면서 자바가 쓰이지 않는 곳이 없다고 할 정도가 되었다.

썬에서 오라클로

썬은 자바와 MySQL데이터베이스등으로 오픈소스 발전에 큰 기여를 해왔으나, 2000년대 후반에 리눅스의 발전과 닷컴 버블의 붕괴로 주 수익원이던 하드웨어 판매의 급락과 수익이 낮은 오픈소스의 특성으로 경영란에 빠지게 되었다.

결국 데이터베이스의 1인자 오라클(Oracle)이 2010년에 썬을 인수함으로써 자바와 MySQL데이터베이스를 확보하게 되었다. 오라클이 오픈소스의 수익성을 강화하면서 그동안 무료로 사용해 왔던 자바의 유료화가 우려되었으나 기업이나 상업적 용도가 아니면 여전히 자바는 무료로 사용할 수 있다.

Oracle JDK와 OpenJDK

오라클은 자바 개발도구인 JDK(Java Development Kit)의 대부분을 Oracle OpenJDK라는 오픈소스로 jdk.java.net 사이트에서 공개하고 있으며, 이를 기반으로 여러 기업이 자신만의 JDK를 오픈소스로 개발해서 공개하고 있다.

참고 | 오픈소스인 OpenJDK는 조건없이 무료이고 소스가 모두 공개되어있다. <https://github.com/openjdk/jdk>

배포판	공급자	설명
Oracle OpenJDK	Oracle	오라클이 제공하는 OpenJDK의 공식 배포판. 6개월만 지원
Eclipse Temurin	Eclipse Adoptium	다양한 플랫폼을 지원하며 무료로 LTS 버전의 장기 지원
Amazon Corretto	Amazon	AWS 서비스와 통합에 적합하며 무료로 LTS 버전의 장기 지원
Azul Zulu	Azul Systems	다양한 플랫폼을 지원하며 무료로 LTS 버전의 장기 지원
Red Hat OpenJDK	Red Hat	대규모 기업 환경(RHEL)에서 안정성과 호환성이 뛰어남.

▲ 표1-1 OpenJDK의 종류

오픈소스인 Oracle OpenJDK와 달리, Oracle JDK는 오픈 소스가 아니며 상업용 목적이 아닌 개인 사용자에게만 무료이다. Oracle OpenJDK는 Oracle JDK가 제공하는 JMC(Java Mission Control)와 JFR(Java Flight Recorder) 등의 상용도구가 포함되지 않는다는 점을 제외하면 Oracle JDK와 거의 동일하다.

JDK의 장기 지원 정책 – LTS, Long Term Support

기존에는 JDK의 발표 후 3년까지 보안 업데이트와 버그 수정이 무료로 지원되었으나 JDK 8부터는 LTS로 지정된 특정 버전만 ‘장기 지원(최소 8년)’을 제공하고, LTS가 아닌 버전은 다음 버전이 나오는 6개월 후면 지원이 종료된다.

학습 목적인 경우는 LTS버전이 아니어도 상관없으나, 상용 제품이나 기업 환경에서는 반드시 LTS버전의 JDK를 선택해야 한다.

❗ 참고 ❗ Oracle JDK와 달리 Oracle OpenJDK는 LTS를 지원 안한다. LTS를 지원하는 OpenJDK가 필요하면 표1-1을 참고.

자바의 새로운 기능 – JDK 8 ~ 21

집필 시점인 2025년을 기준으로 JDK 21이 최신 LTS버전이고, 아래에 JDK 8부터 JDK 21까지 새로 추가된 주요 기능의 목록을 정리하였다. 자세한 내용은 목록에 적힌 책의 페이지를 참고하자.

먼저 가장 중요한 기능 몇가지를 요약하면, JDK 8에서 랴다식과 스트림 API가 추가되면서 함수형 프로그래밍이 가능해졌다는 것과 JDK 9에서 G1(Garbage First)이라는 이름의 가비지 컬렉터(Gabage Collector)가 새로 추가되어 자동으로 메모리를 관리해주는 기능이 획기적으로 개선되었다. 그리고 JDK 9부터 추가된 모듈 시스템 덕분에 JDK가 모듈 단위로 잘 정리되어 6개월마다 새로운 버전을 출시하며 빠르게 변화할 수 있게 되었다.

❗ 참고 ❗ JDK의 버전은 1.x 형식으로 표기하였으나 JDK 1.5부터 JDK 5와 같이 JDK x의 형식으로 변경되었다.

JDK 8 – 2014.3 LTS

- 함수형 프로그래밍 지원 – 랴다식 & 스트림 API
- 새로운 날짜 및 시간 API: java.time패키지
- Optional 클래스: NullPointerException 방지

JDK 9 – 2017.9

- 모듈 시스템(JEP 261, p.774)
- 간결한 문자열(compact strings, JEP 254 p.507)
- 인터페이스에 private method 허용(p.432)
- G1을 기본 가비지 컬렉터(GC)로 변경
- REPL(Read-Eval-Print-Loop) 도구로 jshell을 제공

JDK 10 – 2018.3

- 지역 변수 타입 추론(var, JEP 286, p.54)

JDK 11 – 2018.9 LTS

- HTTP Client API: 현대적인 비동기 HTTP 요청 지원
- String클래스에 메서드 추가: isBlank, lines, repeat 등 (p.498)
- var를 Lambda식에 사용 가능. (JEP 323, p.929)

JDK 14 – 2020.3

- switch식 (switch expressions, JEP 361, p.178)

JDK 15 – 2020.9

- 텍스트 블록 (JEP 378, p.56)

JDK 16 – 2021.3

- 레코드 (records, JEP 395, p.754)
- instanceof를 위한 패턴매칭 (JEP 394, p.390)

JDK 17 – 2021.9 LTS

- 실드 클래스 (sealed classes, JEP 409, p.766)

JDK 18 – 2022.3

- 간단한 웹서버 제공 (/bin/jwebserver, JEP 408)
- 기본 인코딩을 UTF-8로 변경 (JEP 400, p.507)

JDK 21 – 2023.9 LTS

- 가상 쓰레드 (JEP 444, p.867)
- switch문을 위한 패턴매칭 (JEP 441, p.769)
- 레코드 패턴 (JEP 440, p.759)
- Collections Framework에 Sequenced Collections 추가 (JEP 431, p.614)

위의 목록에서 프리뷰 기능 (preview features)은 제외하였다. 프리뷰 기능은 새로운 실험적 기능을 개발자의 피드백을 받기 위해 미리보기 형식으로 제공하는 것으로 향후 자바 버전에서 정식 기능이 되거나 폐기될 수 있다.

프리뷰 기능을 사용하려면 컴파일 할 때와 실행할 때와 실행할 때 아래와 같이 별도의 옵션을 지정해야 한다.

```
c:\jdk21\ch01>javac --enable-preview --release 21 PreviewTest.java
c:\jdk21\ch01>java --enable-preview PreviewTest
```

참고 버전별로 추가 및 변경된 사항에 대한 보다 자세한 내용은 아래의 링크에서 확인할 수 있다.

<https://docs.oracle.com/en/java/javase/21/language/java-language-changes-release.html>

자바 개선 제안 제도 – JEP, Java Enhancement Proposal

JDK 8부터 도입된 ‘자바 개선 제안 제도(JEP)’로, 자바에 새로운 기능의 추가나 기존 기능을 개선을 제안하는 공식 문서이다.

JEP마다 고유 번호가 있으며, OpenJDK 커뮤니티(openjdk.org)를 통해 논의, 승인, 구현 과정을 거친다. 차기 JDK버전에 어떤 JEP들을 포함시킬지 결정한다.

❗ 참고 ❗ <https://openjdk.org/jeps/0> 에서 JEP의 목록을 확인할 수 있다.

OpenJDK는 오픈소스 답게 OpenJDK 커뮤니티와 JEP를 통해 Java 플랫폼의 발전 방향을 개발자들과 함께 고민하고 결정하며, 개발자들에게 Java의 미래를 이해할 수 있게 한다.

❗ 참고 ❗ JEP와 유사한 JSR(Java Spec Request)도 있는데, JSR은 자바의 표준을 위한 제안으로 JEP보다 상위 개념으로 서로 연관되어 있다. 예를 들어 JSR 376은 JEP 261과 JEP 282에 의해 구현되었다.

1.3 자바언어의 특징

자바는 최근에 발표된 언어답게 기존의 다른 언어에는 없는 많은 장점들을 가지고 있다. 그 중 대표적인 몇 가지에 대해서 알아보도록 하자.

1. 운영체제에 독립적이다.

기존의 언어는 한 운영체제에 맞게 개발된 프로그램을 다른 종류의 운영체제에 적용하기 위해서는 많은 노력이 필요하였지만, 자바에서는 더 이상 그런 노력을 하지 않아도 된다. 이것은 일종의 에뮬레이터인 자바가상머신(JVM)을 통해서 가능한 것인데, 자바 응용프로그램은 운영체제나 하드웨어가 아닌 JVM하고만 통신하고 JVM이 자바 응용프로그램으로부터 전달받은 명령을 해당 운영체제가 이해할 수 있도록 변환하여 전달한다. 자바로 작성된 프로그램은 운영체제에 독립적이지만 JVM은 운영체제에 종속적이어서 썬에서는 여러 운영체제에 설치할 수 있는 서로 다른 버전의 JVM을 제공하고 있다.

그래서 자바로 작성된 프로그램은 운영체제와 하드웨어에 관계없이 실행 가능하며 이것을 ‘한번 작성하면, 어디서나 실행된다.(Write once, run anywhere)’고 표현하기도 한다.

2. 객체지향 언어이자 함수형 언어이다.

자바는 프로그래밍의 대세로 자리 잡은 객체지향 프로그래밍언어(object-oriented programming language) 중의 하나로 객체지향개념의 특징인 상속, 캡슐화, 다형성이 잘 적용된 순수한 객체지향언어라는 평가를 받고 있다. JDK 8부터 함수형 프로그래밍을 지원하고 있어서 최신 변화의 흐름에 맞춰 지속적으로 성장해 가고 있다.

3. 비교적 배우기 쉽다.

자바의 연산자와 기본구문은 C++에서, 객체지향관련 구문은 스몰토크(small talk)이라는 객체지향언어에서 가져왔다. 이 둘 언어의 장점은 취하면서 복잡하고 불필요한 부분은 과

감히 제거하여 단순화함으로서 쉽게 배울 수 있으며, 간결하고 이해하기 쉬운 코드를 작성할 수 있도록 하였다. 객체지향언어의 특징인 재사용성과 유지보수의 용이성 등의 많은 장점에도 불구하고 배우기가 어렵기 때문에 많은 사용자층을 확보하지 못했으나 자바의 간결하면서도 명료한 객체지향적 설계는 사용자들이 객체지향개념을 보다 쉽게 이해하고 활용할 수 있도록 하여 객체지향 프로그래밍의 저변확대에 크게 기여했다.

4. 자동 메모리 관리(Garbage Collection)

자바로 작성된 프로그램이 실행되면, 가비지 컬렉터(garbage collector)가 자동적으로 메모리를 관리해주기 때문에 프로그래머는 메모리를 따로 관리 하지 않아도 된다. 가비지 컬렉터가 없다면 프로그래머가 사용하지 않는 메모리를 체크하고 반환하는 일을 수동적으로 처리해야할 것이다. 자동으로 메모리를 관리한다는 것이 다소 비효율적인 면도 있지만, 프로그래머가 보다 프로그래밍에 집중할 수 있도록 도와준다.

❗참고❗ JDK 9부터 성능이 크게 향상된 G1이 기본 가비지 컬렉터가 되었다. 실행시 옵션으로 가비지 컬렉터를 변경가능

5. 네트워크와 분산처리를 지원한다.

인터넷과 대규모 분산환경을 염두에 둔 까닭인지 풍부하고 다양한 네트워크 프로그래밍 라이브러리(Java API)를 통해 비교적 짧은 시간에 네트워크 관련 프로그램을 쉽게 개발할 수 있도록 지원한다.

6. 멀티쓰레드를 지원한다.

일반적으로 멀티쓰레드(multi-thread)의 지원은 사용되는 운영체제에 따라 구현방법도 상이하며, 처리 방식도 다르다. 그러나 자바에서 개발되는 멀티쓰레드 프로그램은 시스템과는 관계없이 구현가능하며, 관련된 라이브러리(Java API)가 제공되므로 구현이 쉽다. 그리고 여러 쓰레드에 대한 스케줄링(scheduling)을 자바 인터프리터가 담당하게 된다.

❗참고❗ JDK 21부터 가상 쓰레드(virtual thread)가 추가되어 자바로 고성능, 고처리량의 서버를 만들 수 있게 되었다.

7. 동적 로딩(Dynamic Loading)을 지원한다.

보통 자바로 작성된 애플리케이션은 여러 개의 클래스로 구성되어 있다. 자바는 동적 로딩을 지원하기 때문에 실행 시에 모든 클래스가 로딩되지 않고 필요한 시점에 클래스를 로딩하여 사용할 수 있다는 장점이 있다. 그 외에도 일부 클래스가 변경되어도 전체 애플리케이션을 다시 컴파일하지 않아도 되며, 애플리케이션의 변경사항이 발생해도 비교적 적은 작업만으로도 처리할 수 있는 유연한 애플리케이션을 작성할 수 있다.

1.4 JVM(Java Virtual Machine)

JVM은 'Java virtual machine'을 줄인 것으로 직역하면 '자바를 실행하기 위한 가상 기계'라고 할 수 있다. 가상 기계라는 말이 좀 어색하겠지만 영어권에서는 컴퓨터를 머신(machine)이라고도 부르기 때문에 '머신'이라는 용어대신 '컴퓨터'를 사용해서 '자바를 실행하기 위한 가상 컴퓨터'라고 이해하면 좋을 것이다.

'가상 기계(virtual machine)'는 소프트웨어로 구현된 하드웨어를 뜻하는 넓은 의미의 용어이며, 컴퓨터의 성능이 향상됨에 따라 점점 더 많은 하드웨어들이 소프트웨어화되어 컴퓨터 속으로 들어오고 있다. 그 예로는 TV와 비디오를 소프트웨어화한 윈도우 미디어 플레이어라던가, 오디오 시스템을 소프트웨어화한 윈앰프(winamp) 등이 있다.

이와 마찬가지로 '가상 컴퓨터(virtual computer)'는 실제 컴퓨터(하드웨어)가 아닌 소프트웨어로 구현된 컴퓨터라는 뜻으로 컴퓨터 속의 컴퓨터라고 생각하면 된다.

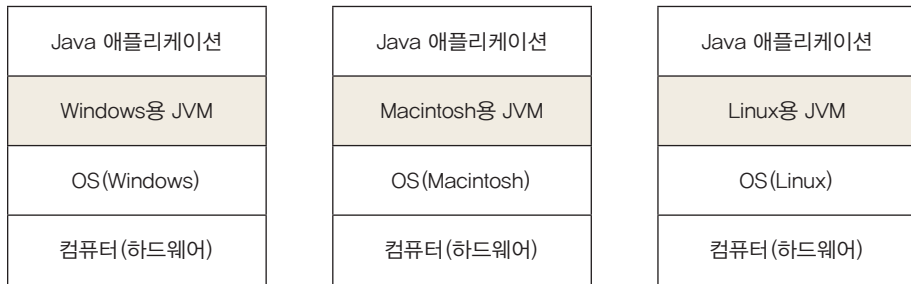
자바로 작성된 애플리케이션은 모두 이 가상 컴퓨터(JVM)에서만 실행되기 때문에, 자바 애플리케이션이 실행되기 위해서는 반드시 JVM이 필요하다.



▲ 그림 1-1 Java애플리케이션과 일반 애플리케이션의 비교

일반 애플리케이션의 코드는 OS만 거치고 하드웨어로 전달되는데 비해 Java애플리케이션은 JVM을 한 번 더 거치기 때문에, 그리고 하드웨어에 맞게 완전히 컴파일된 상태가 아니고 실행 시에 해석(interpret)되기 때문에 속도가 느리다는 단점을 가지고 있다. 그러나 요즘엔 바이트코드(컴파일된 자바코드)를 하드웨어의 기계어로 바로 변환해주는 JIT컴파일러와 향상된 최적화 기술이 적용되어서 속도의 격차를 많이 줄였다.

그림1-1에서 볼 수 있듯이 일반 애플리케이션은 OS와 바로 맞붙어 있기 때문에 OS종속적이다. 그래서 다른 OS에서 실행시키기 위해서는 애플리케이션을 그 OS에 맞게 변경해야 한다. 반면에 Java 애플리케이션은 JVM하고만 상호작용을 하기 때문에 OS와 하드웨어에 독립적이라 다른 OS에서도 프로그램의 변경없이 실행이 가능한 것이다. 단, JVM은 OS에 종속적이기 때문에 해당 OS에서 실행가능한 JVM이 필요하다.



▲ 그림 1-2 다양한 OS용 JVM

그래서 오라클에서는 일반적으로 많이 사용되는 주요 OS용 JVM을 제공하고 있고, 이렇게 함으로써 자바의 중요한 장점 중의 하나인 “Write once, run anywhere.(한 번 작성하면 어디서든 실행된다.)”이 가능하게 되는 것이다.

그랄 VM – Graal Virtual Machine

그랄 VM은 고성능, 다중언어 실행환경으로 기존의 핫스팟 VM의 JIT 컴파일러를 그랄 컴파일러로 대체한 가상 머신(Virtual Machine)이다. 자바 외에도 다양한 프로그래밍 언어(JavaScript, python, Ruby, R 등)를 지원하고 심지어는 여러 언어를 혼합해서 코드를 작성하는 것도 가능한데, 그 이유는 그랄VM은 각 언어의 소스 코드를 인터프리터가 그랄 VM이 이해할 수 있는 중립적인 표현으로 변환해서 처리하기 때문이다.

프로그래밍 언어마다 런타임 환경 성능이 제각각이라 어떤 언어는 성능이 상대적으로 많이 떨어지기도 하는데, 그랄VM은 입력된 중간 표현을 자동으로 최적화하고 런타임에 JIT컴파일까지 해주기 때문에 때로는 네이티브 컴파일러보다 실행 성능이 나을 수 있다.

특히 서버리스 및 클라우드 환경에 맞게 애플리케이션을 최적화하는 네이티브 바이너리(native image)로 변환하는 기능을 제공하는 것이 큰 장점이다.

오라클은 2018년 5월에 그랄 VM을 처음으로 공식 발표하였으나, 보다 효율적인 개발을 위해 JDK 16부터 JDK에서 독립시켜서 별도의 JDK로 개발하고 있다. 머지않아 Oracle JDK의 기본 VM으로 포함될 것이다.

❗ 참고 ❗ 그랄VM을 사용하려면 그랄VM이 포함된 JDK를 설치해야한다. <https://www.graalvm.org/downloads>

2. 자바개발환경 구축하기

2.1 자바 개발도구(JDK)설치하기

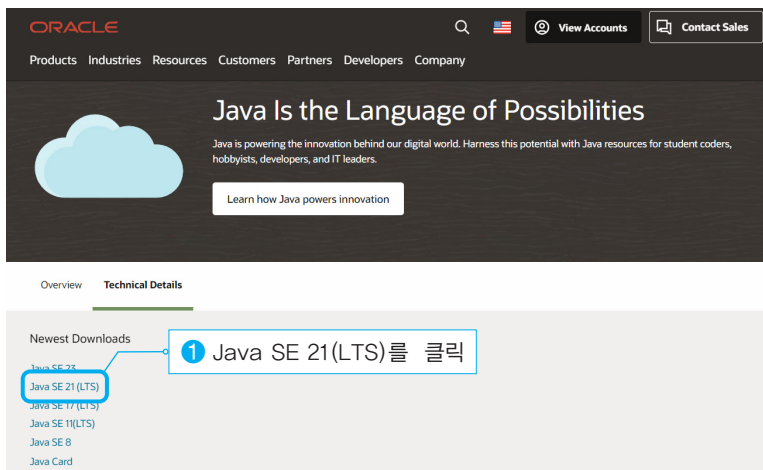
자바로 프로그래밍을 하기 위해서는 먼저 JDK(Java Development Kit)를 설치해야 한다. JDK를 설치하면, 자바가상머신(Java Virtual Machine, JVM)과 자바클래스 라이브러리(Java API)외에 자바를 개발하는데 필요한 프로그램들이 설치된다.

JDK 다운로드 받기

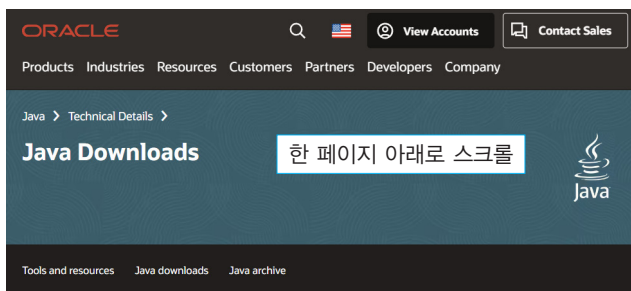
이 책을 학습하기 위해서는 JDK 21 이상의 버전이 필요하며, <http://java.sun.com/>에서 다운로드 받아서 설치할 것이다. JDK를 설치하는 것만으로는 자바를 학습하기에 불편하기 때문에, 보다 편리한 개발환경을 제공하는 통합 개발 환경(IDE)가 필요하다. 본인에게 익숙한 것을 사용해도 되지만 가능하면 이 책을 학습하는 동안 인텔리제이(IntelliJ IDEA)을 추천한다. 인텔리제이를 설치하는 방법은 JDK를 설치한 다음에 설명할 것이다.

❗ 참고 ❗ JDK를 설치하는 방법이 변경된 경우 <https://github.com/castello/javajungsuk4> 에서 JDK21_설치방법.pdf를 확인

1 브라우저를 열고 <http://java.sun.com>을 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.



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

Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use beyond the [limited free grants](#) of the OTN license will require a fee.

Linux	macOS	Windows
		1 클릭
Product/file description	File size	Download
x64 Compressed Archive	185.92 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	164.31 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.06 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

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.

Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use beyond the [limited free grants](#) of the OTN license will require a fee.

Linux	macOS	Windows
		1 클릭
Product/file description	File size	Download
ARM64 Compressed Archive	182.01 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-aarch64_bin.tar.gz (sha256)
ARM64 DMG Installer	181.32 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-aarch64_bin.dmg (sha256)
x64 Compressed Archive	184.25 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-x64_bin.tar.gz (sha256)
x64 DMG Installer	183.57 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-x64_bin.dmg (sha256)

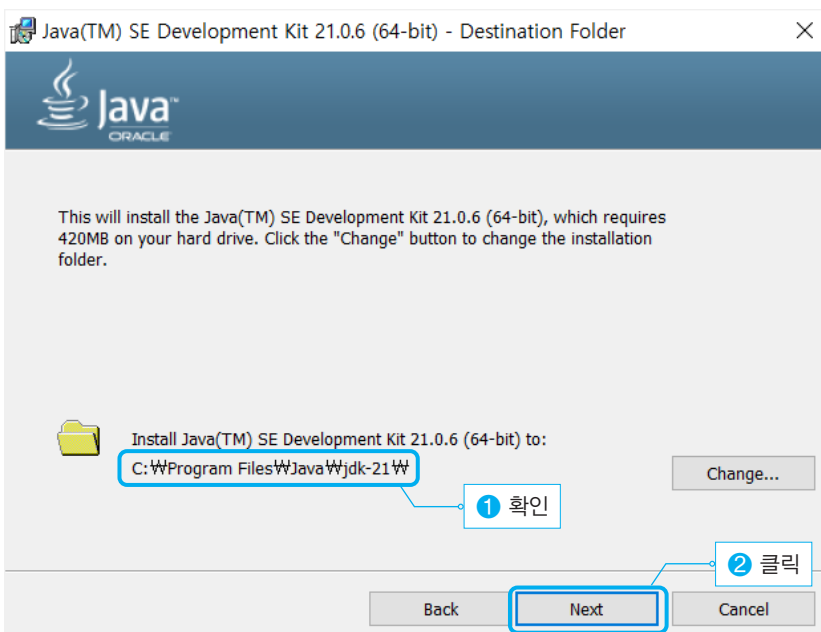
JDK 설치하기 – 윈도우즈

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, Next 버튼을 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.

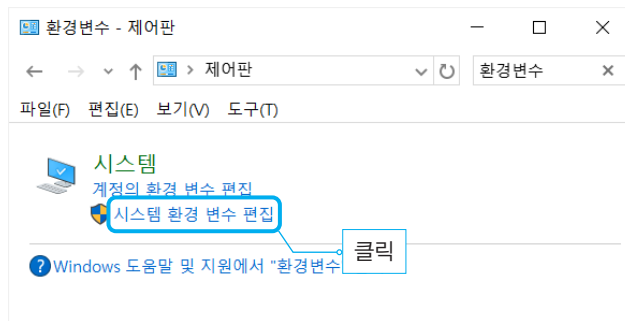


4 설치가 끝났으나 한가지 설정이 남았다. 제어판을 열고, 검색창에 '환경변수'라고 입력하자.

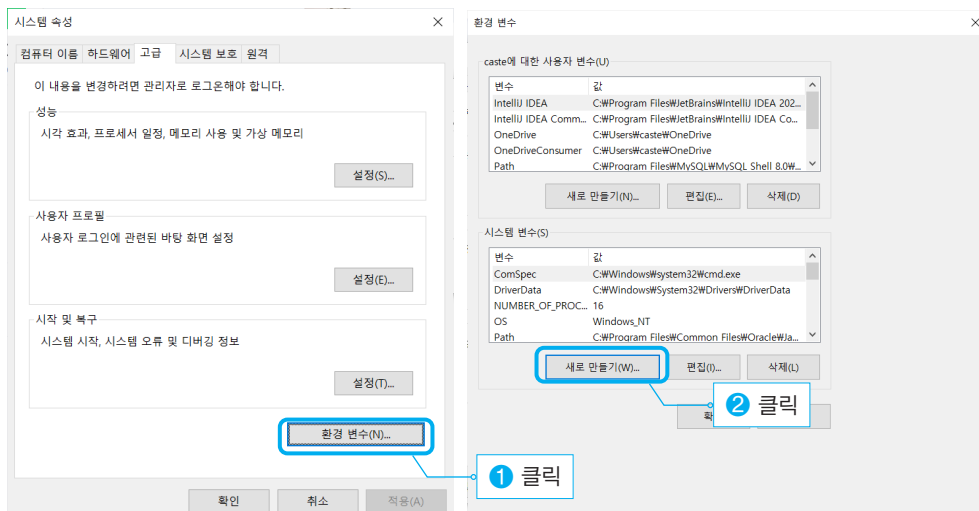
❗ 참고 ❗ 제어판은 윈도우키를 누르고, 찾기에 '제어판'이라고 입력하면 찾을 수 있다.



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

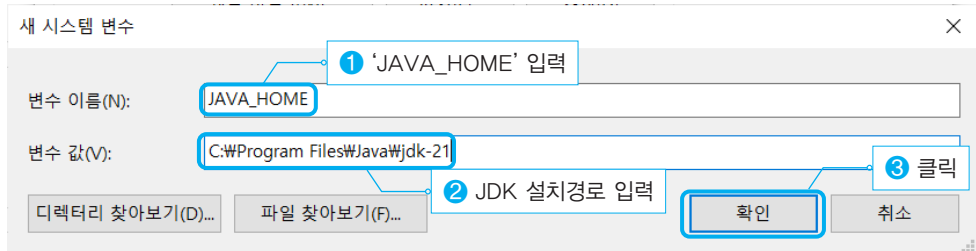


5 아래 왼쪽의 화면에서 '환경 변수(N)...' 버튼을 클릭하면 오른쪽과 같은 '환경 변수' 화면이 나타나는데 여기서 '새로 만들기(W)...' 버튼을 누르자.

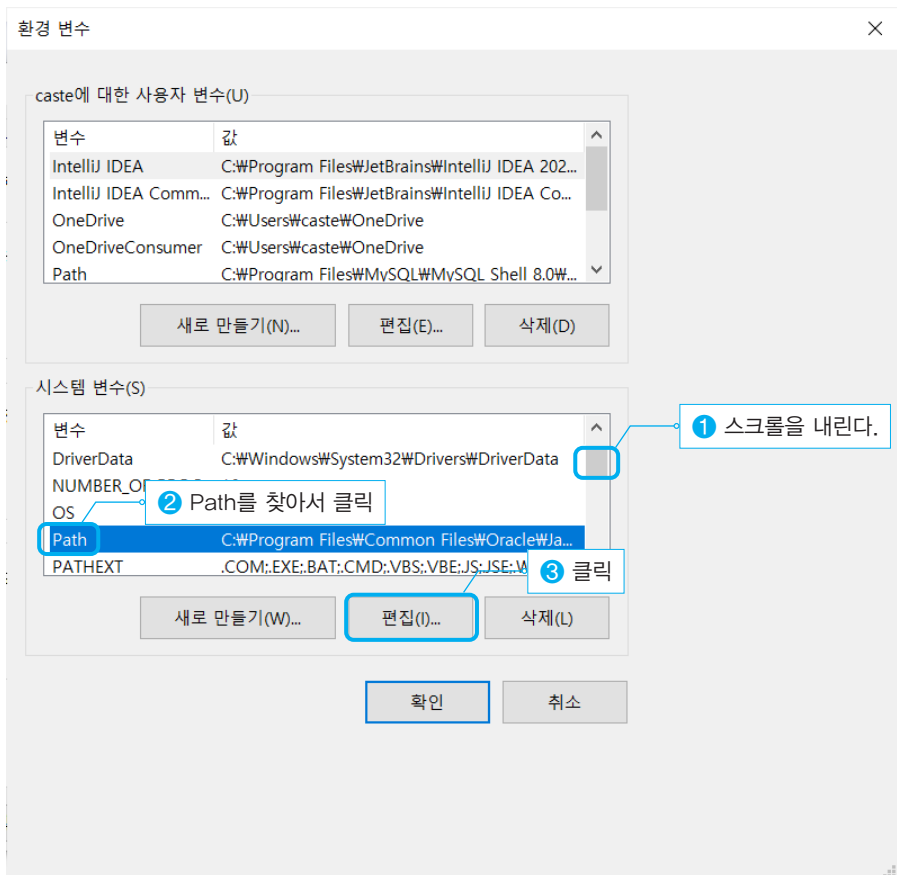


6 아래와 같은 화면이 나타나면, '변수 이름'으로 'JAVA_HOME'을 입력하고 '변수 값'에는 아까 JDK를 설치한 경로인 'C:\Program Files\Java\jdk-21'을 입력하자. 만일 JDK를 다른 곳에 설치했으면, 설치한 경로를 입력해야 한다. 입력한 내용을 다시 한번 확인하고 '확인'을 누르자.

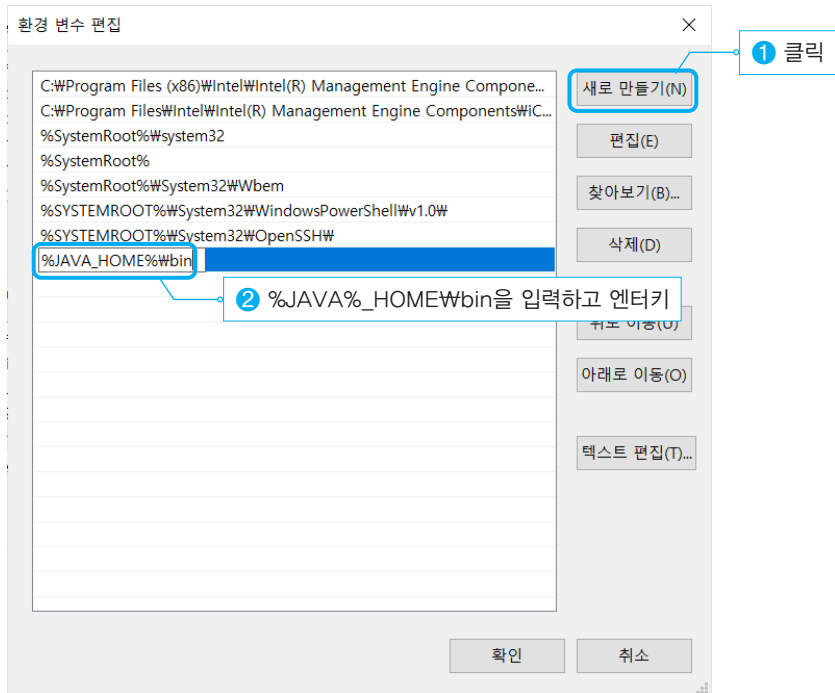
참고 '변수 값'을 직접 입력하기 보다 '디렉터리 찾아보기(D)...'버튼을 눌러서 찾는 것이 확실하다.



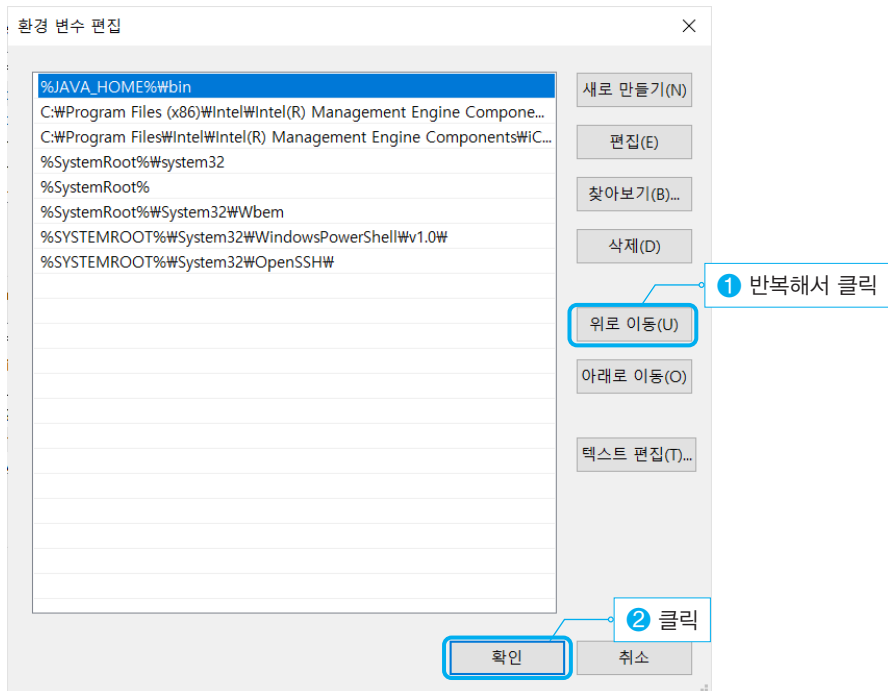
7 아래와 같은 화면이 나타나면, '시스템 변수'목록의 스크롤바를 아래로 내리면 'Path'항목을 찾을 수 있다. 이 항목을 클릭하고, '편집(I)...'버튼을 누르자.



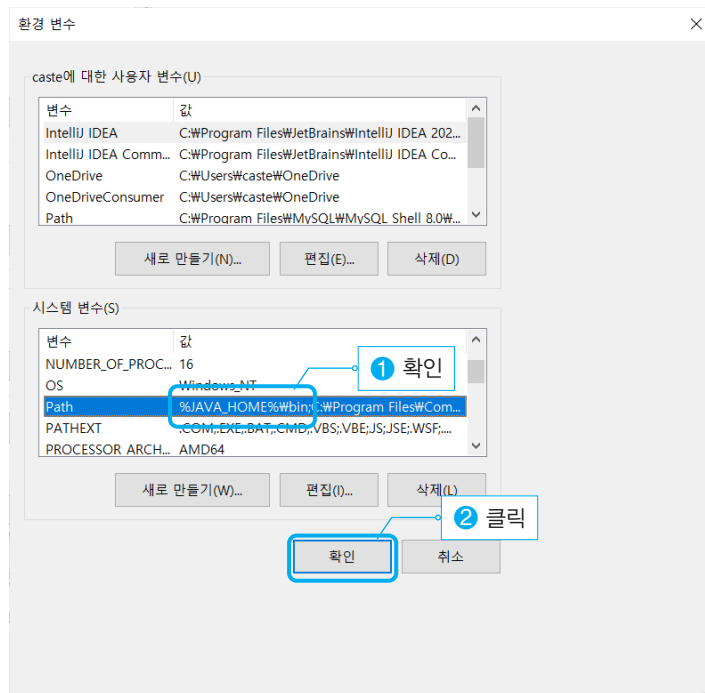
8 아래와 같이 새로운 화면이 열리면 우측 상단의 '새로 만들기(N)'버튼을 누르고, '%JAVA_HOME%\bin'을 입력하고 Enter키를 누르자.



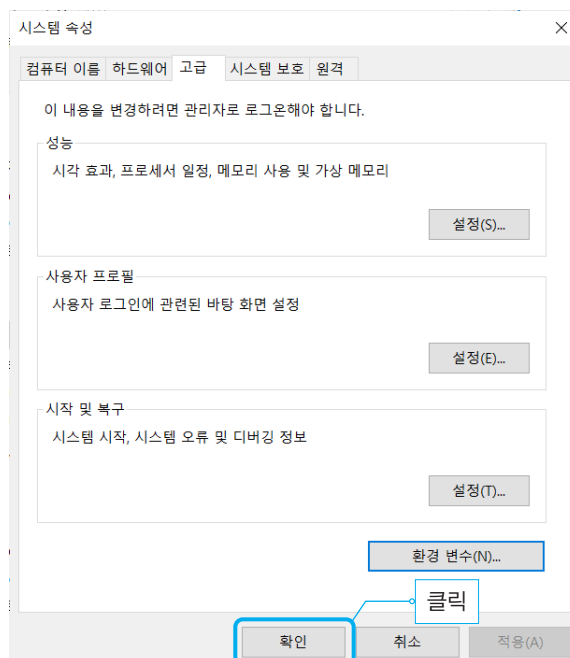
9 우측의 '위로 이동(U)'버튼을 여러번 눌러서 새로 추가한 경로가 맨 위로 올라가게 하고 '확인'버튼을 눌러서 창을 닫는다.



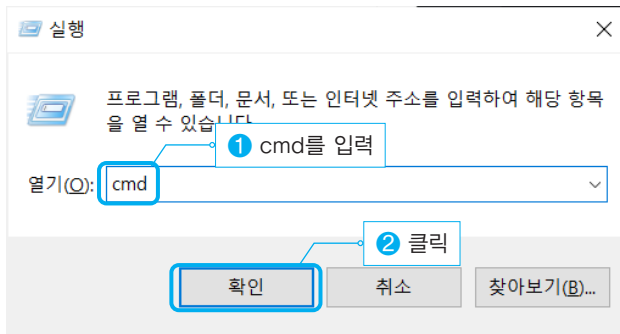
10 아래의 화면에서 전에 입력한 내용이 추가된 것을 확인하고, '확인'버튼을 클릭하자.



11 아래와 같은 화면에서 다시 '확인'버튼을 클릭하자.

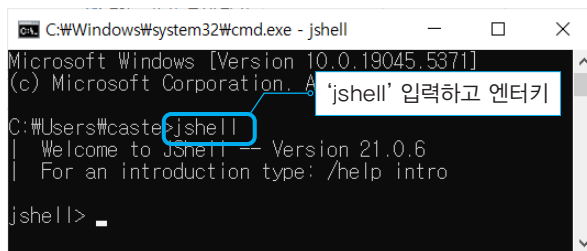


12 이제 설정은 다 끝났고, 설정이 잘되었는지 확인해 보자. '윈도우키+R'을 누르면 아래와 같은 화면이 나오는데, 'cmd'라고 입력하고 '확인'버튼을 누르자.



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.



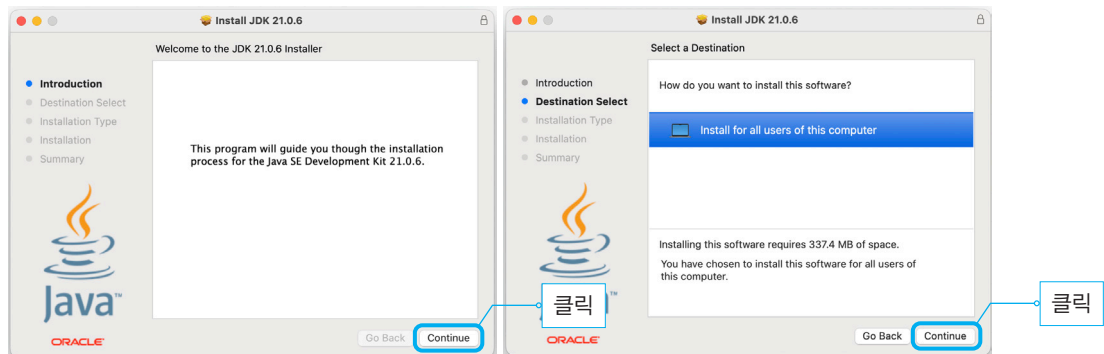
JDK 설치하기 – 맥OS

맥OS 사용자를 위한 JDK설치 방법에 대해서 알아보자.

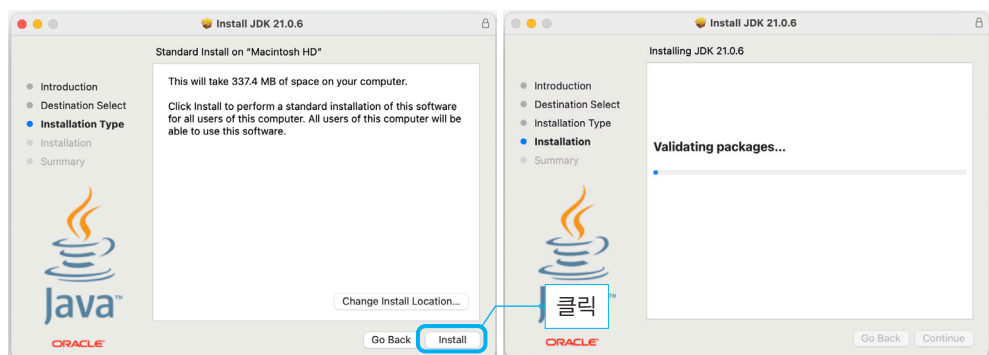
1 앞에서 다운받은 jdk-21_macos-aarch64_bin.dmg파일을 실행하면 다음과 같은 화면이 나온다. 'JDK 21.0.6.pkg'를 더블 클릭하자.



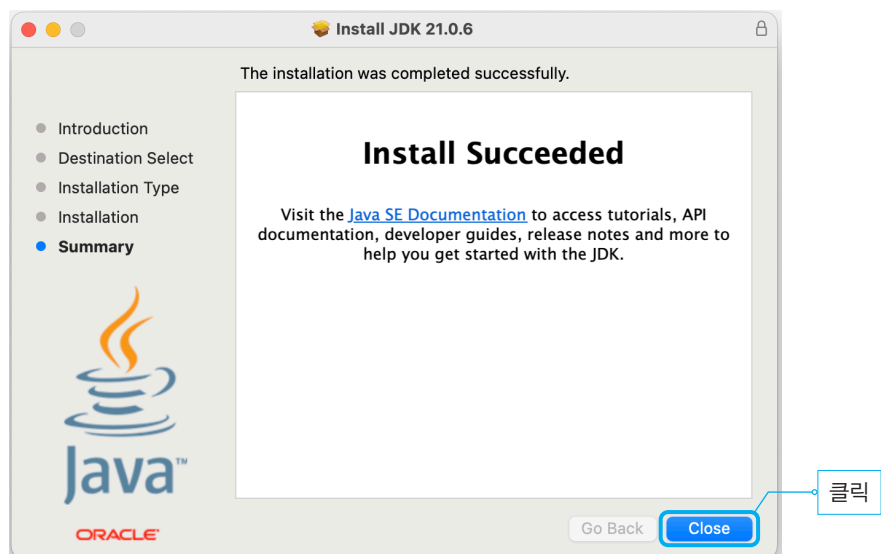
2 아래의 왼쪽 화면에서 'Continue'버튼을 클릭하면, 오른쪽 화면이 나오는데 다시 'Continue'버튼을 클릭하자.



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



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



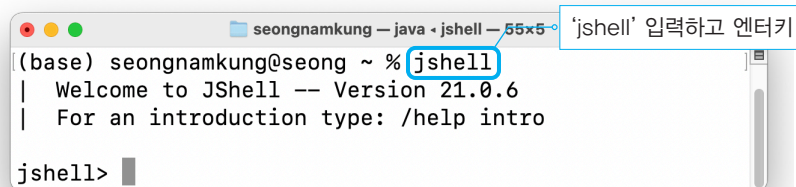
5 이제 설치가 끝났으니 설정을 할 차례이다. 사용자 디렉토리 아래의 '.bash_profile'파일을 열고 아래의 두줄을 마지막에 추가하고 저장하자.

```
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.

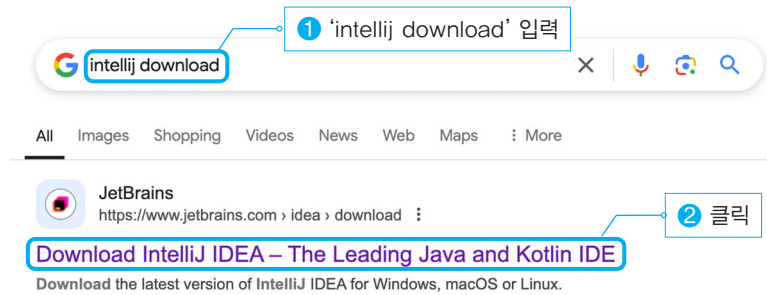
인텔리제이는 두가지 버전이 있는데, 무료 버전으로도 학습에 아무런 지장이 없기 때문에 무료 버전을 설치할 것이다.

- IntelliJ IDEA Ultimate – 유료, 30일 무료
- IntelliJ IDEA Community Edition – 무료

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

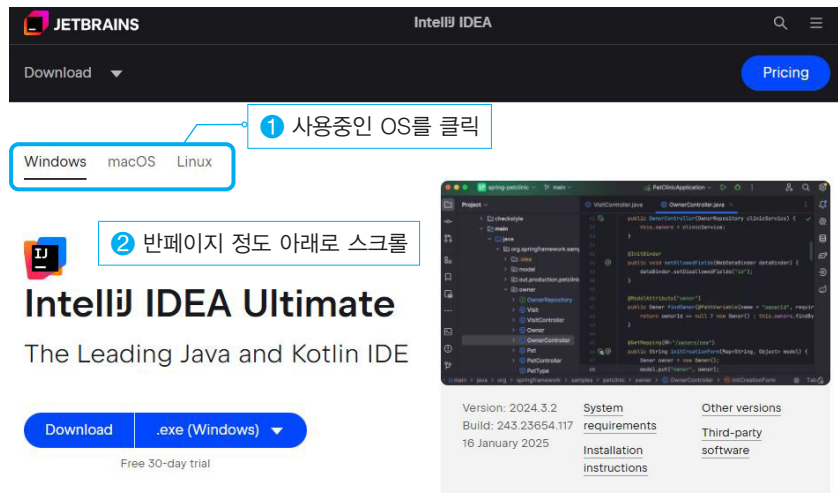
IntelliJ 다운로드하기

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



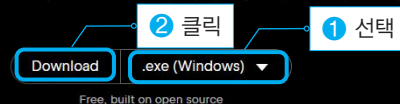
2 아래의 페이지가 나타나면 OS의 종류를 선택해서 클릭하고, 화면을 아래로 반 페이지 정도 스크롤하자. 두번째 그림처럼 IntelliJ IDEA Community Edition이 나오는데, 여기서 윈도우즈의 경우 .exe(Windows), 맥OS의 경우 .dmg(Apple Silicon)을 선택하고, 'Download'버튼을 클릭하자.

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



We're committed to giving back to our wonderful community, which is why IntelliJ IDEA Community Edition is completely free to use

IntelliJ IDEA Community Edition
The IDE for Java and Kotlin enthusiasts



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

Thank you for downloading IntelliJ IDEA!

Your download should start shortly. If it doesn't, please use the **direct link**.

Download and verify the file [SHA-256 checksum](#).

Learn more about the [digital signatures of JetBrains binaries](#).
[Third-party software used by IntelliJ IDEA Ultimate Edition](#)

수동 다운로드 링크



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 다운로드 받은 파일의 이름은 새로운 버전이 나오면 달라질 수 있다.



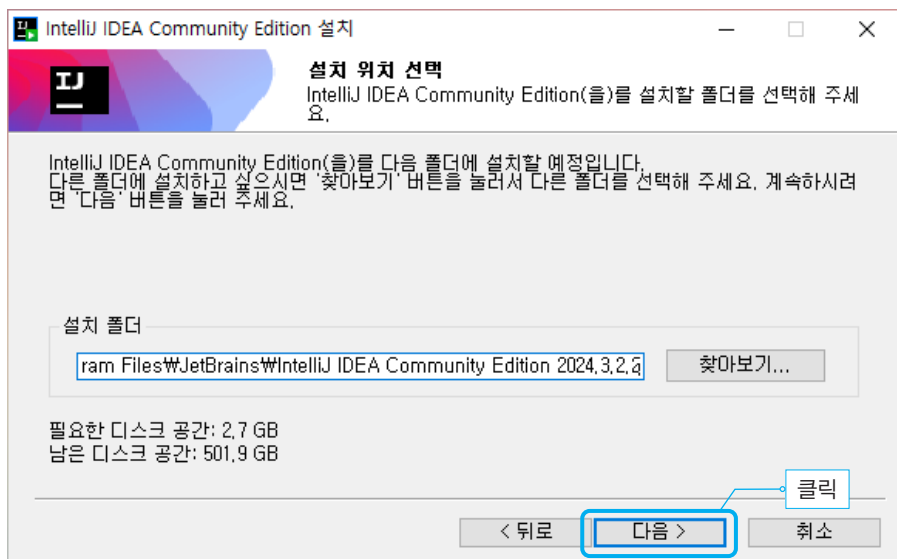
IntelliJ 설치하기 – 윈도우즈

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

참고 다운로드 받은 파일의 이름은 새로운 버전이 나오면 달라질 수 있다.



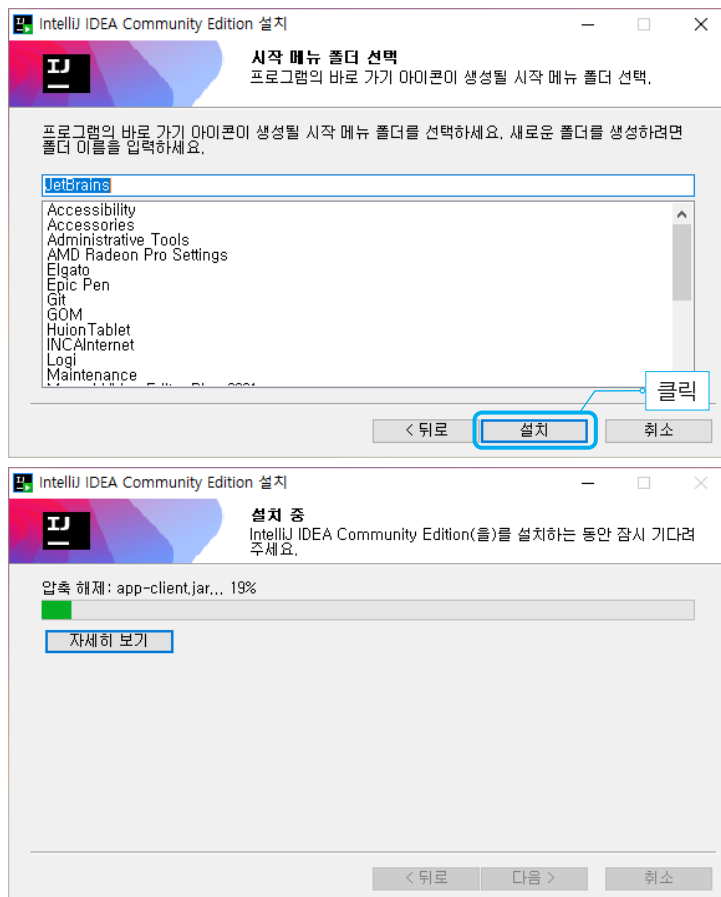
2 인텔리제이를 설치할 위치를 지정하는 화면이 나오는데, 원하는 곳으로 변경해도 되지만 특별한 이유가 없으면 기본을 지정된 위치에 설치하자. 그냥 '다음 >' 버튼을 클릭하자.



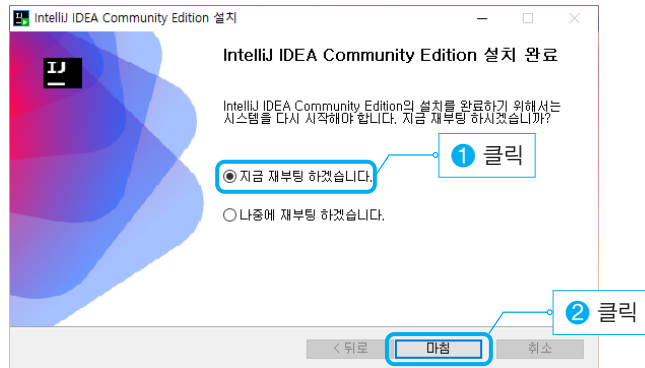
3 아래에 표시된 옵션들을 클릭해서 체크하고, '다음 >' 버튼을 누른다.



4 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.



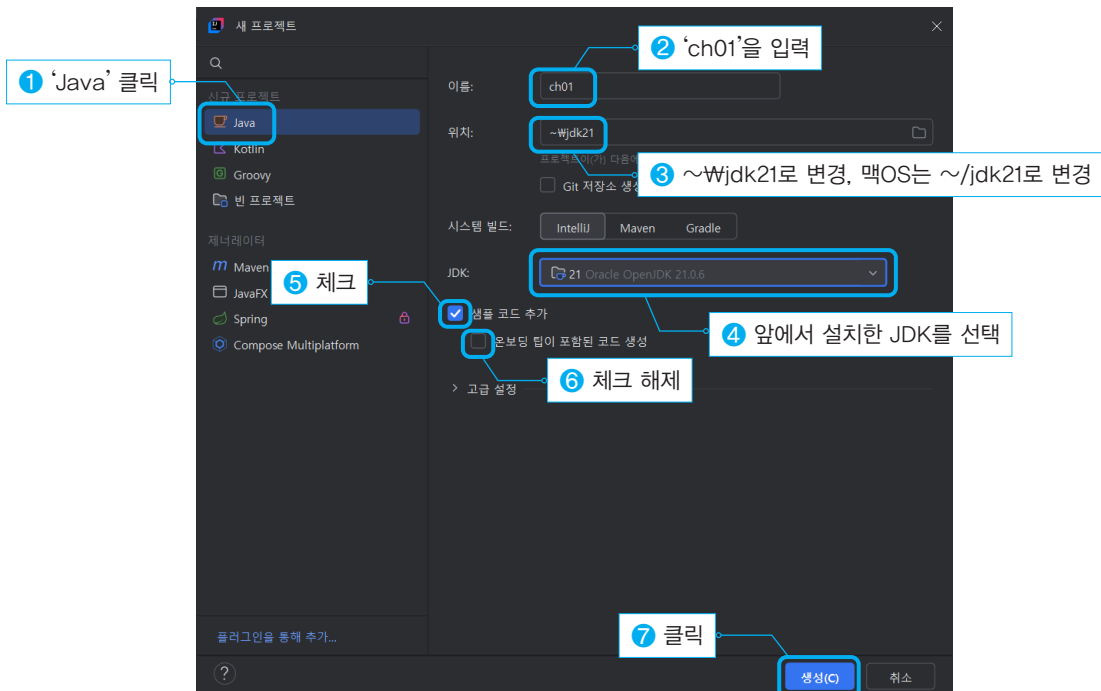
2 아래의 화면에서 좌측의 '프로젝트'를 클릭하고, 중앙의 '새 프로젝트'를 클릭하자.

참고 | 아래의 화면이 나타나지 않으면, 메뉴에서 File > New > Project...를 클릭하면 다음 단계의 화면이 나타난다.

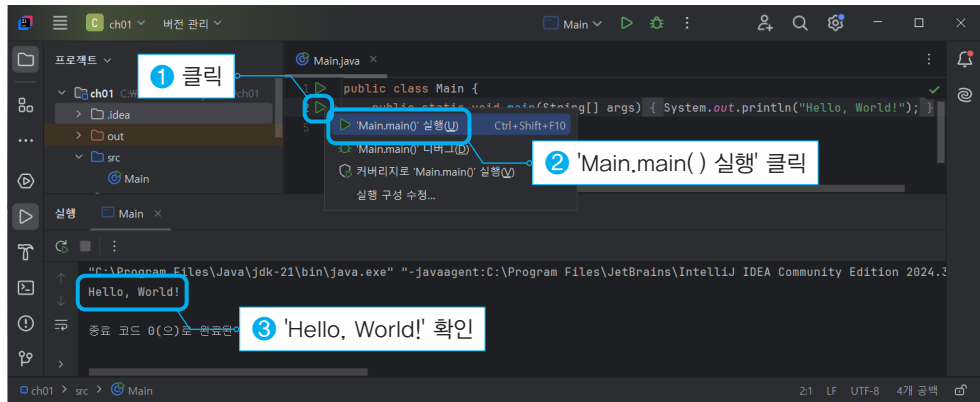
참고 | 좌측의 '프로젝트' 아래의 '사용자 지정'을 클릭하면, 사용 언어와 테마 등의 설정을 변경할 수는 화면이 나온다.



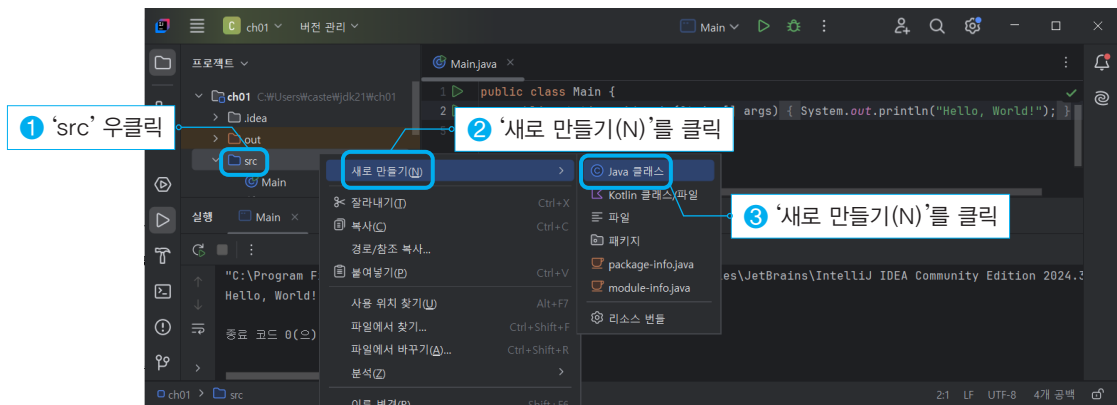
3 새로 생성할 프로젝트의 정보를 입력하는 화면이 나오는데, 프로젝트의 이름을 입력하고 경로를 변경하자. JDK는 전에 설치한 것이 자동으로 나타난다. 앞으로 챕터마다 이름만 다르게 새 프로젝트를 생성하자.



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 눌러서 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 비주얼 스튜디오 코드(Visual Studio Code)라는 것도 있다.

❗ 참고 ❗ 비주얼 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.  
    }  
}
```

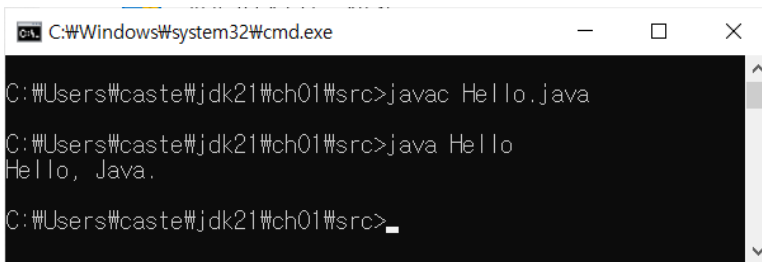
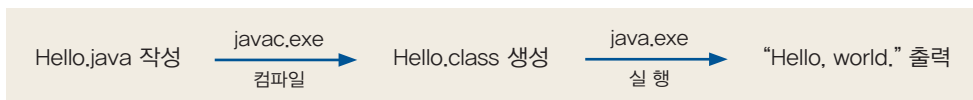
▼ 실행결과

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.



▲ 그림 1-3 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 블록{} 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
<pre> Hello2.java public class Hello2 {} class Hello3 {} </pre>	public class가 있는 경우, 소스파일의 이름은 반드시 public class의 이름과 일치해야한다.
<pre> Hello2.java class Hello2 {} class Hello3 {} </pre>	public class가 하나도 없는 경우, 소스파일의 이름은 'Hello2.java', 'Hello3.java' 둘 다 가능하다.

잘못된 작성 예	설 명
<pre> Hello2.java public class Hello2 {} public class Hello3 {} </pre>	하나의 소스파일에 둘 이상의 public class가 존재하면 안 된다. 각 클래스를 별도의 소스파일에 나눠서 저장하던가 아니면 둘 중의 한 클래스에 public을 붙이지 않아야 한다.
<pre> Hello3.java public class Hello2 {} class Hello3 {} </pre>	소스파일의 이름이 public class의 이름과 일치하지 않는다. 소스파일의 이름을 'Hello2.java'로 변경해야 맞다.
<pre> hello2.java public class Hello2 {} class Hello3 {} </pre>	소스파일의 이름과 public class의 이름이 일치하지 않는다. 대소문자를 구분하므로 대소문자까지 일치해야한다. 그래서, 소스파일의 이름에서 'h'를 'H'로 바꿔야 한다.

▲ 표 1-2 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 명심하자.

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 달지 말고 주석이 필요 없을 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 독자들 도 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 입력해 보고 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 섣길로 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. 이 중에서 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 확인한 다.

The screenshot shows the Naver Cafe interface for '남궁성의 코드초보스터디(java, c언어)'. The header features the cafe name, URL (<http://cafe.naver.com/javachobostudy>), and the founding date (Since 2004.07.24). Below the header, there are navigation tabs for '전체글보기', '필수자바강의', '초보프로그래머에게', '이미지모아보기', and '베스트게시물'. The left sidebar contains '카페정보' (Cafe Info) with details about the owner '남궁성' and a '2014 네이버 대표카페' badge, and '나의활동' (My Activity) showing '114,434' posts and '21,544' members. The main content area displays a list of posts under the '전체글보기' tab. The posts are as follows:

제목	작성자	작성일	조회	좋아요
[공지] 자바의 정석 3판 2016년 1월 25일에 출시예정입니다. [185]	남궁성	2015.11.03	2821	0
[필독] 자동등업방법입니다. 여기에 댓글 5개달아주세요. [14462]	남궁성	2015.05.28	5997	0

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 받을까봐 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. 예러 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
