

독하게 시작하는 Java Part 1

취업목적! 차별화된 경쟁력을 갖추려는 입문자를 위한 안내

넌넌한 개발자 최호성 (cx8537@naver.com)

YouTube: 넌넌한 개발자 TV

문서 개정이력

[illegible]

1. 시작에 앞서...

Windows 환경에서 독하게 Java를 시작하기 위한 준비운동

학습준비에 대해 가정하는 것들

- Java책은 한 권 가지고 있음 (서점에서 구매한 것)
- 넓고 얕게 외워서 컴공 전공자 되기
 - CPU(Register, Cache), RAM, SSD, HDD
 - File system, OS
 - ASCII 코드와 인코딩
- 한 번에 끝낸다는 말에 속지 않을 정도의 분별력
 - 국비교육의 한계
- CS이론의 중요성에 대해 공감하고 있음
 - Java 백엔드 개발자로 가기 위한 필수 이론

Windows OS 사용에 관한 용어

- 파일, 파일명(이름 + 확장명)
 - Test.txt, abc.docx
- 폴더, 디렉토리, 경로(Path)
 - C:\Windows\System\notepad.exe
- 절대경로와 상대경로
 - .\test.txt, ..\test.txt
- 바이너리와 텍스트
 - .exe, .txt

알고 있으면 좋을 콘솔 명령어

- 명령 프롬프트에서 명령 실행
 - PATH 지정된 명령 수행 확인
 - 윈도우를 Linux처럼 사용하기
- dir
 - 디렉토리(폴더)에 저장된 파일 목록 확인
- cd
 - 디렉토리 위치 변경
 - ex) cd ..

Windows OS 사용에 관한 용어

- 실행파일

- .exe, .dll, .sys

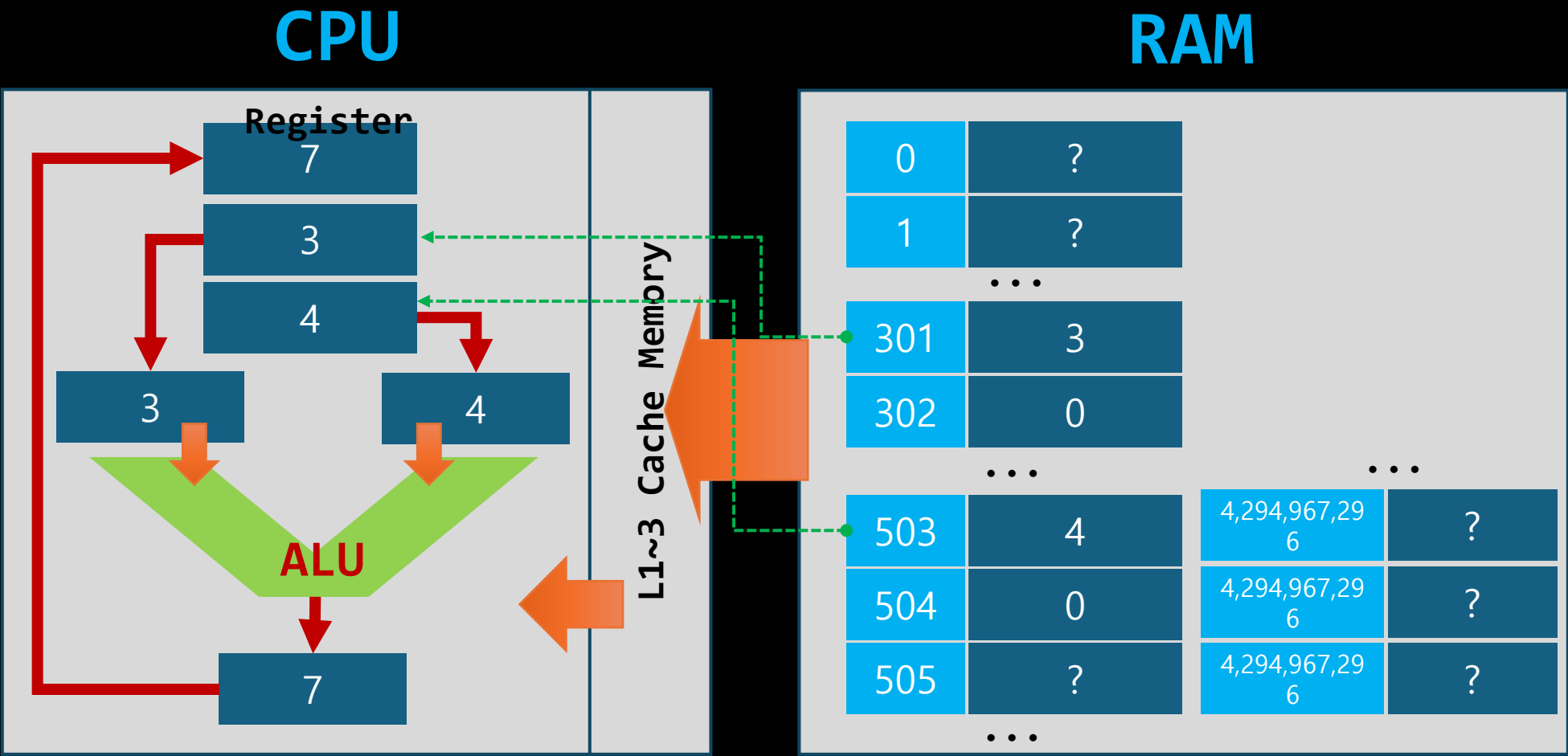
- 프로그램과 프로세스

- word.exe는 프로그램 word.exe를 실행하면 프로세스
 - word 프로세스는 word.exe에 대한 인스턴스

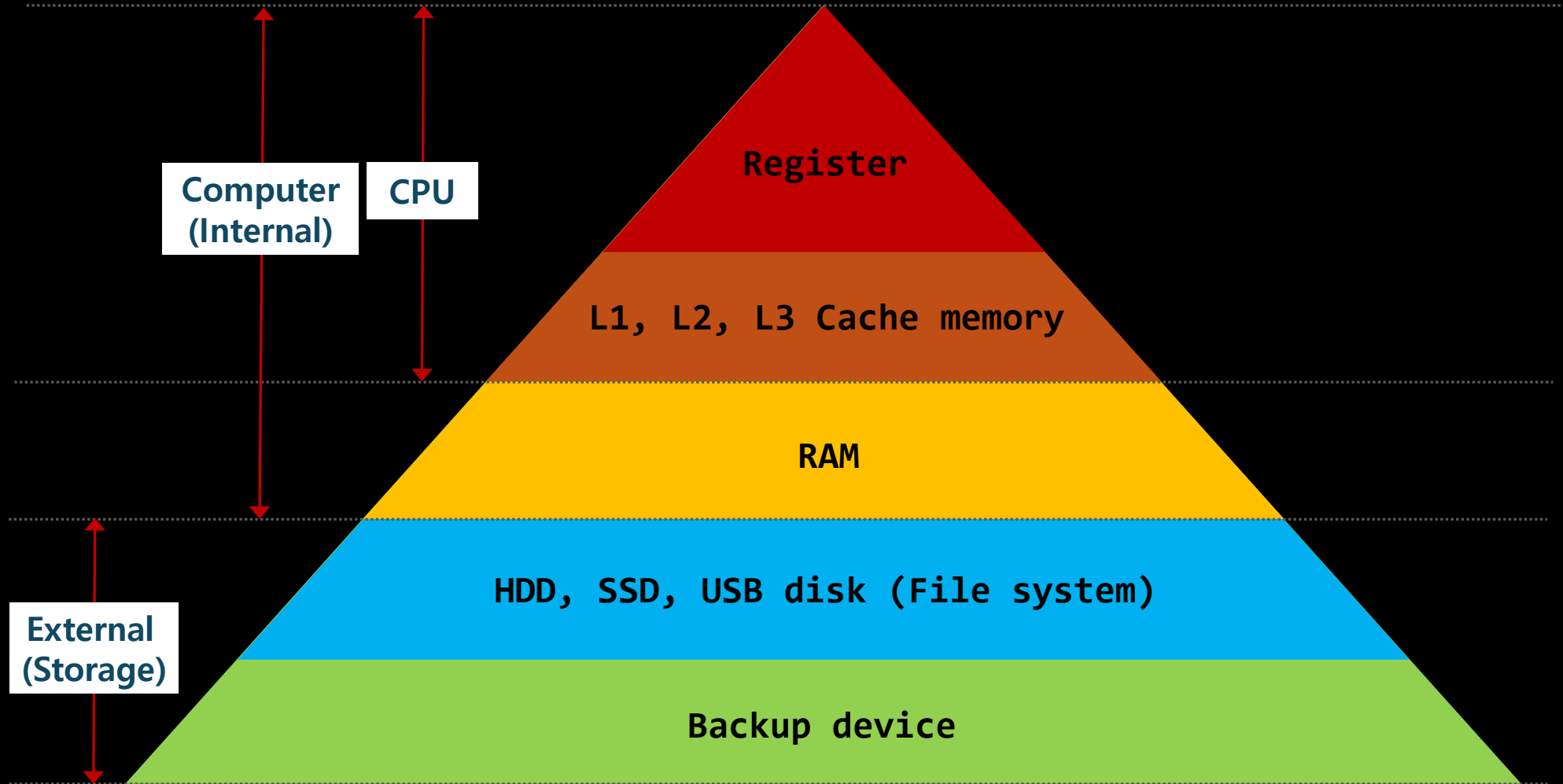
- 스레드 (선택)

- 한 프로세스 내부에 공존하는 개별적인 실행(연산) '흐름'

컴퓨터 구조에 대한 상식



메모리 구조



2진수, 16진수 변환

2진수(4비트)	16진수
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

2진수(4비트)	16진수
1010	A (10진수 10)
1011	B (10진수 11)
1100	C (10진수 12)
1101	D (10진수 13)
1110	E (10진수 14)
1111	F (10진수 15)
※ 4비트는 16진수 한 자리 숫자다. ※ 16진수는 0~F(십진수 15) 까지 한 자리에 쓴다.	

실행의 다른 이름 '연산'

- 실행의 실체는 CPU 연산
- 연산에 필요한 정보는 메모리에서 가져와야 함
- 메모리에서 CPU 레지스터로 전달된 후 연산
- 연산 결과가 저장된 레지스터 값을 다시 메모리로 보내는 과정을 반복

Computer와 운영체제



2. 목표는 취업!

제대로 달리는 학습순서

Spring framework으로 넘어가기 위한 준비

(Java reflection + Custom annotation, Maven+Gradle, WAS, Design pattern)

웹 인프라 기술에 대한 이해와 DB

(HTML+CSS+JS, HTTP, SSL, SQL+RDBMS, JDBC, ORM)

3

시스템 활용 프로그램 작성

(Thread & Synchronization, File I/O, Stream, Socket)

2

객체지향 프로그래밍 + 자료구조

(OOP언어의 일반적인 특징과 문법, 상속의 존재 이유, 프로그래밍 경험 쌓기)

1

기초적인 문법, 절차적 프로그래밍

(특정 언어에 대한 의존성이 낮음)

교육목표

- **절차적 프로그래밍에 대한 이해**

- 문제를 정의하고 해결 방법에 대해 글로 쓰는 것이 우선
- 절대로 타자치면서 생각하지 말 것
(우연에 맡기는 개발자가 되는 흔한 방법)
- 변수, 상수 및 범위 경우에 수를 파악하는 연습이 필요

- **Java 언어 문법**

- 모든 예제는 직접 작성하는 것이 중요
- 객체지향은 제외하고 Class에 대한 개념과 문법까지 학습

현실적인 이야기, 취업

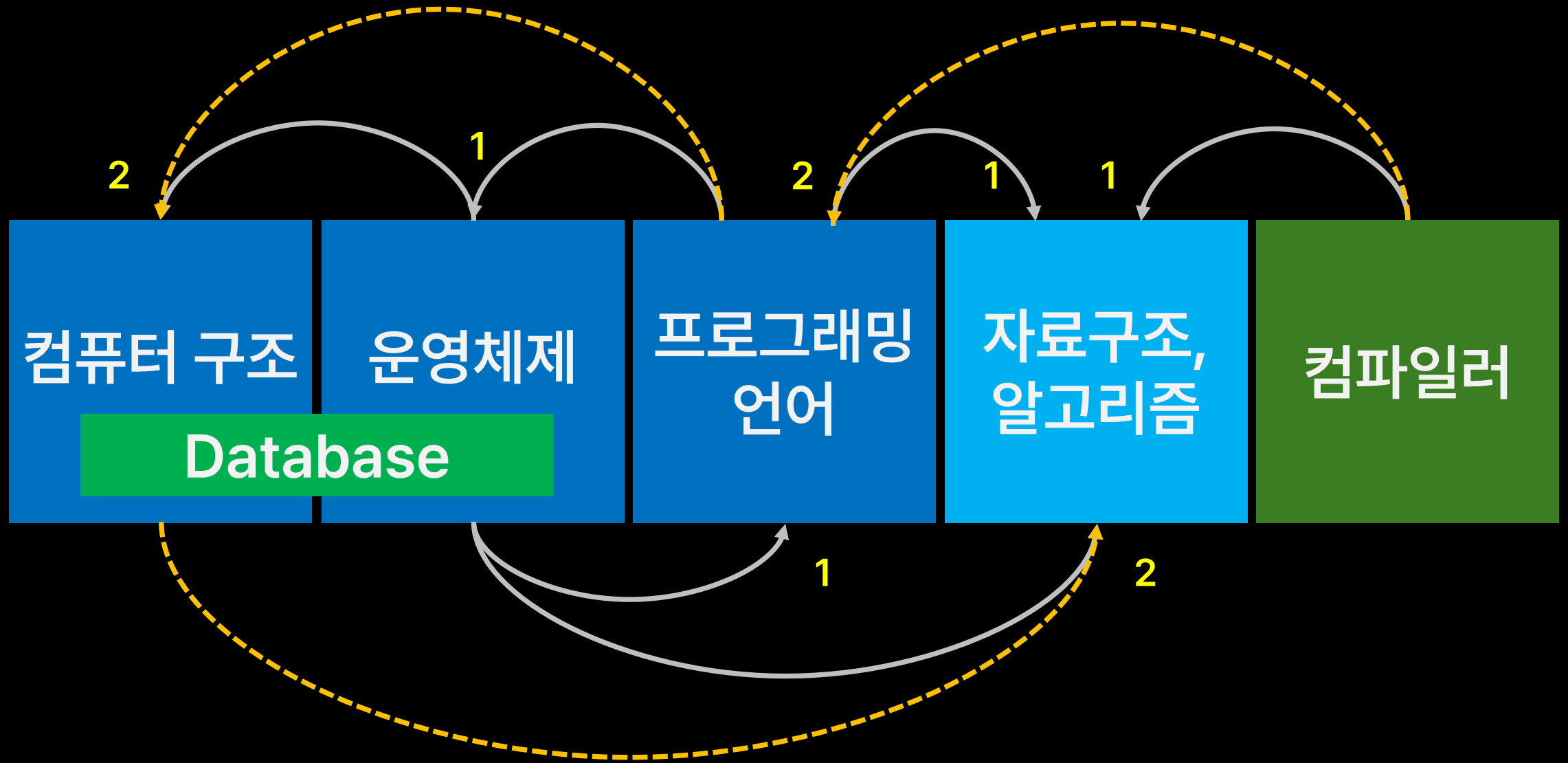
- 좋은 기업 vs 여건이 열악한 SI 기업
- 실력 있는 시니어의 존재가 **빛나는 순간은 장애 대응**
 - 장애 상황 인지 그 자체도 능력 (실시간 감시체계)
 - 빠른 대응과 근본원인 규명은 별개의 문제
 - 어떤 경우라도 인프라에 대한 높은 이해도는 필수
- 좋은 기업을 원한다면 **먼저 그들이 원하는 인재** 혹은 **그들이 동료로 함께 하고 싶은 사람이 될 것!**
 - '기본에 충실'할 것 (직장은 교육센터가 아님)
 - 문제에 대해 함께 의견을 교환할 수 있어야 함

프로그래밍에 익숙해지는 시간

- 의지력이 강한 성인 기준 매일 꾸준히 최소 21일
(매일 4시간 이상)
- 보편적 기준 2개월 이상
- 프로그래밍, 자료구조, OOP등의 개념을 6개월 동안 배우는 것을 권장
 - 절차적 기법 (2개월)
 - 자료구조 및 알고리즘 (2개월)
 - OOP (2개월)



전공자가 되고 싶은 개발자의 기본기



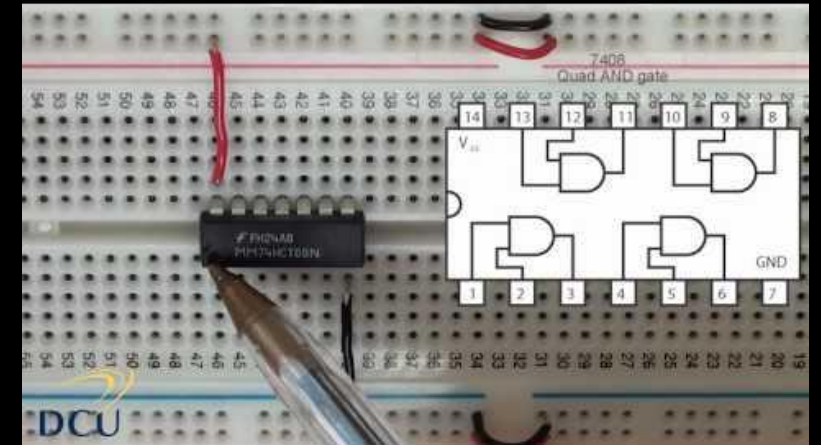
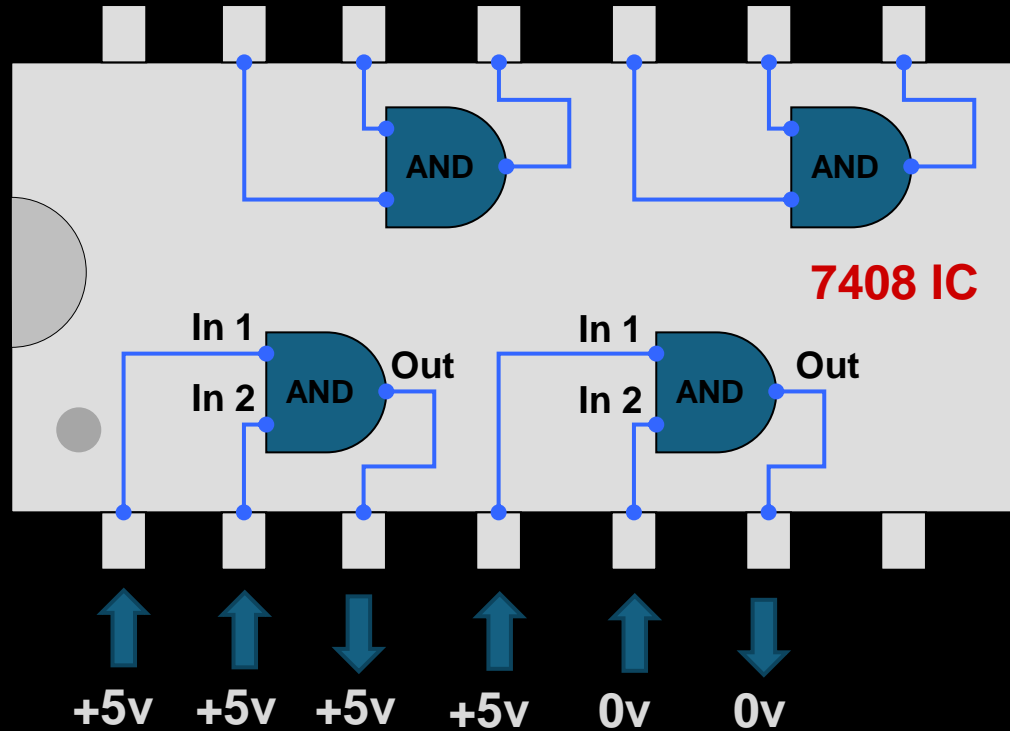
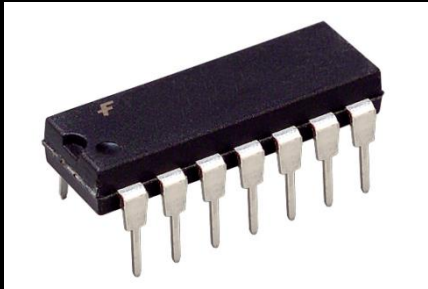
3. 프로그래밍 언어의 시작

Java를 넘어 모든 프로그래밍 언어에 관한 이야기

고급어와 저급어

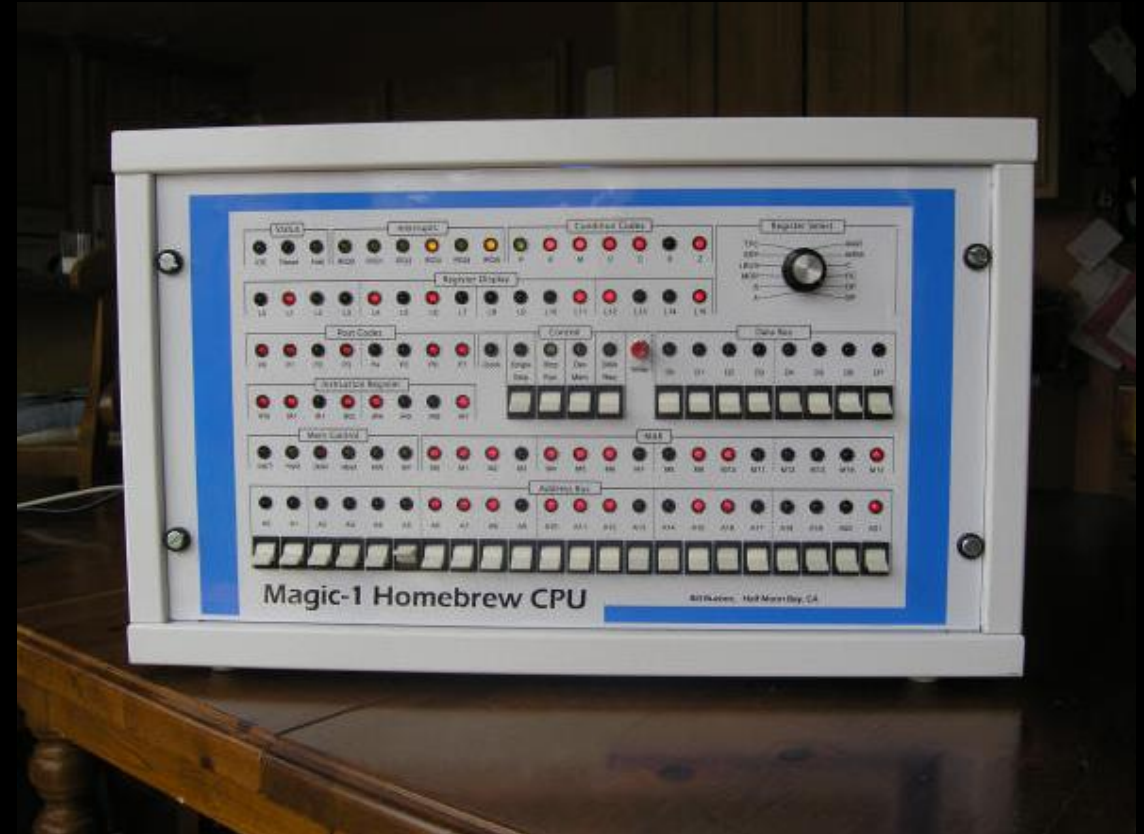
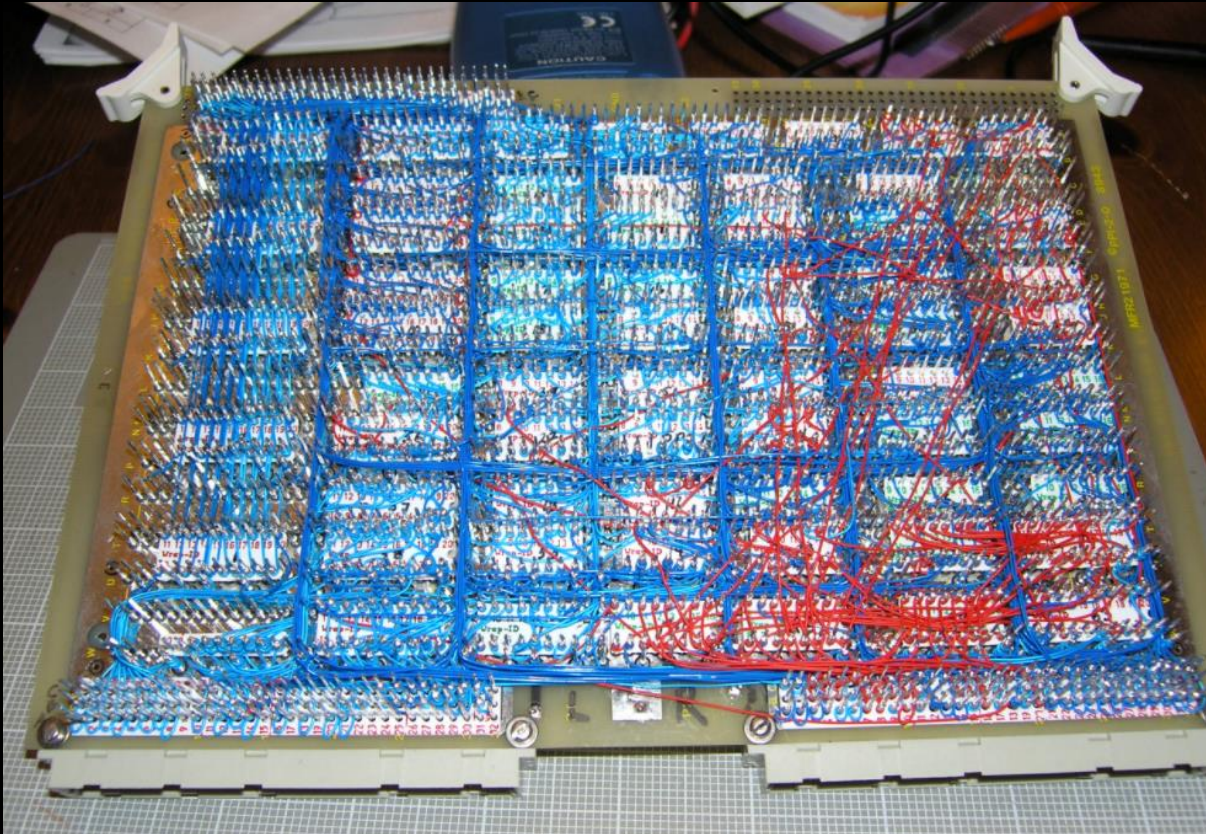
- CPU가 인식 할 수 있는 명령어는 기계어
 - CPU 접점에 전기 신호(+5v)를 넣으면 1, 그렇지 않으면 0
 - 2진수로 표기 가능
(보통 16진수로 변환해 표기)
- 장치에 의존적인 기계어를 사람이 이해하기 용이한 문자열 표기로 정의한 것이 고급어
 - C, C++
 - Java

디지털 회로 (비트 연산의 이해)



수제 CPU

<http://www.homebrewcpu.com/>



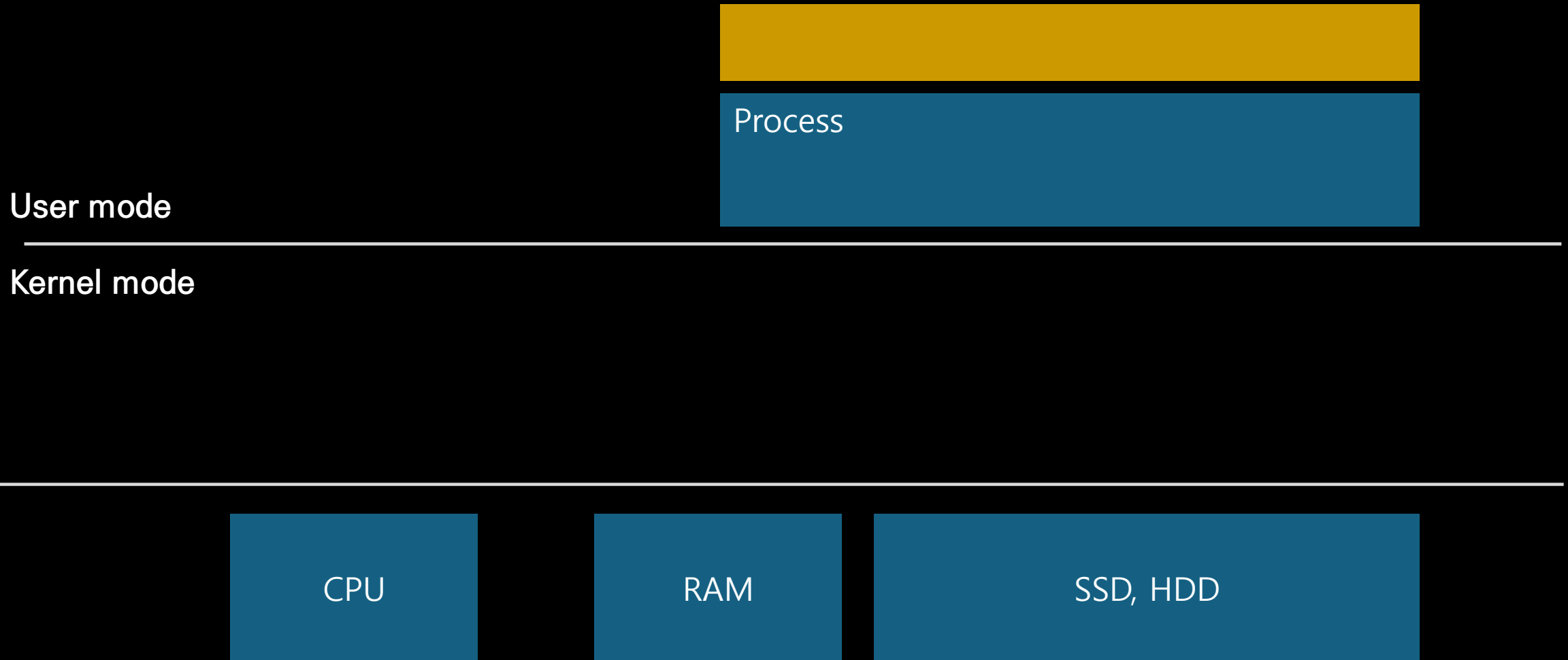
기계어 코드 예

메모리 1									
주소: 0x00007FF65C4F1AC0								열: 8	
0x00007FF65C4F1AC0	48	89	4c	24	08	55	57	48	H?L\$.UWH
0x00007FF65C4F1AC8	81	ec	08	01	00	00	48	8d	??....H?
0x00007FF65C4F1AD0	6c	24	20	48	8d	0d	36	05	l\$ H?.6.
0x00007FF65C4F1AD8	01	00	e8	b4	f8	ff	ff	b9	..????..?
0x00007FF65C4F1AE0	48	00	00	00	ff	15	1e	f8	H.....?
0x00007FF65C4F1AE8	00	00	48	89	45	08	41	b8	..H?E.A?
0x00007FF65C4F1AF0	48	00	00	00	33	d2	48	8b	H...3?H?
0x00007FF65C4F1AF8	4d	08	e8	e0	f7	ff	ff	48	M.???..H
0x00007FF65C4F1B00	8b	45	08	4c	8b	85	00	01	?E.L??...
0x00007FF65C4F1B08	00	00	ba	40	00	00	00	48	..?@...H
0x00007FF65C4F1B10	8b	c8	ff	15	08	f8	00	00	??...?..

```
mov     qword ptr [rsp+8],rcx
push    rbp
push    rdi
sub     rsp,108h
lea     rbp,[rsp+20h]
lea     rcx,[__AA290256_SingleList@c
call    __CheckForDebuggerJustMyCode (
```

- Machine code는 전기 신호의 조합(0, 1)
- 2진수로 표기하기 보다는 16진수로 변환해 표기
- 0x는 16진수를 의미
- 보통 디스어셈블 코드로 표기

고급어와 저급어



오류를 버그라 부르게 된 사연



- 미해군제독(1906~1992)이자 컴퓨터 공학자
- 함포 단도 계산과 관련해 컴퓨터 개발 프로젝트에 참여
- 컴파일러라는 개념의 창시자
- COBOL(COmmon Business Oriented Language)의 어머니
- NVIDIA H100 (Hopper)

COBOL 코드 예

<https://www.ibmmainframer.com/cobol-tutorial/cobol-hello-world/>

IDENTIFICATION DIVISION.

PROGRAM-ID. IDSAMPLE.

ENVIRONMENT DIVISION.

PROCEDURE DIVISION.

 DISPLAY 'HELLO WORLD'.


 STOP RUN.

First actual case of bug being found

9/9

0800 Antam started
1000 " stopped - antam ✓ { 1.2700 9.037 847 025
1300 (032) MP - MC 1.582642000 9.037 846 995 correct
2.130476415 (2) 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
in relay 11.000 test.

1100 Started Cosine Tape (Sine check)
1525 Started Multi-Adder Test.

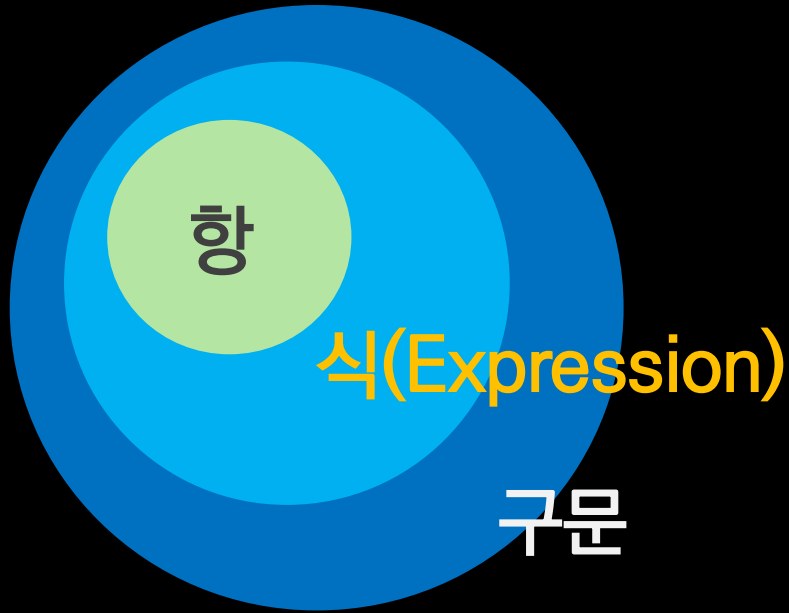
1545  Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1630 Antam started.
1700 closed down.

- 1946년 Mark II의 계산오류의 원인을 규명
- 릴레이(#70, Panel F)에 끼어든 나방 때문에 발생한 것으로 확인 후 이를 기록
- 에디슨이 말한 버그의 First actual case

항, 식, 구문



```
x = ( y + 3 );  
x++;  
x = y = 0;  
proc( arg1, arg2 );  
y = z = ( f( x ) + 3 );
```

- 여러 항이 모여 식을 이룸
- 식은 평가 대상
(식을 평가하기 위해서는
계산(연산)을 해야 함)
- 여러 식을 모아 하나의
구문을 이루며 고급어 기준
실행단위가 됨
- 연속되는 구문을 위에서
아래로 순차적 실행

컴파일러와 인터프리터

Compiler

- 고급어 소스코드를 기계어로 번역하는 프로그램
- 전체 소스코드를 모두 기계어로 변환한 후 실행
- 성능 최적화가 용이하여 다수 언어가 채택
- C, C++등이 여기에 해당

Interpreter

- 고급어 소스코드를 직접 실행하는 프로그램이나 환경을 의미
- 보통 한번에 한 줄 단위로 실행
- 성능(특히 속도)면에서 컴파일러 방식보다 느림
- JavaScript나 Python등이 여기에 해당

4. 프로그램을 ‘쓰다’

프로그래밍에 대한 개념적 이해

- 프로그래밍 그 자체는 절차적 순서를 기술한 글쓰기
- 논리적으로 하나씩 풀어내는 훈련이 필요
(※ 프로그래밍 언어 문법과 별개의 문제)
- 최대한 단순하게 시작

절차적 글쓰기

내 아이가 살아갈
로봇 세상

SBS



너희들이 만드는 방법을 적어 와서
이 아빠한테 어떻게 만드는지 알려 줄 수 있겠니?

절차적 글쓰기

내 아이가 살아갈
ROBOT
로봇세상

SBS



칼로 식빵 하나를 골고루 펴 바른다

절차적 글쓰기

우리의 언어표현은 생략된 것들이 의외로 많음

1. 잼, 식빵, 나이프를 가져와 식탁 위에 올려둔다.
2. 양손을 이용해 잼이 담긴 병의 뚜껑을 연다.
3. 오른손으로 나이프를 들어 병에 든 잼을 조금 뜯다.
4. 식빵의 넓은 면이 위로 보이도록 왼손에 올린다.
5. 나이프에 붙은 잼을 식빵 넓은 면에 고르게 펴 바른다.

※고민 없는 글쓰기는 부적격으로 가는 지름길

상수




- 연산식을 기술하는 시점에 값이 정해진 수
- 값이 확정되어 앞으로 변할 가능성이 없는 수
- 리터럴 (상수)
 - ‘A’, “Hello”, 3, 3.4F, 123.45
- 심볼릭 상수
 - final

변수

- 연산식을 기술하는 시점에 값이 정해지지 않은 수
- 구체화하지 않았거나 앞으로 변경될 가능성이 있는 수 (혹은 미지의 수)
- 개발자가 메모리를 사용하는 가장 일반적인 방법
- 구체적으로 결정되는 값에 따라 연산의 내용이 달라질 수 있는 원인으로 작용
 - 예) 날씨변수 (맑음, 흐림, 비, 눈, 온도)
 - 의존성(Dependency)의 시작

잼바르기 절차와 변수

식빵 대신 크로와상?

1. 잼, , 나이프를 가져와 식탁 위에 올려둔다.
2. 양손을 이용해 잼이 담긴 병의 뚜껑을 연다.
3. 오른손으로 나이프를 들어 병에 든 잼을 조금 뜯는다.
4. 의 넓은 면이 위로 보이도록 왼손에 올린다.
5. 나이프에 붙은 잼을  넓은 면에 고르게 펴 바른다.

경우의 수 그리고 흐름 제어




- 빵의 종류가 세 가지로 한정된다고 가정
 - 식빵, 크로와상, 소금빵
- 잼과 더불어 버터가 존재하는 경우를 고려
 - 잼만 바르는 경우
 - 버터만 바르는 경우
 - 둘 다 바르는 경우
- 잼 바르기 절차는 여전히 유효한 것인지 확인

재사용을 고려한 잼바르기 절차

어떤 빵이든 이렇게 처리하기로 단위화

잼바르기절차()

{

1. 잼, , 나이프를 가져와 식탁 위에 올려둔다.
2. 양손을 이용해 잼이 담긴 병의 뚜껑을 연다.
3. 오른손으로 나이프를 들어 병에 든 잼을 조금 뜬다.
4. 의 넓은 면이 위로 보이도록 왼손에 올린다.
5. 나이프에 붙은 잼을 넓은 면에 고르게 펴 바른다.

}

5. 개발환경 구축

JDK, JRE

JDK는 개발환경, JRE는 실행환경

JDK

- Java Development Kit
 - JRE를 포함
 - 컴파일러 등 프로그램 개발도구가 포함
 - 버전 중요(언어버전과 일치)
- Java SE 스펙 준수
 - Open JDK, Oracle JDK

JRE

- Java Runtime Environment
 - Java 프로그램을 실행하는데 필요한 패키지
- 포함사항
 - JVM, 각종 명령
 - 클래스 라이브러리


IntelliJ 다운로드

<https://www.jetbrains.com/ko-kr/idea/download/?section=windows>

IntelliJ IDEA JetBrains IDEs

2024.3에서 지원 새로운 기능 기능 ▼ 리소스 **가격 책정**

JetBrains는 멋진 도움을 주고 있는 커뮤니티에 보답하고자 하며, 이것이 IntelliJ IDEA Community Edition을 완전 무료로 제공하는 이유입니다.

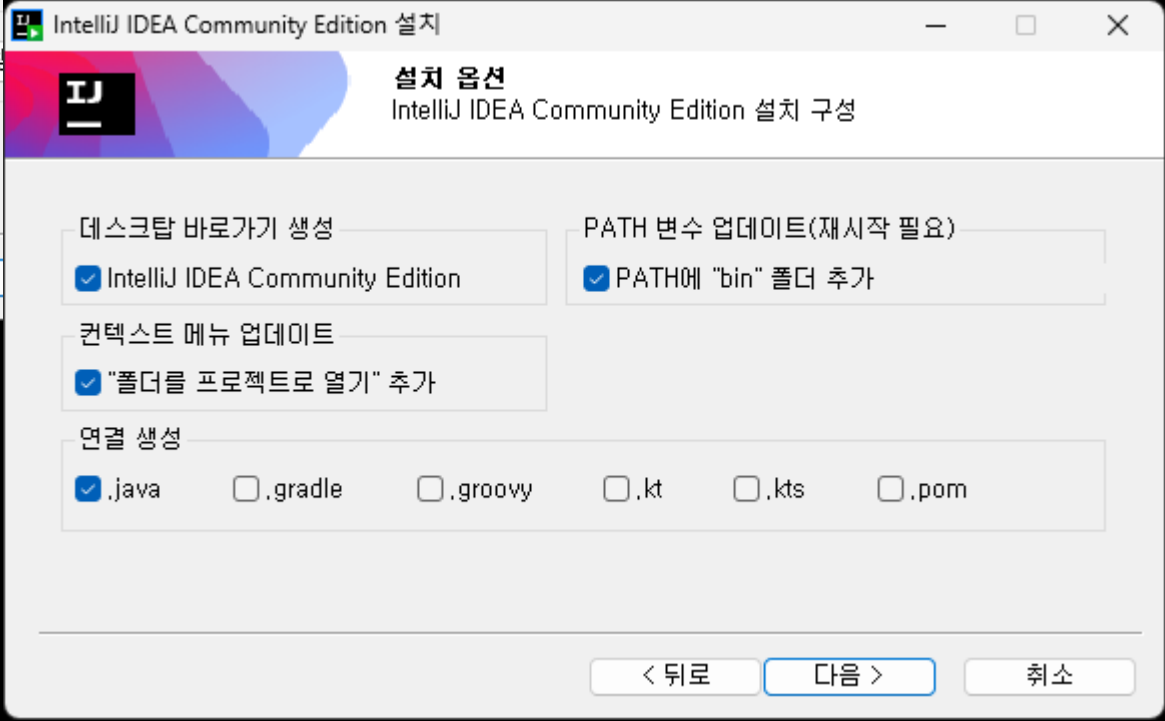
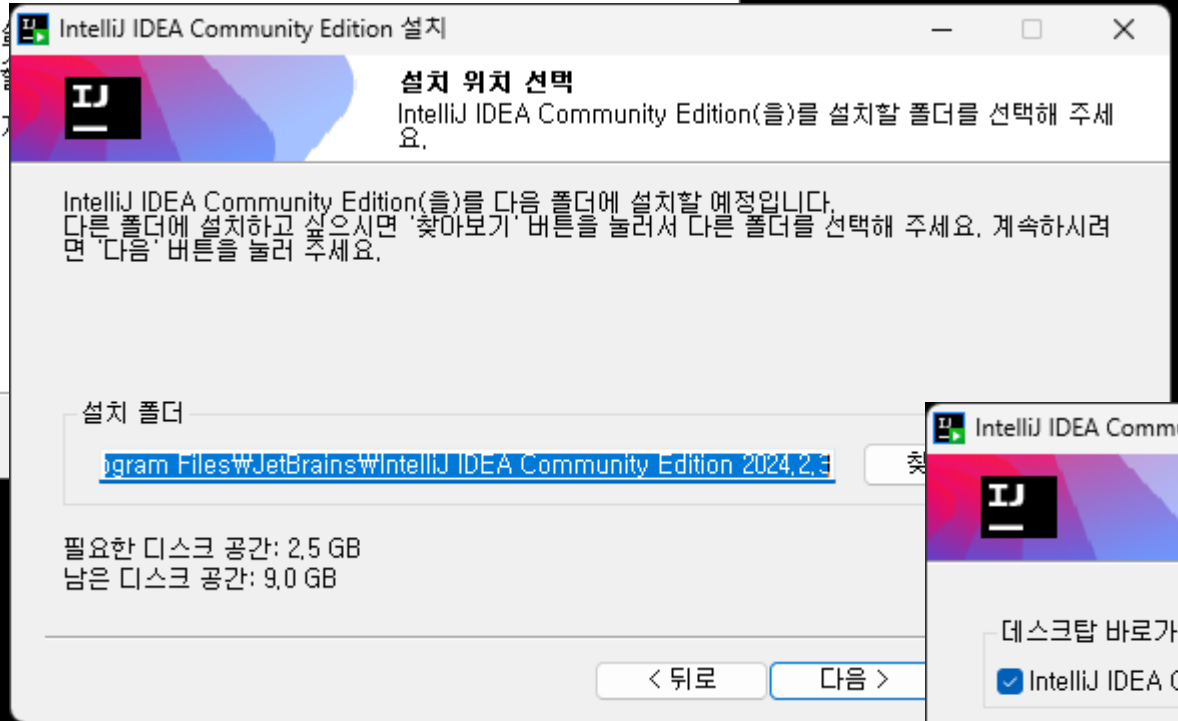
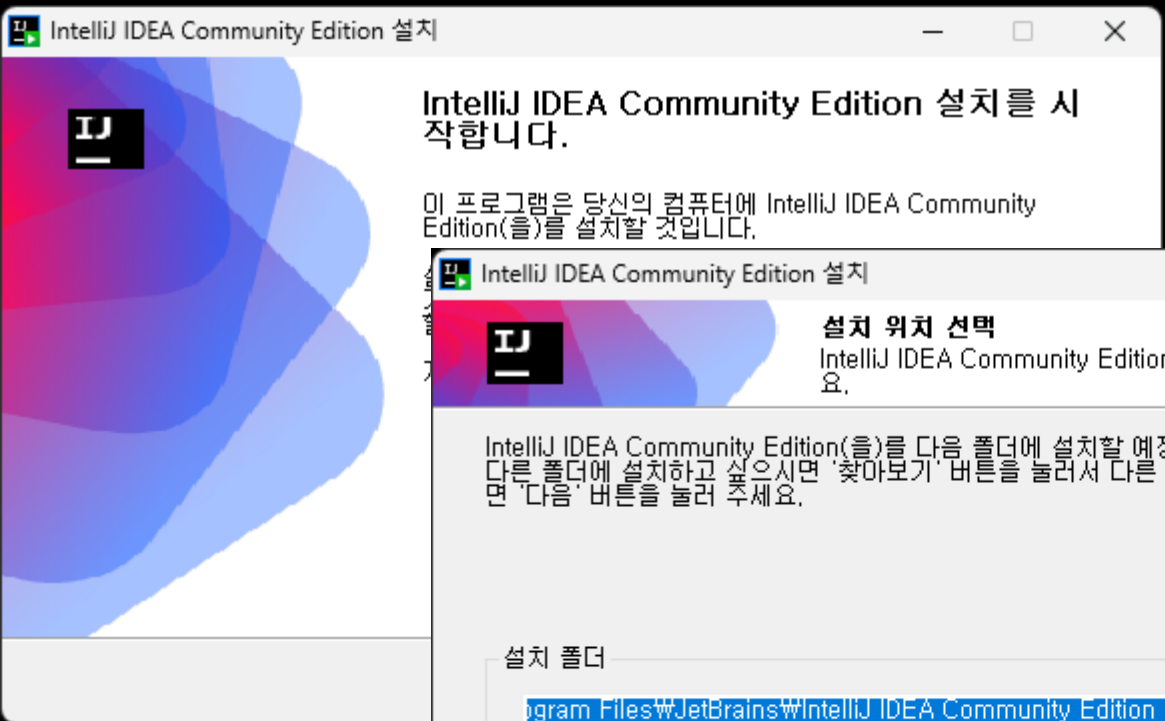
 **IntelliJ IDEA Community Edition**

Java 및 Kotlin 애호가를 위한 IDE

다운로드


.exe (Windows) ▼

무료, 오픈 소스로 빌드됨



환경변수 확인

← 설정





최호성


cx8537@outlook.com


설정 검색


Q


 홈


 시스템


 Bluetooth 및 장치


 네트워크 및 인터넷

 개인 설정

 앱

 계정

 시간 및 언어

 게임

시스템 > 정보

① 장치 사양

장치 이름

프로세서 Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.6

설치된 RAM 64.0GB

장치 ID

제품 ID

시스템 종류 64비트 운영 체제, x64 기반 프로세서


펜 및 터치 펜 지원

관련 링크

도메인 또는 작업 영역

시스템 보호

고급 시스템 설정

 Windows 사양

에디션 Windows 11 Pro for Workstations

버전 23H2

시스템 속성

컴퓨터 이름 하드웨어 고급 시스템 보호 원격

이 내용을 변경하려면 관리자로 로그인해야 합니다.

성능

시각 효과, 프로세서 일정, 메모리 사용 및 가상 메모리

설정(S)...

사용자 프로필

사용자 로그인에 관련된 바탕 화면 설정

설정(E)...

시작 및 복구

시스템 시작, 시스템 오류 및 디버깅 정보

설정(T)...

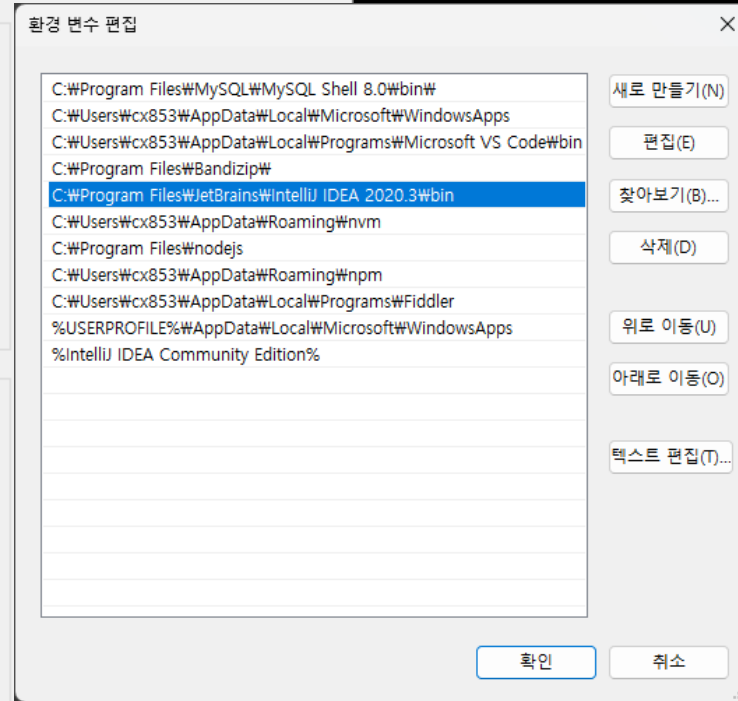
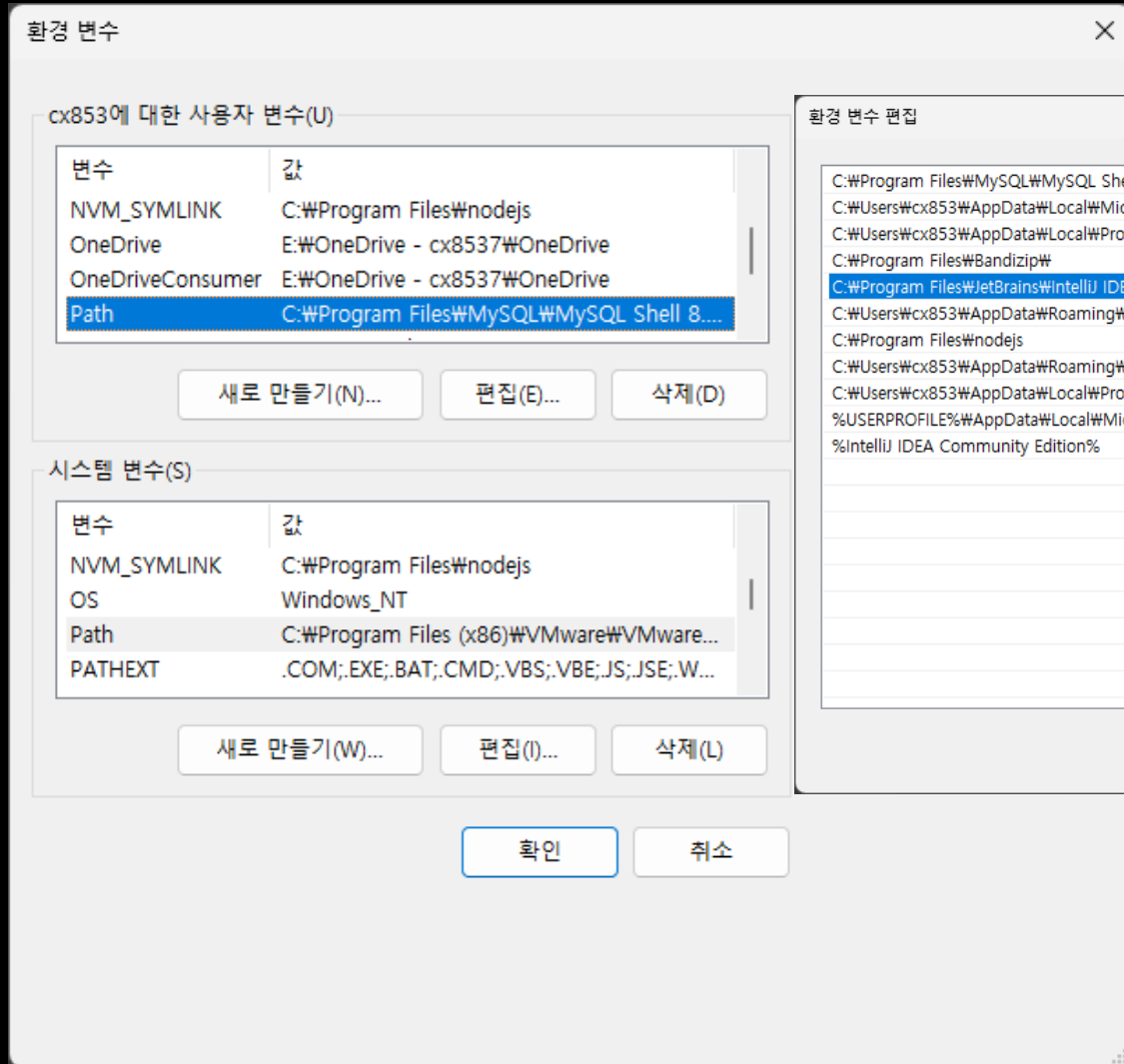
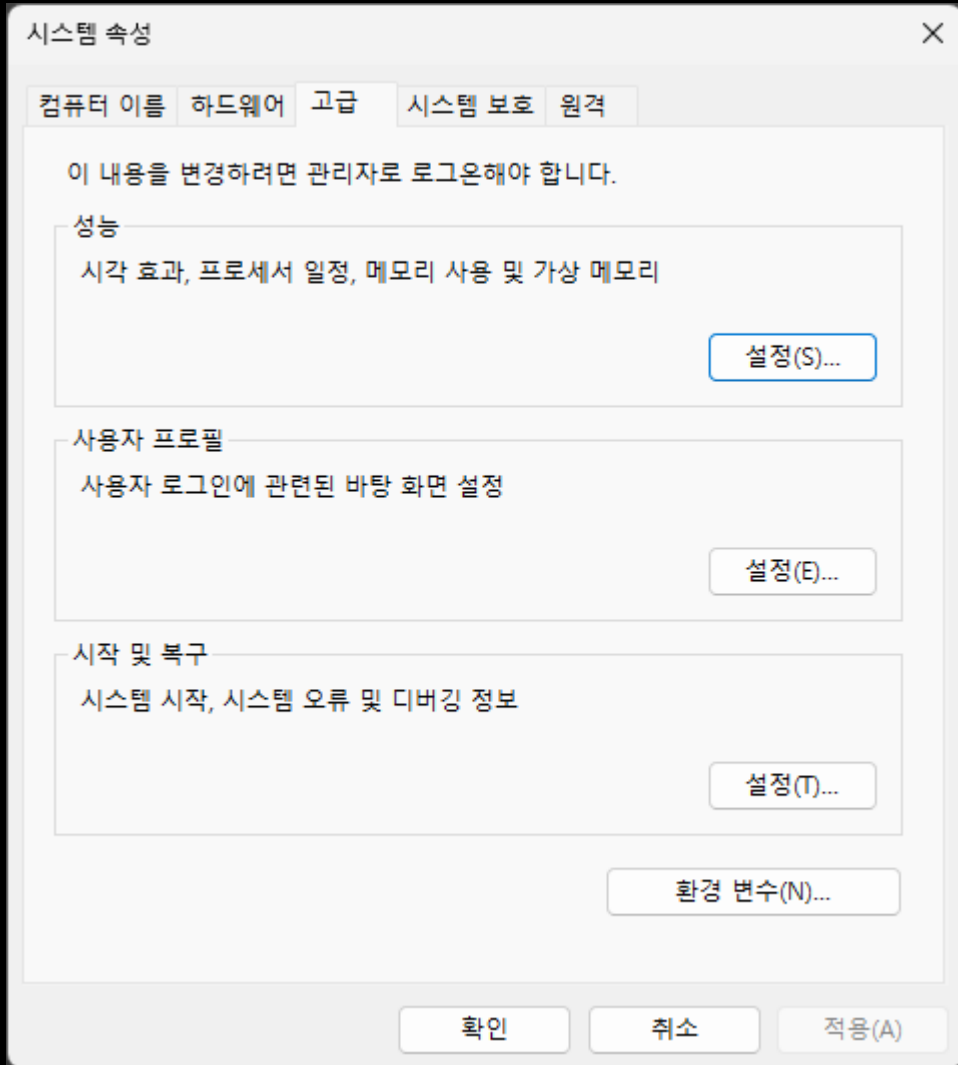
환경 변수(N)...

확인

취소

적용(A)

환경변수 편집



IntelliJ 첫 실행

언어 및 지역

라이선싱, JetBrains Marketplace 및 다른 특정 지역에 적용되는 기능과 링크가 올바르게 동작하도록 만들기 위해 지역을 선택하세요. 자세한 정보는 [문서](#)를 참조하세요

Korean 한국어 ▾

지역이 지정되지 않았습니다 ▾

다음

IntelliJ IDEA 사용자 계약

JETBRAINS COMMUNITY EDITION TERMS

IMPORTANT! READ CAREFULLY:
THESE TERMS APPLY TO THE JETBRAINS INTEGRATED DEVELOPMENT ENVIRONMENT TOOLS CALLED 'INTELLIJ IDEA COMMUNITY EDITION' AND 'PYCHARM COMMUNITY EDITION' (SUCH TOOLS, "COMMUNITY EDITION" PRODUCTS) WHICH CONSIST OF 1) OPEN SOURCE SOFTWARE SUBJECT TO THE APACHE 2.0 LICENSE (AVAILABLE HERE: <https://www.apache.org/licenses/LICENSE-2.0>), AND 2) JETBRAINS PROPRIETARY SOFTWARE PLUGINS PROVIDED IN FREE-OF-CHARGE VERSIONS WHICH ARE SUBJECT TO TERMS DETAILED HERE: <https://www.jetbrains.com/legal/community-bundled-plugins>.
"JetBrains" or "we" means JetBrains s.r.o., with its principal place of business at Na Hrebenech II 1718/10, Prague, 14000, Czech Republic, registered in the Commercial Register maintained by the Municipal Court of Prague, Section C, File 86211, ID No.:

☒ 본인은 본 사용자 계약 약관을 읽고 동의했음을 확인합니다.

종료

계속

IntelliJ IDEA

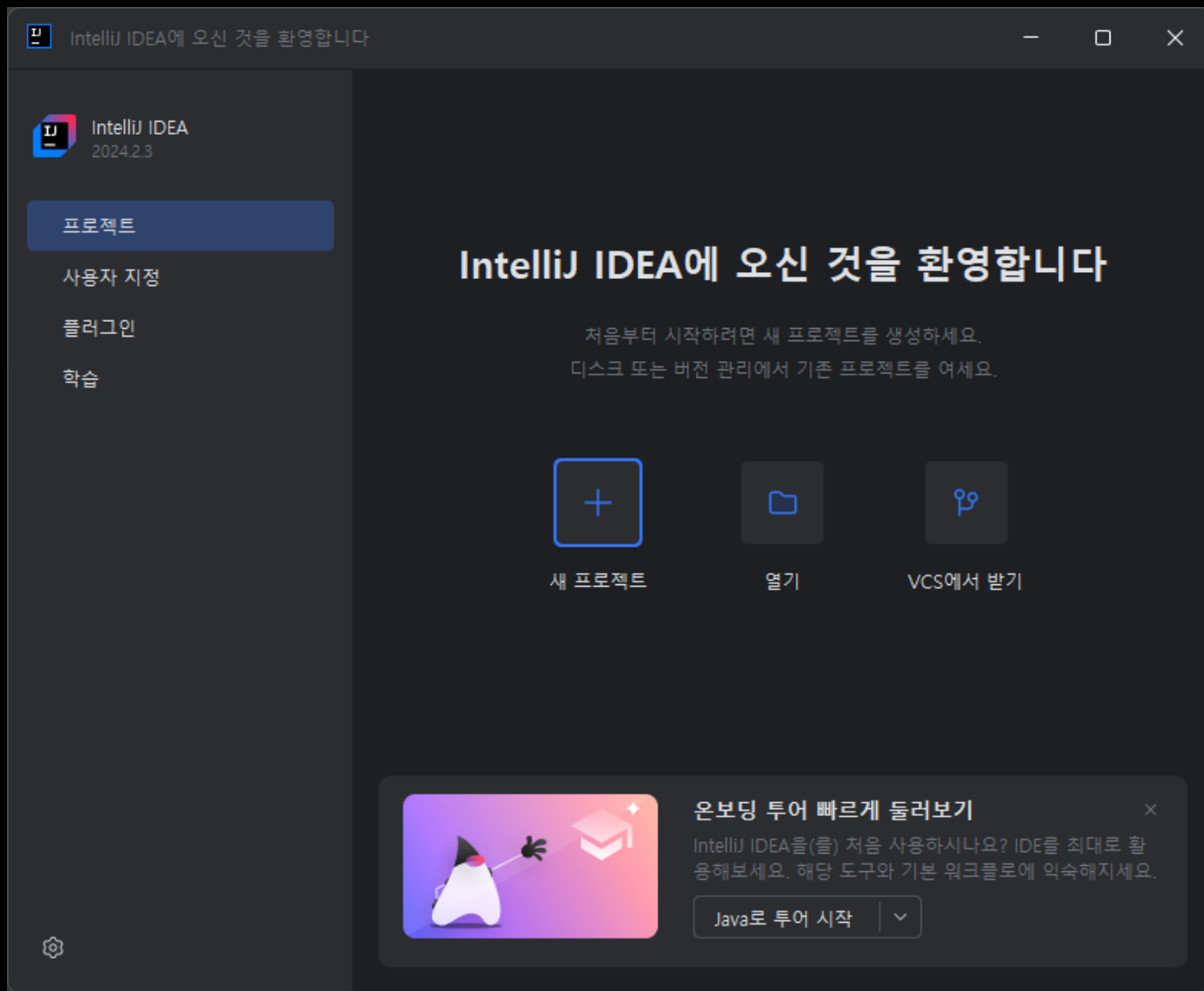
설정 가져오기

Visual Studio Code

가져오기 건너뛰기

기타 옵션 ▾

새 프로젝트 생성





신규 프로젝트

Java

Kotlin

Groovy

빈 프로젝트

제너레이터

Maven 원형

JavaFX

Spring

Compose for Desktop



플러그인을 통해 추가...

이름:

HelloWorld

위치:

F:\₩독하게 시작하는 Java - Part 1



프로젝트 이름(가) 다음에 생성됩니다. F:\₩독하게 시작하는 Java - Part 1\HelloWorld

☐ Git 저장소 생성

시스템 빌드:

IntelliJ

Maven

Gradle

JDK:

20 Oracle OpenJDK 20.0.1

☒ 샘플 코드 추가☒ 온보딩 팁이 포함된 코드 생성

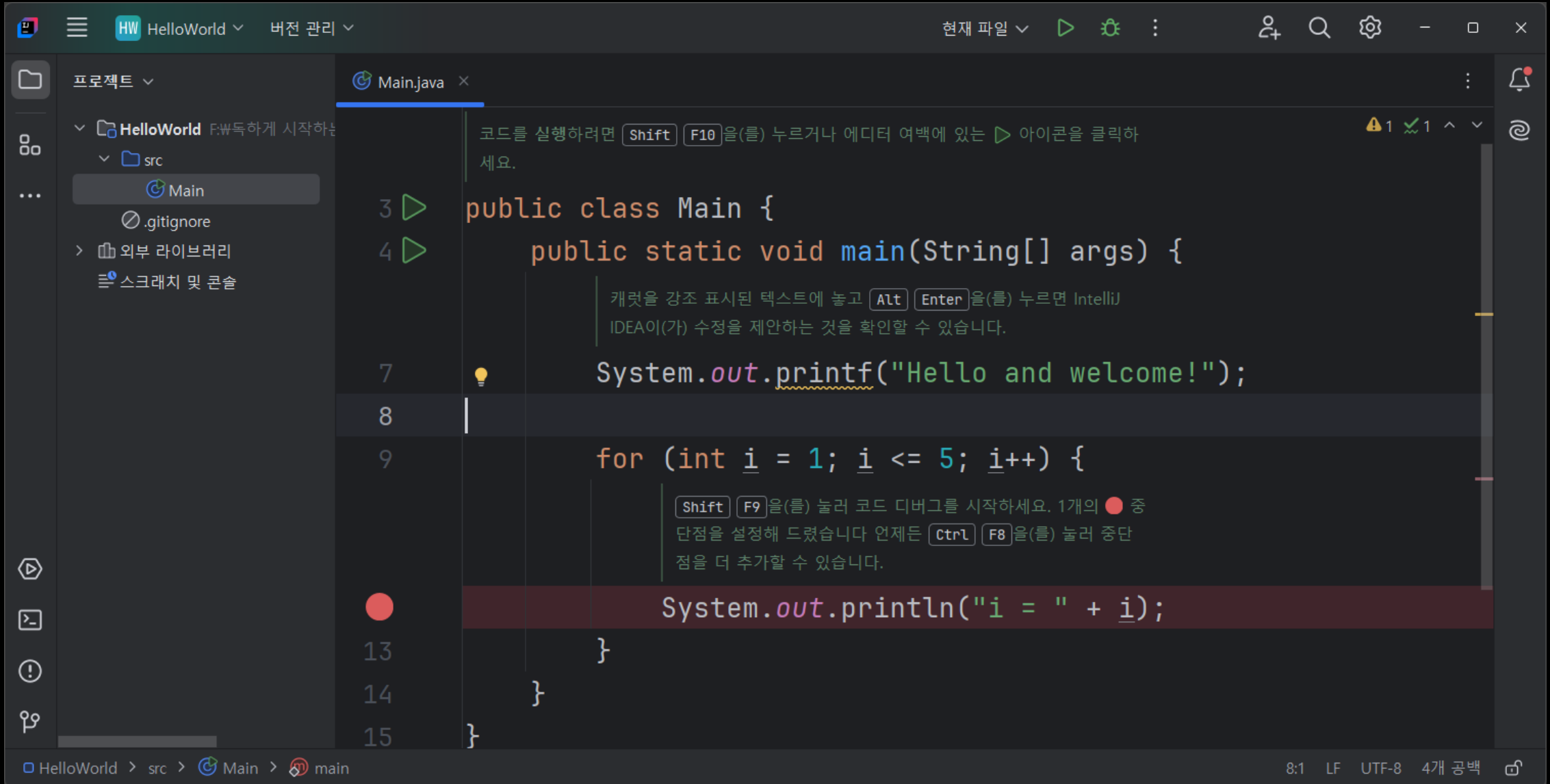
> 고급 설정

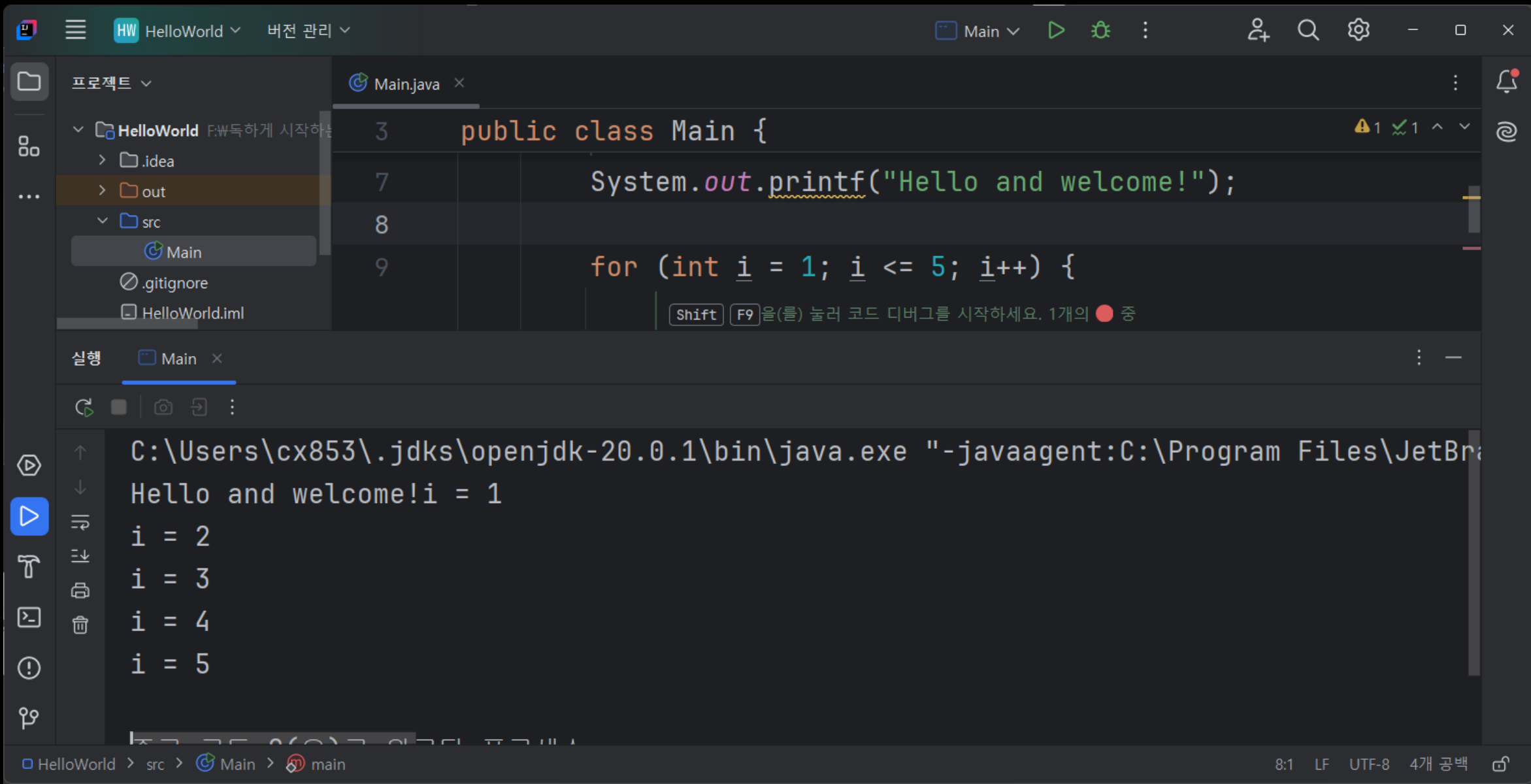


생성(C)

취소

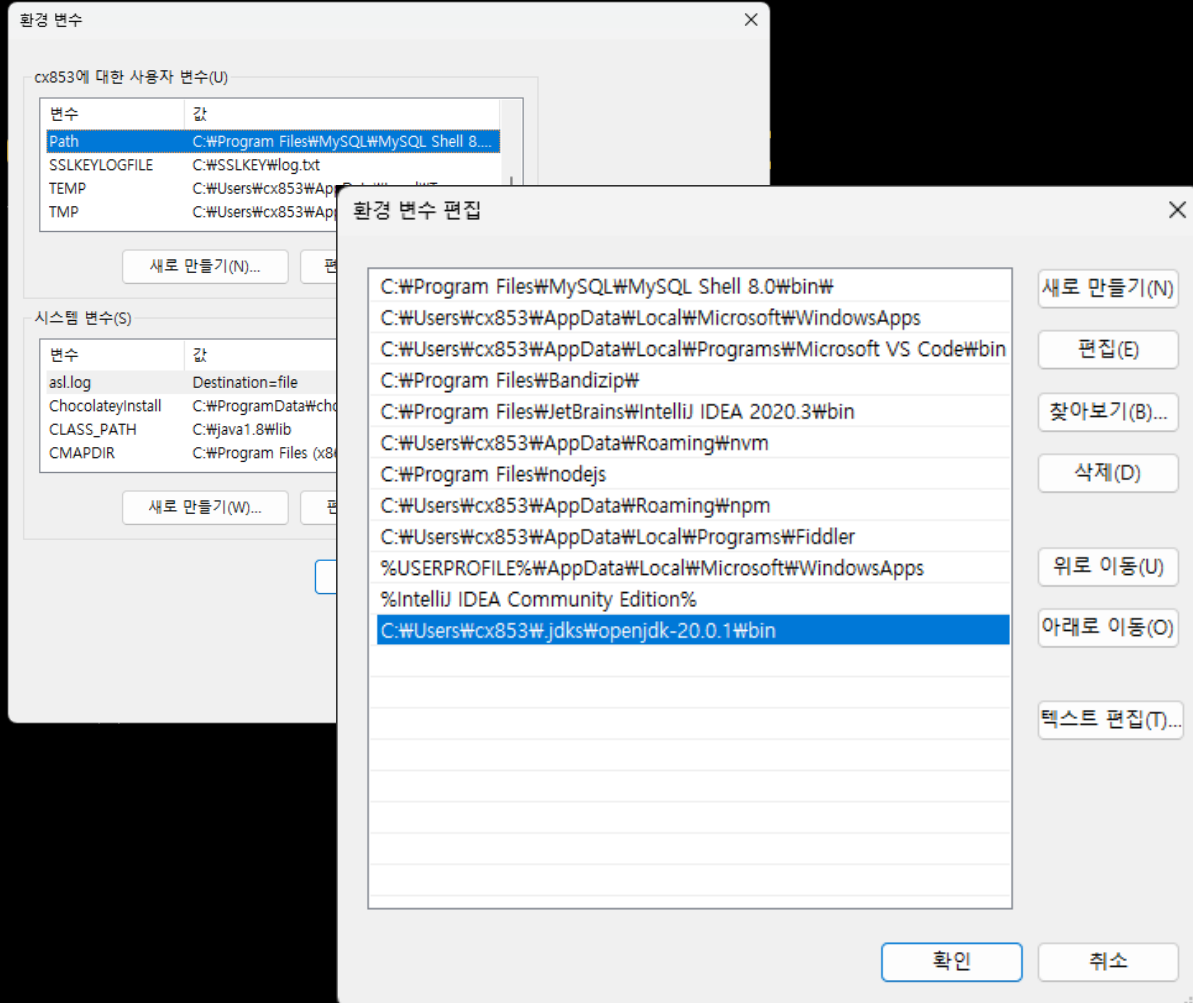
기본 제공 예제 실행





OpenJDK Path 추가

C:\Users\계정명\.jdk\openjdk-20.0.1\bin



- 사용자 환경 변수에 JDK 설치 위치를 Path 등록
- 확인 후 명령 프롬프트를 실행하고 버전 확인 (java -version)

IntelliJ 단축키

- Build Ctrl + F9
- Run Shift + F10, Ctrl + F5(마지막 빌드)
- Debug mode run Shift + F9
- Break point Ctrl + F8
- 디버깅 강제 중단 Ctrl + F2
- 프로그램 재개 F9
- Step over F8
- Step into F7

첫 번째 예제 작성

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

중요한 시점

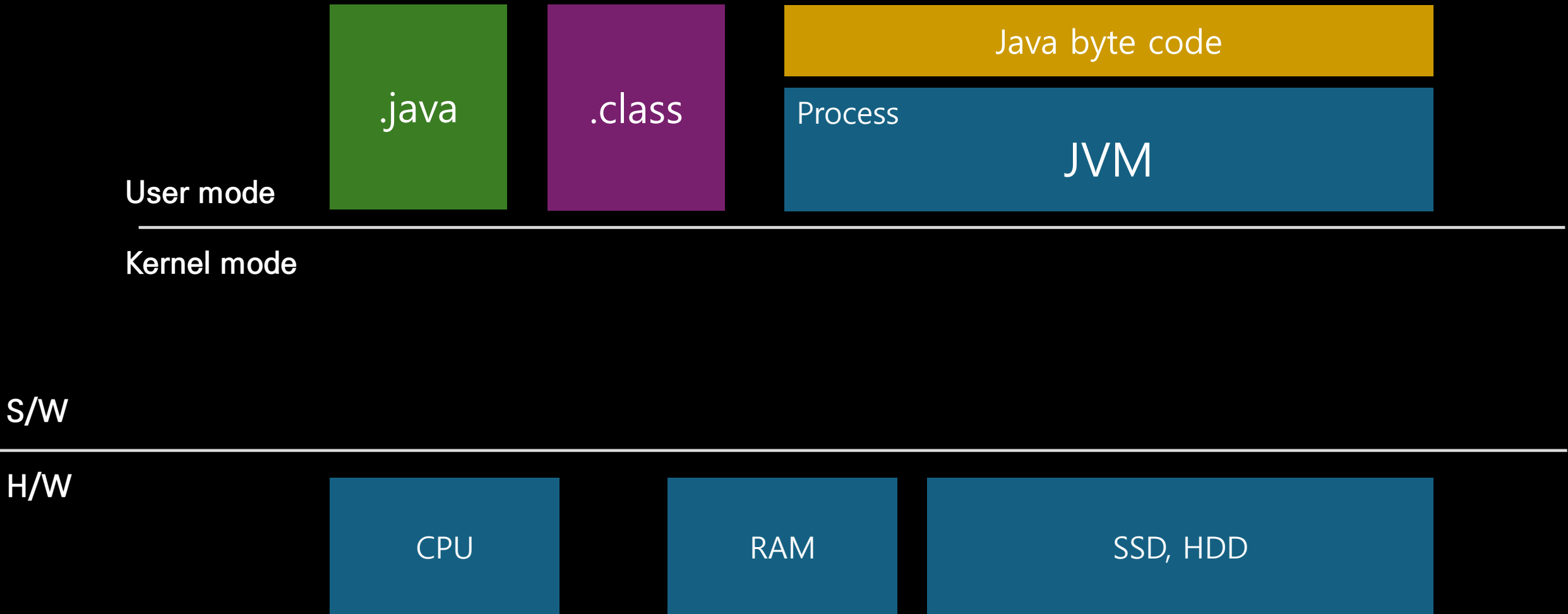
• 빌드 타임 (컴파일 타임 포함)

- .java 소스코드를 .class(Java byte code)로 번역
- 외부 라이브러리 사용 시 Symbolic references만 존재하며 정적 빌드 개념은 없음
- 외부 라이브러리(클래스 파일) 링크는 JVM에서 처리

• 런타임 (링크 타임 포함)

- JVM의 Class loader가 절차에 따라 Bytecode 로딩 후 메모리에 적재
- 추가 클래스 파일도 로드하고 링크

컴파일, 링크 시점



C/C++ 개발자를 위한 안내

Java

- .class 파일
- 참조 .class 파일
- Package
(모듈 수준)
- JVM class loader
- 참조
- Stack, Heap

C/C++

- .obj 및 .exe (PE 파일)
- .lib, .dll (PE 파일)
- namespace
(코드 수준)
- Linker
- 포인터, 참조
- Stack, Heap

C/C++ 개발자를 위한 안내

- 디폴트 복사 생성자 없음
- Java의 참조 구조는 C++의 구조와 거의 동일하며 Deep copy 이슈가 동일하게 있음
- Java에서 문자열 상수는 String 클래스 객체
- import문은 #include 전처리기와 유사 (단, Java는 전처리가 없음)
- Java는 전역변수 개념이 없음
- 정적 메서드에서 다른 정적 멤버 접근 시 멤버접근 연산이 필요 없음
- 범위지정연산(::)과 멤버접근연산(.)을 함께 사용

C/C++ 개발자를 위한 안내

- 정적 메서드에서 다른 정적 멤버 접근 시 멤버접근 연산이 필요 없음
- C++에서는 선언과 정의를 각각 .h와 .cpp로 명확히 구분하지만 Java는 그렇지 않음
- 클래스 이름과 파일명이 대/소문자까지 맞아야 함
- 멤버 데이터 대신 필드
- 참조형 필드의 경우 class 선언 코드에서 new 연산 초기화 가능

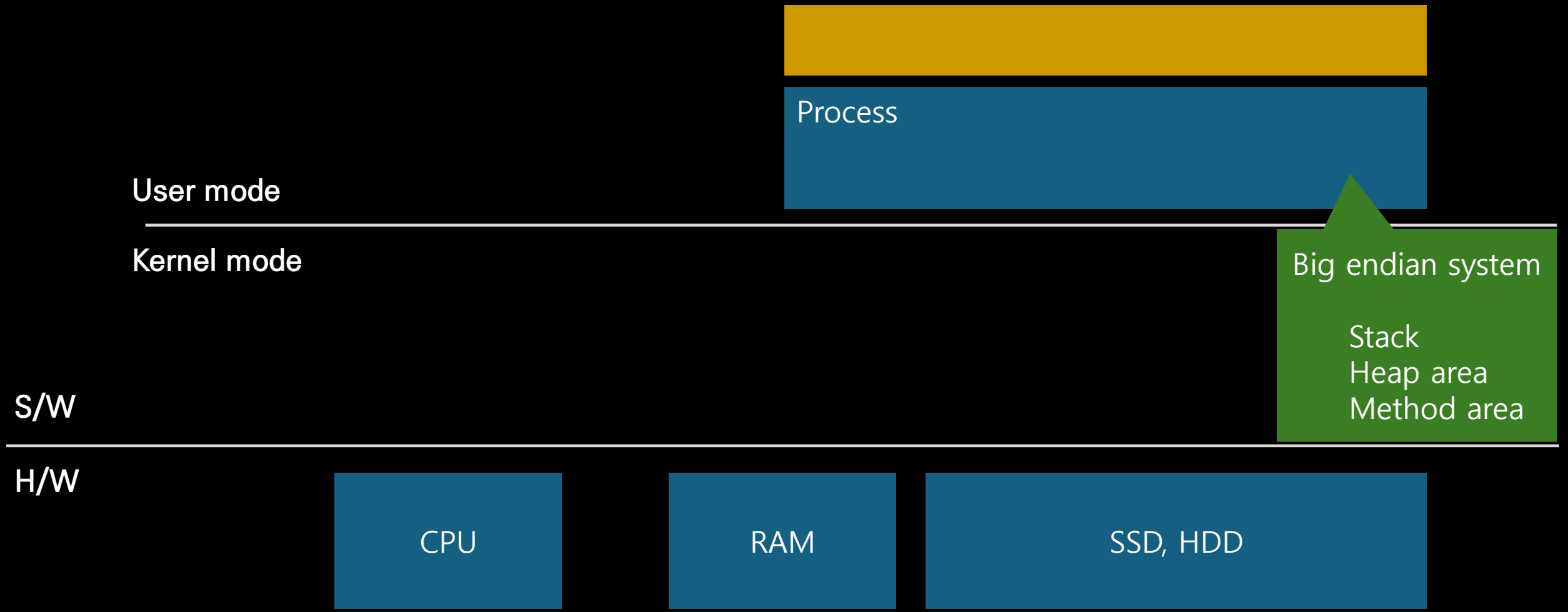
6. 독하게 시작하기

본격적인 프로그래밍에 앞서 개발자라면 반드시 알아야 할 필수 이론 정리

Java 특징

- JVM 기반에서 작동하는 OOP언어
- C/C++의 가장 큰 특징인 메모리 관리와 책임이슈를 구조적으로 제거
- OS(혹은 Platform)에 대한 의존성 없음
- 컴파일러, 인터프리터 특징을 모두 가짐
 - 하이브리드
- 문법적으로 C/C++와 매우 유사

User mode process JVM



JVM 구성요소

Class loader

Loading
(Bootstrap/Extension/Application class loader)

Linking
Verify → Prepare → Resolve

Initialization

Runtime data area

Method area
(Runtime constant pool)

Heap area

Stack area

PC register

Native method stack

Execution engine

Interpreter

JIT compiler

Garbage collector

JNI,
Native method
interface

Native method
library

JIT(Just In Time) compiler

- Java bytecode를 실제 기계어로 번역
- JVM이 반복되는 코드를 발견할 경우 효율을 높일 목적으로 사용
 - Intermediate code generator
 - Code optimizer
 - Target code generator
 - Profiler (Hotspot)
- 실행 기록을 모아 자주 사용되는 코드에 주로 적용 (반복문)
- 프로그램을 오래 실행 할 수록 성능개선에 유리

하드웨어 수준 자료형

• 정수

- bit수에 따라 표현 범위 결정
- 2의 보수를 더하는 방식으로 뺄셈 구현

• IEEE 754 기반 실수

- 단정도 (float)
- 배정도 (double, 소수점 이하 15번째 자리까지 유효)
- 특수정도

실수형 표현 범위

float



double

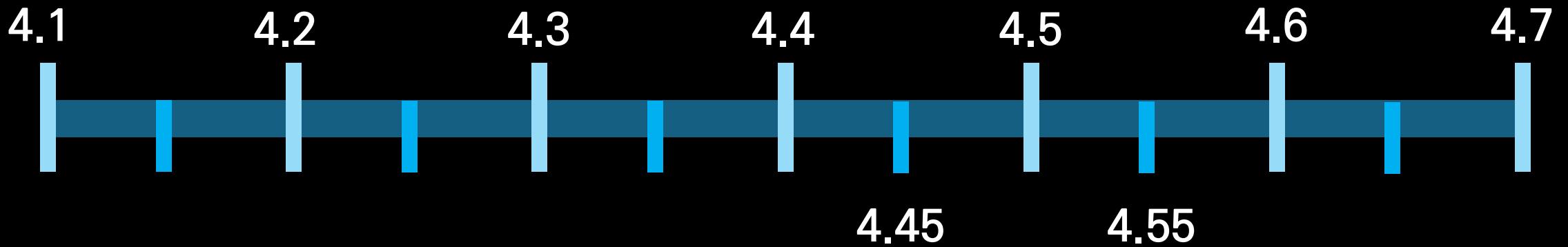


실수형

- IEEE(Institute of Electrical and Electronics Engineers, 전기전자 기술자협회)가 규정한 표준사용
 - IEEE 754 표준
- 소수점 이하 정보를 표시할 수 형식
- 부동 소수점 표현
 - 100.0 , $10.0 * 10$, $1.0 * 10^2$ 은 같은 값에 대한 표현
- 두 정수 사이에는 무수히 많은 실수가 존재하기 때문에 일정 수준이 오류(부동소수점 오차)를 인정함

부동소수점 오차

4.999998은 5.0과 같다고 할 수 있나? (면접질문)



실수형 – 부동소수점 오차 (06_floatError)

[illegible]

실행

Main x



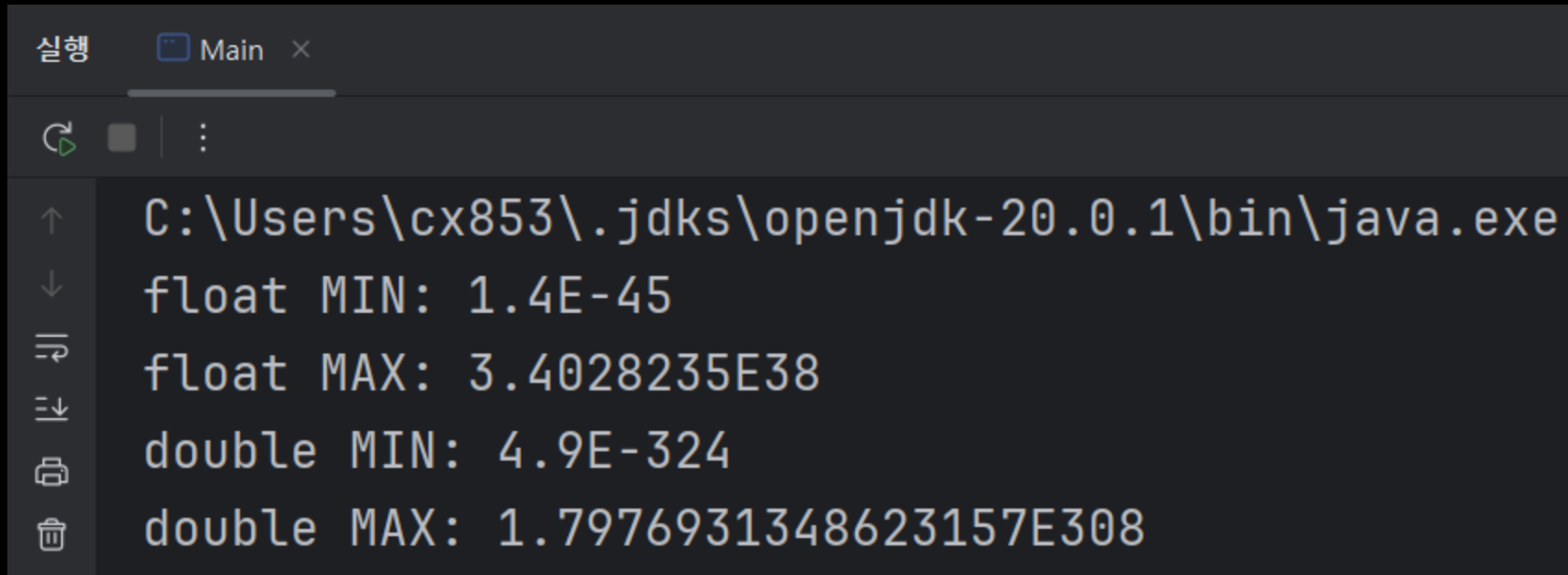
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe

4.999998

종료 코드 0(으)로 완료된 프로세스

Java 실수형 값 범위 (06_floatRange)

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("float MIN: " + Float.MIN_VALUE);  
        System.out.println("float MAX: " + Float.MAX_VALUE);  
        System.out.println("double MIN: " + Double.MIN_VALUE);  
        System.out.println("double MAX: " + Double.MAX_VALUE);  
    }  
}
```





The screenshot shows a Java IDE with a tab labeled 'Main'. The output window displays the following text:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
float MIN: 1.4E-45  
float MAX: 3.4028235E38  
double MIN: 4.9E-324  
double MAX: 1.7976931348623157E308
```

부동소수점 오류에 따른 치명적 결함 사례

https://www.chosun.com/site/data/html_dir/2011/07/25/2011072500100.html

朝鮮日報



[오피니언](#)[정치](#)[사회](#)[국제](#)[건강](#)[재테크](#)[스포츠](#)[문화·연예](#)[쇼핑의맛](#)

[사회 > 신문은 선생님](#)

나이스, 프로그램에 문제... 쓰레기값 (컴퓨터 연산과정서 드물게 나오는 엉뚱한 값) 처리 누락

개발 맡은 삼성SDS 프로그래머 실수로 판명
고교 내신 동점자 처리 문제
원래 소수점 16자리까지 표시
일부선 평가와 다른 숫자 나와

김연주 기자
입력 2011.07.25. 03:02

가

대입 수시모집(8월1일부터)을 일주일가량 앞두고 고교생 약 3만명의 내신성적 석차가 잘못 배포된 차세대 나이스(NEIS·교육행정정보시스템)의 오류와 관련, "차세대 나이스 프로그램 개발 업체인 삼성SDS 프로그래머들의 실수가 있었다"고 한국교육학술정보원 관계자가 24일 밝혔다. 차세대 나이스를 관리하는 정부 산하기관인 한국교육학술정보원 신명호 교사지원부장은 이날 본지와 통화에서 "나이스에서 성적과 관련된 프로그램은 삼성SDS의 프로그래머 7명이 맡아 제작했는데, 프로그램을 정밀하게 짜지 못했다"고 말했다.

...중략...

신 부장은 "원래 동점자에게 일정 기준을 적용해 등수를 가릴 경우 **소수점 이하 16자리까지 표시를 하도록** 프로그램이 짜여 있는데, 소수점 이하 자릿수 가운데 **평가 결과와 상관없는 '1'이란 숫자가 느닷없이 표시**되는 현상이 발생해 고교생 **2만9000명의 내신석차가 틀렸다**"고 말했다.

통상 컴퓨터가 숫자를 처리하는 과정에선 이처럼 엉뚱한 수치(이른바 '쓰레기값')가 나오는 경우가 드물게 발생하는데 이를 잡아주는 작업을 실수로 빠뜨렸다는 것이다. 신 부장은 "쓰레기값은 소수점 이하 16개 자릿수 중 일정한 자리에서 발생한 것이 아니라 불규칙적으로 여러 자리에서 나왔다"고 했다.

Java 자료형 – 기본형(Primitive type)

- 정수

- byte, short, char, int, long

- 실수

- float, double

- 논리

- boolean(true, false)

Java 자료형 – 기본형(Primitive type)

자료형	크기 (bit)	표현범위	설명
byte	8	$-2^7 \sim (2^7 - 1)$	0 표현은 양수범위에 포함
short	16	$-2^{15} \sim (2^{15} - 1)$	–
char	16	$0 \sim (2^{16} - 1)$	0~65535 범위 유니코드
int	32	$-2^{31} \sim (2^{31} - 1)$	–
long	64	$-2^{63} \sim (2^{63} - 1)$	–
float	32	$1.4 \times 10^{-45} \sim (3.4 \times 10^{38})$	유효범위: 소수점 이하 7자리
double	64	$4.9 \times 10^{-324} \sim (1.8 \times 10^{308})$	유효범위: 소수점 이하 15자리

유도형(Non-primitive or Derived type)

- Java에서 객체로 언급하는 대상
- String
 - 문자열을 다루기 위한 클래스
- Class
 - Array, List, Queue, Stack
 - Interface

그 외 형식

- 무치형

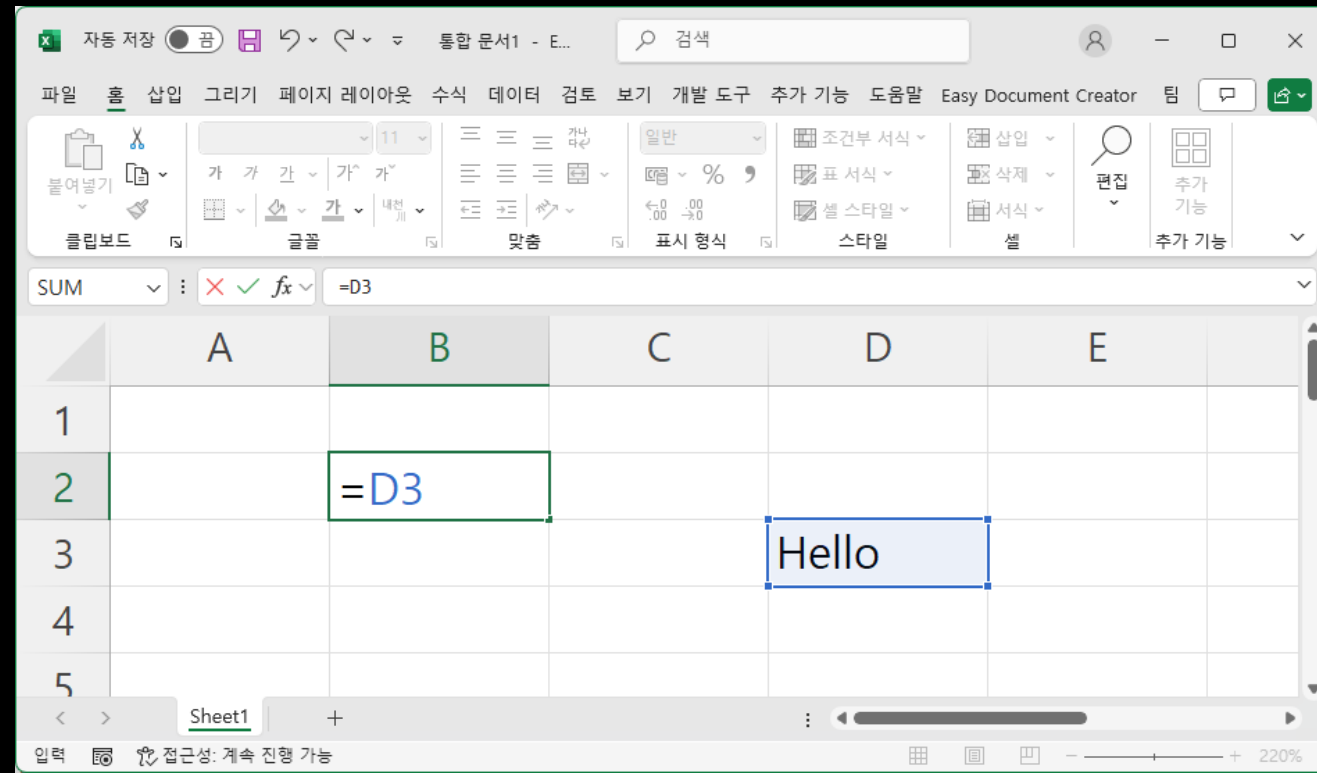
- void

- 함수형

- testFunc(int a)

개념부터 챙기는 '참조'

- 객체와 자료형
- 기본 자료형
- 참조 자료형 및 사용자 정의 형식

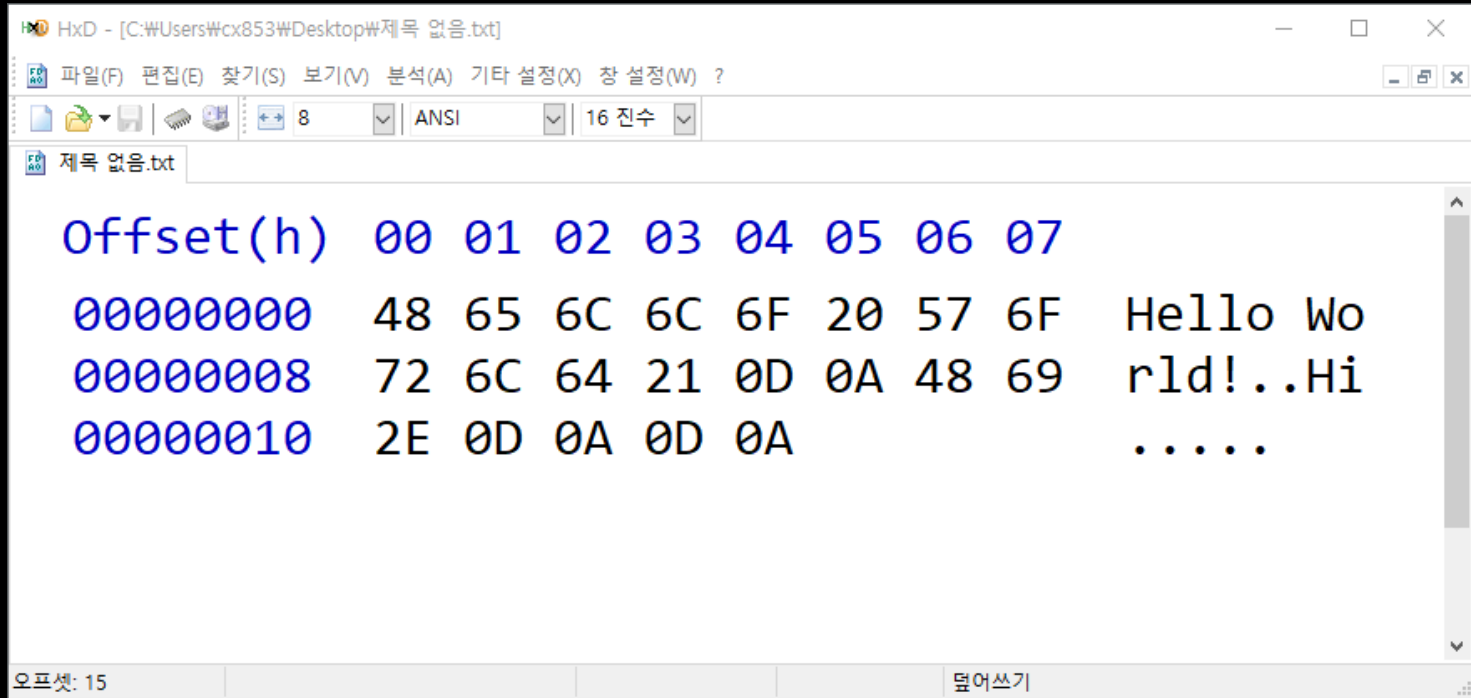
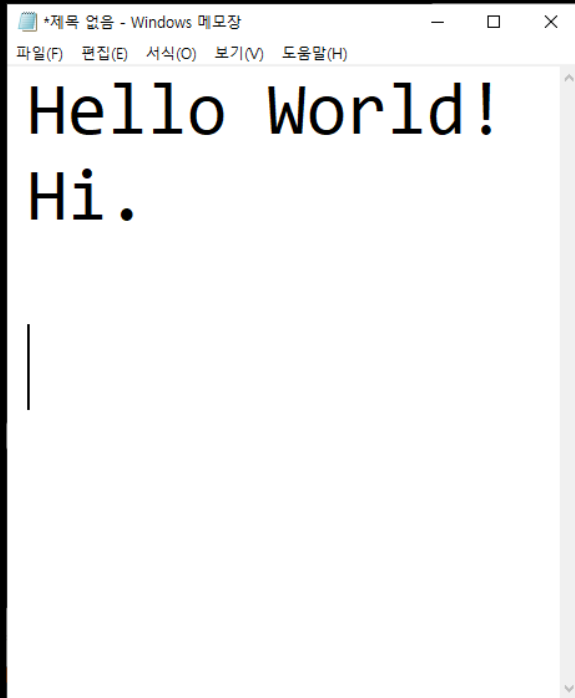


객체, 클래스, 인스턴스

- 객체(Object)란 특정 목적을 가진 코드와 연산에 필요한 자료(변수)들을 한 세트로 묶어 구현할 대상
- Class는 객체를 기술하기 위한 문법으로 새로운 자료형이 될 수 있음
- 인스턴스는 특정 자료형에 대한 변수
 - `int a;`
(a는 int 형식에 대한 인스턴스)
 - `String hello = new String("Hello");`
(hello는 새로 생성된 String 클래스 인스턴스에 대한 참조)

문자와 문자열

- 영문 혹은 한글 한 글자를 '문자'로 규정
- 문자를 연이어진 배열 형태로 나열하면 '문자(배)열'



문자, 문자(배)열

- char형은 **한 문자(유니코드)**를 저장하기 위한 자료형
 - `char ch = 'A';`
- char형은 정수 형식에 속함
 - `'A' + 1;` 연산 가능
- char형 자료가 여럿 **연이어진 배열 형태**가 '문자열'
 - `char[] hello = {'H', 'e', 'l', 'l', 'o'}`

Unicode와 인코딩 규칙

- C언어의 문자열은 MBCS(Multi-bytes Character Sets)과 Unicode 문자열로 나눔
 - Java에서 문자열의 끝은 NULL이 아님
- 유니코드는 한글 영문 모두 한 글자가 2바이트
- Java는 기본적으로 UTF-16 BE 체계와 Modified UTF-8 체계 병행

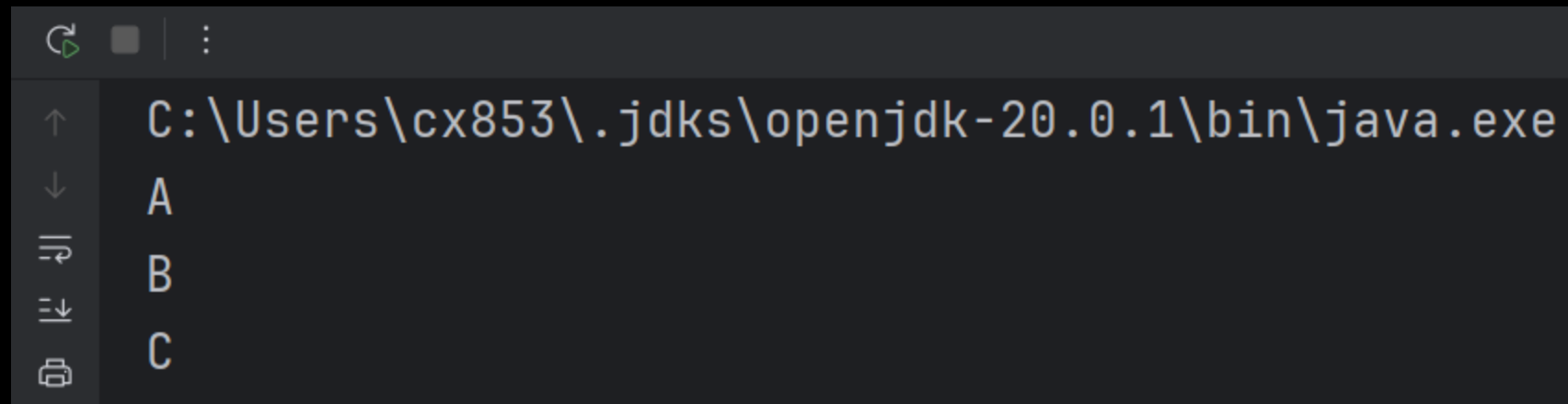
프로그래밍을 배우려는 사람들을 위해...

**문자를 다루는 인코딩 규칙에
대한 모든 것! (※매우 중요)**

**UTF-8, UTF-8 BOM, UTF-16,
Unicode, 한글 완성형/조합형**

문자 출력 (06_charInt)

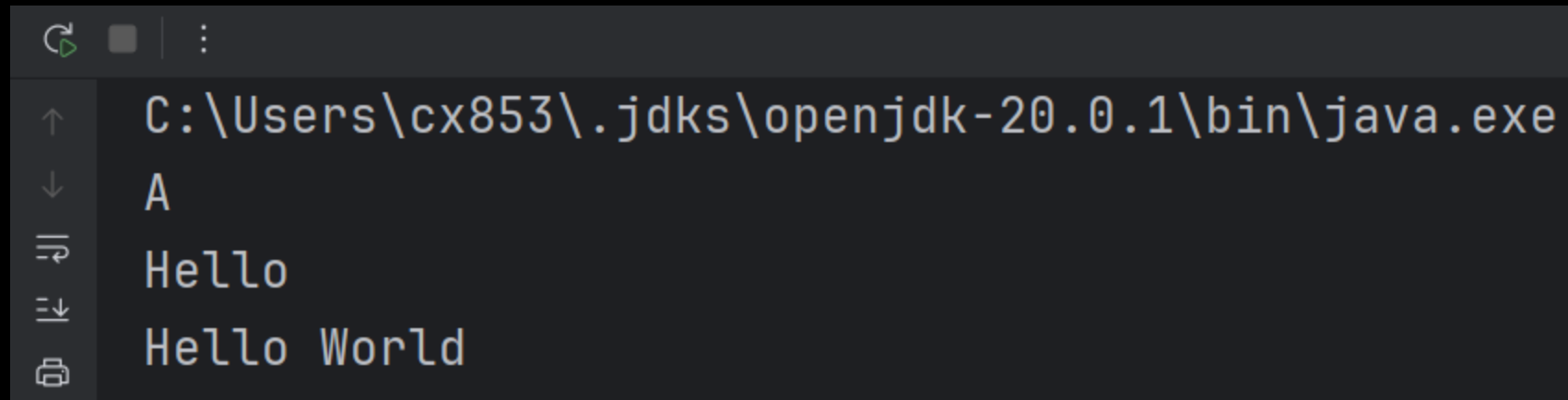
```
public class Main {  
    public static void main(String[] args) {  
        System.out.println('A');  
        System.out.println((char)(65 + 1));  
        System.out.println((char)('A' + 2));  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
A  
B  
C
```

문자열 출력 (06_stringSample)

```
public class Main {  
    public static void main(String[] args) {  
        char ch = 'A';  
        System.out.println(ch);  
        char[] hello = {'H', 'e', 'l', 'l', 'o'};  
        System.out.println(hello);  
        String strHello = "Hello";  
        System.out.println(strHello + " World");  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
A  
Hello  
Hello World
```

7. 변수와 주석문

변수

- 연산식을 기술하는 시점에 값이 정해지지 않은 수
- 구체화하지 않았거나 앞으로 변경될 가능성이 있는 수 (혹은 미지의 수)
- 개발자가 메모리를 사용하는 가장 일반적인 방법
- 구체적으로 결정되는 값에 따라 연산의 내용이 달라질 수 있는 원인으로 작용
 - 예) 날씨변수 (맑음, 흐림, 비, 눈, 온도)
 - 의존성(Dependency)의 시작

상수

- 연산식을 기술하는 시점에 값이 정해진 수
- 값이 확정되어 앞으로 변할 가능성이 없는 수
- 리터럴 (상수)
 - ‘A’, “Hello”, 3, 3.4F, 123.45
- 심볼릭 상수
 - final

변수 이름(식별자, Identifier) 작성 규칙

- 이름의 다른 표현은 식별자
- 영문 대/소문자, '_', 숫자 가능
 - int data, int _data, int _data1
 - int \$data, int #data (X)
- 첫 글자는 숫자 사용 불가 (식으로 인식)
 - int 1data (X)
- 이름 중간에 공백 문자 사용 불가
 - int data length (X)

변수 이름(식별자, Identifier) 작성 규칙

- 예약어 사용 불가

- int for (X)

- 너무 긴 이름 금지 (권장사항)

- int multibytestringtowidestring;

- 카멜 표기법 권장 (int multiByteStringToWideString)

- 의미를 알 수 없는 이름

- int aaa, bbb, ccc, ddd;

- 6개월 후 당신을 야근하게 만드는 가장 쉬운 방법

변수 종류 및 사용

- 지역변수

- 접근성에 따른 분류
- static 선언이 없다면 자동변수이며 Stack 사용

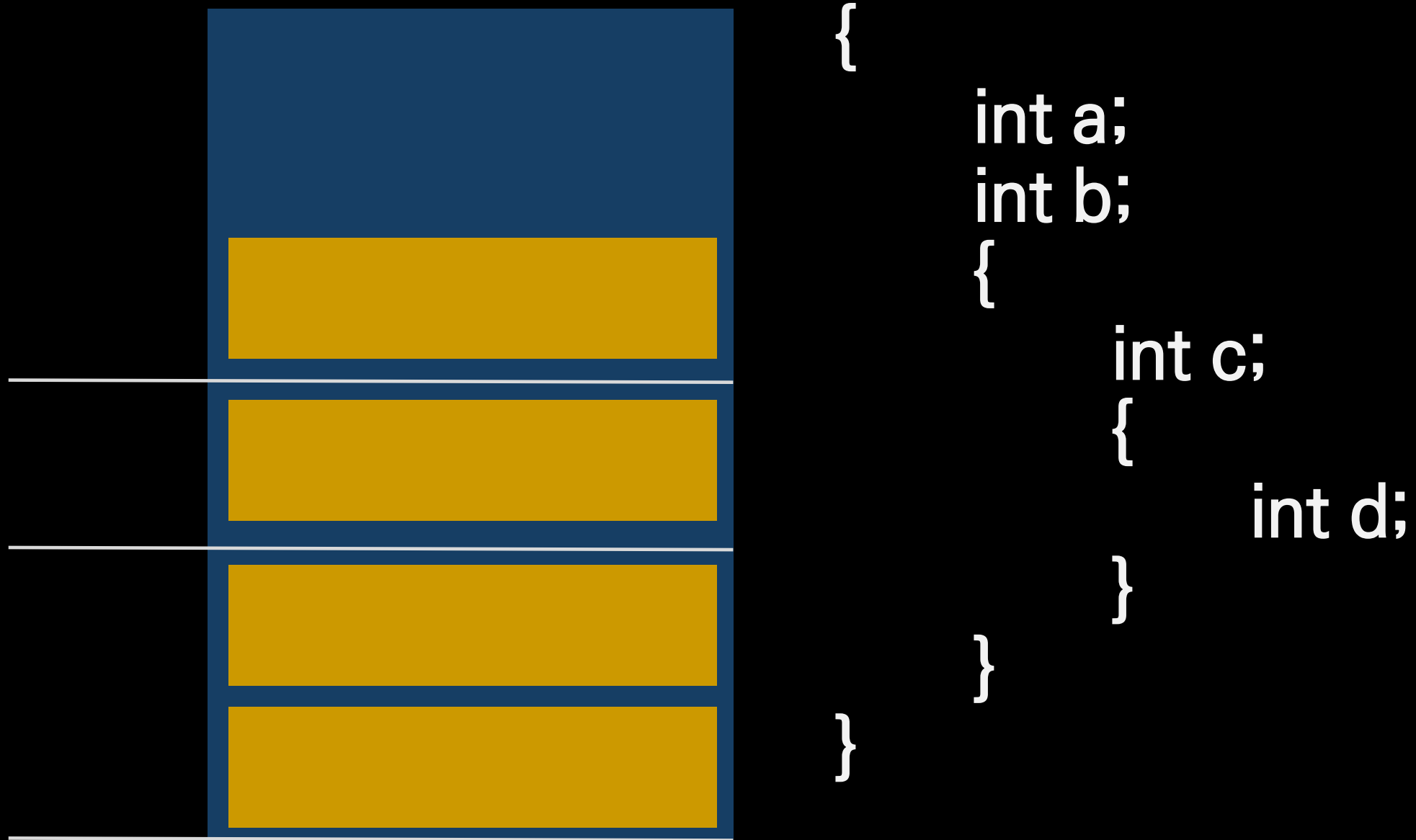
- 매개변수

- 함수 매개변수로 접근성은 지역

- 인스턴스 변수(멤버 변수)

- 클래스 변수

자동변수와 스택



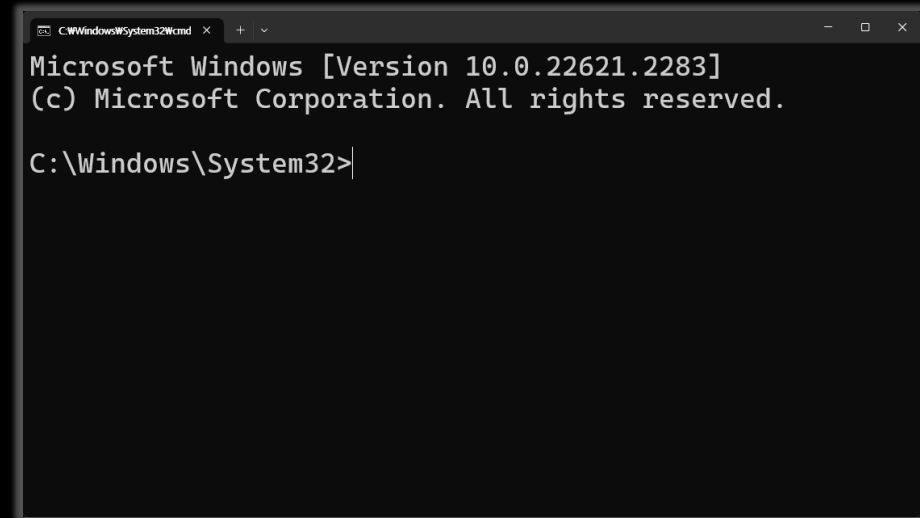
코드와 주석

- 프로그램 코드에 **메모**를 남기는 문법
 - 유지보수성 극대화
- 프로그램 코드에 포함되지 않음
 - 코드와 주석은 반드시 동기화 할 것!
- **//** (한 행 전체)
- **/* */** (구간 전체, 여러 행 가능)
- **Ctrl + /** (한 줄 주석, 토글)
- **Ctrl + Shift + /** (구간 주석, 토글)

8. 콘솔 입/출력

Console과 CLI

- CLI(Command Line Interface) 기반 HCI(Human Computer Interface)는 키보드 입력으로 구현
- 키보드 입력 시 그 값은 메모리(I/O Buffer)에 연속적으로 저장
- I/O Buffer에서 글자(혹은 키코드) 단위로 처리



```
C:\Windows\System32>Microsoft Windows [Version 10.0.22621.2283](c) Microsoft Corporation. All rights reserved.C:\Windows\System32>
```

Console



콘솔 입력 키코드 값 읽기

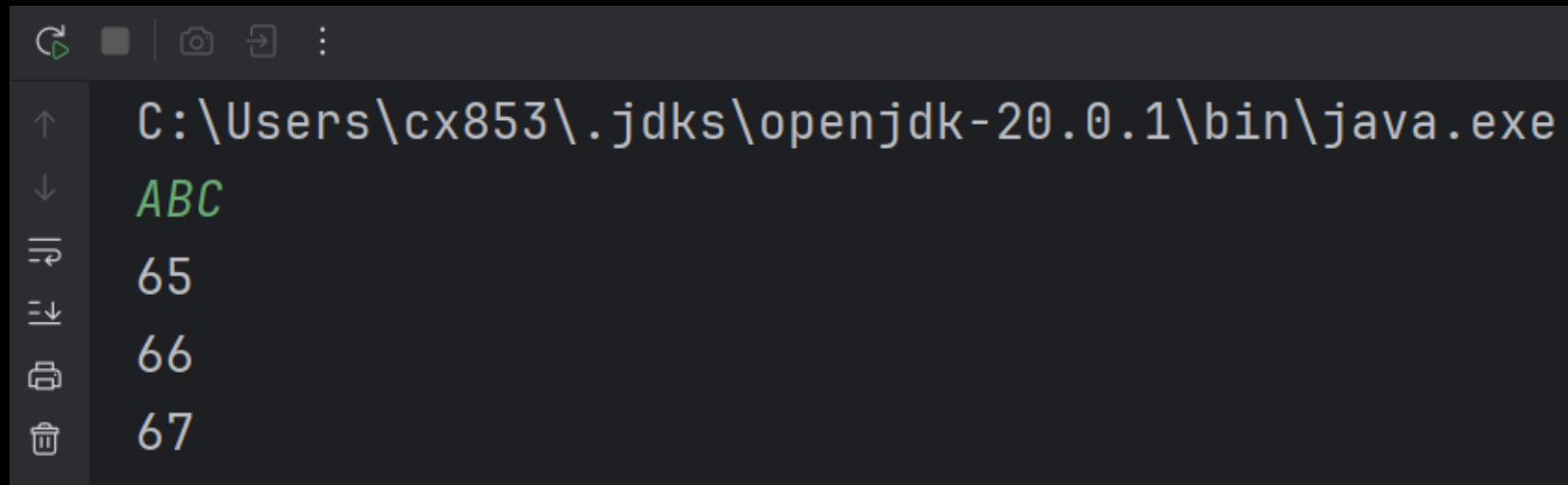
`System.in.read()`



- 키코드 값은 대부분 ASCII 코드 값과 일치 (Shift, Ctrl, Page Up/Down, 방향키)
- 1byte씩 읽어서 값을 반환
- 한글 한 글자는 2바이트

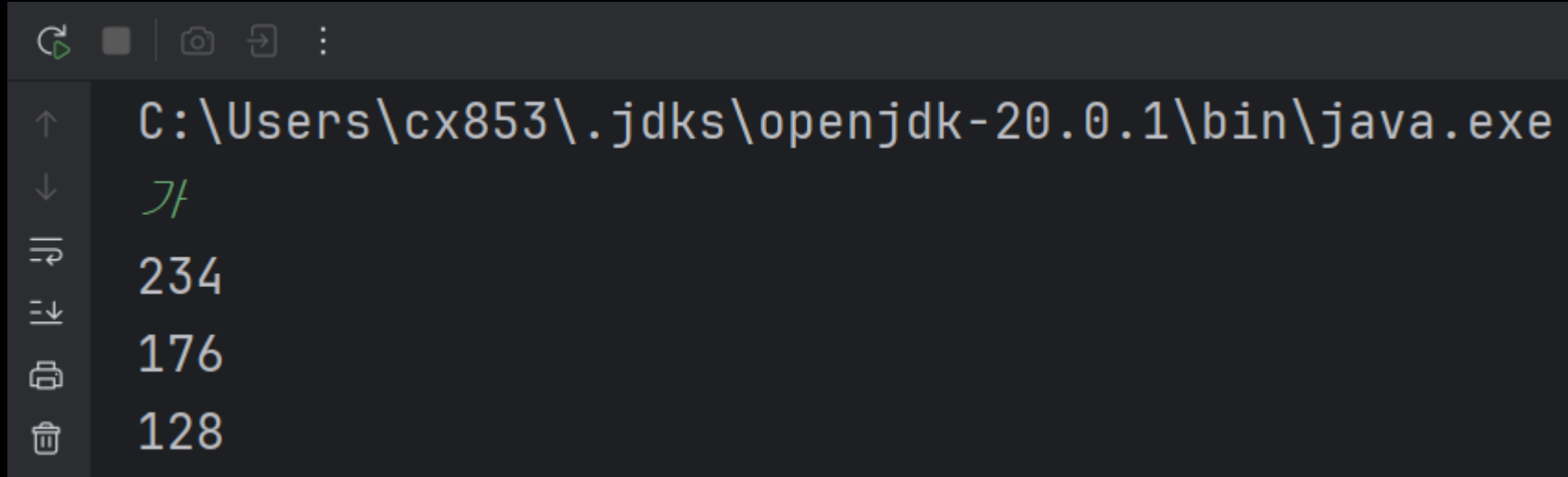
키코드 값 확인 및 출력 (08_keyCode)

```
public class Main {  
    public static void main(String[] args) throws Exception {  
        int keyCode = System.in.read();  
        System.out.println(keyCode);  
        keyCode = System.in.read();  
        System.out.println(keyCode);  
        keyCode = System.in.read();  
        System.out.println(keyCode);  
    }  
}
```



```
C:\Users\cx853\jdk\openjdk-20.0.1\bin\java.exe  
ABC  
65  
66  
67
```


08_keyCode 예제 한글 입력 결과



A screenshot of a Java IDE window. The title bar shows standard Windows icons. The main text area displays the file path `C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe`. Below the path, the Korean character '가' is entered. To the left of the character, a vertical toolbar contains icons for undo, redo, copy, paste, and delete. To the right of the character, the decimal values for the key codes are listed: 234, 176, and 128.

Icon	Value
Undo	234
Redo	176
Copy	128

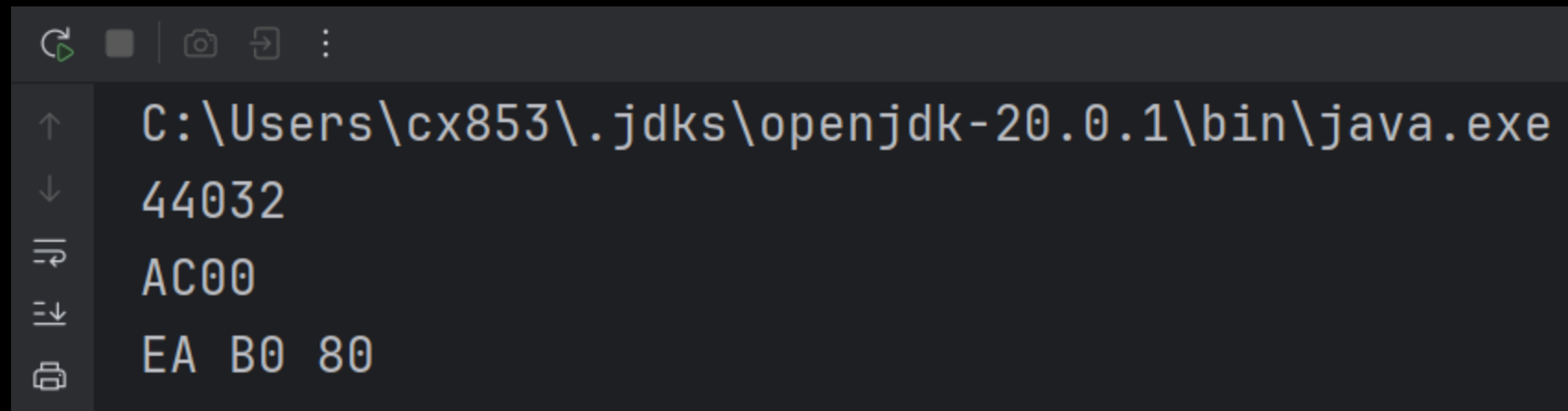
- 234(0xEA), 176(0xB0), 128(0x80)
 - ‘가’에 대한 UTF-8 인코딩 결과
 - UTF-8은 유니코드 값을 1~4바이트로 가변 인코딩
 - Java에서 문자열의 끝은 NULL이 아님(※C개발자 주의)

Java 인코딩 규칙

- Java 환경에서 문자열은 UTF-16 BE(Big Endian)로 인코딩
- 문자열 처리 과정에서는 UTF-16 BE를 Modified UTF-8로 변경해 처리
- 문자열의 끝인 **NULL**(0x0000)은 본래 UTF-8 규칙으로 인코딩 시 00이지만 Modified UTF-8로 인코딩할 경우 **0xC080**

UTF-8 문자열 확인 (08_keyCodeHangul)

```
public class Main {  
    public static void main(String[] args) throws Exception{  
        String ga = "가";  
        System.out.println((int)'가');  
        System.out.printf("%04X\n", (int)'가');  
        byte[] codeData = ga.getBytes();  
        System.out.printf("%02X %02X %02X\n",  
                            codeData[0], codeData[1], codeData[2]);  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
44032  
AC00  
EA B0 80
```

Scanner 클래스와 입력 버퍼 – 숫자 입력

`s.nextInt()`



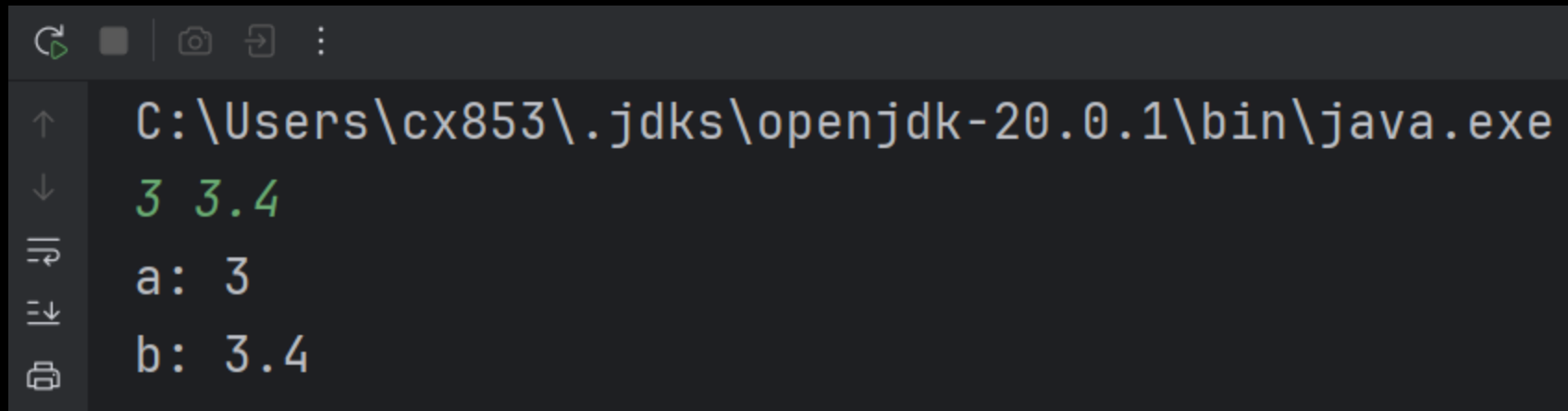
`s.nextDouble()`



정수, 실수 입/출력 (08_intDoubleInput)

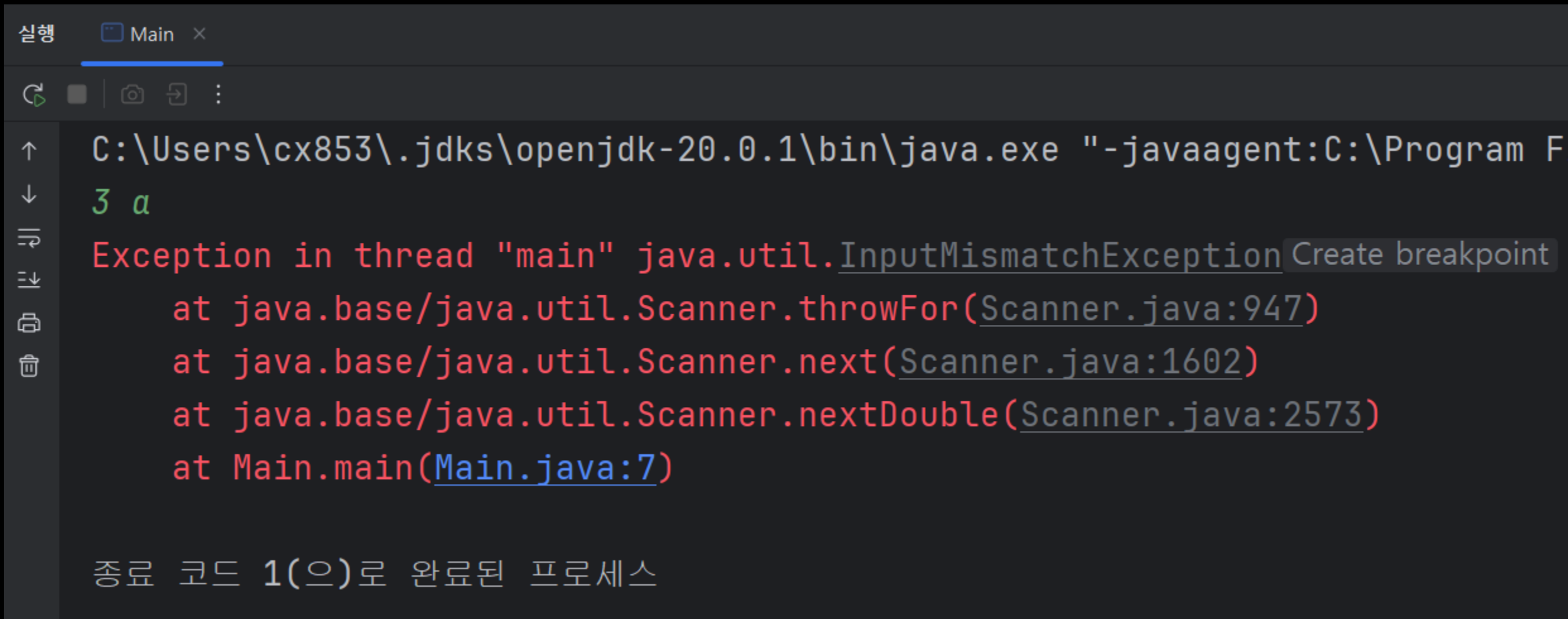
```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int a = s.nextInt();
        double b = s.nextDouble();
        System.out.println("a: " + a);
        System.out.println("b: " + b);
    }
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
3 3.4
a: 3
b: 3.4
```

정수, 실수 입력 오류 시 예외 발생



The screenshot shows an IDE's console window with a dark theme. At the top, there's a tab labeled 'Main' with a close button. Below the tab is a toolbar with icons for running, debugging, and other actions. The main area of the console displays the following text:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe "-javaagent:C:\Program F
3 a
Exception in thread "main" java.util.InputMismatchException Create breakpoint
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)
    at Main.main(Main.java:7)
```

At the bottom of the console, there is a status bar that reads: 종료 코드 1(으)로 완료된 프로세스

Scanner 클래스와 입력 버퍼 – 문자열 입력

`s.nextInt()`



`s.nextLine()`

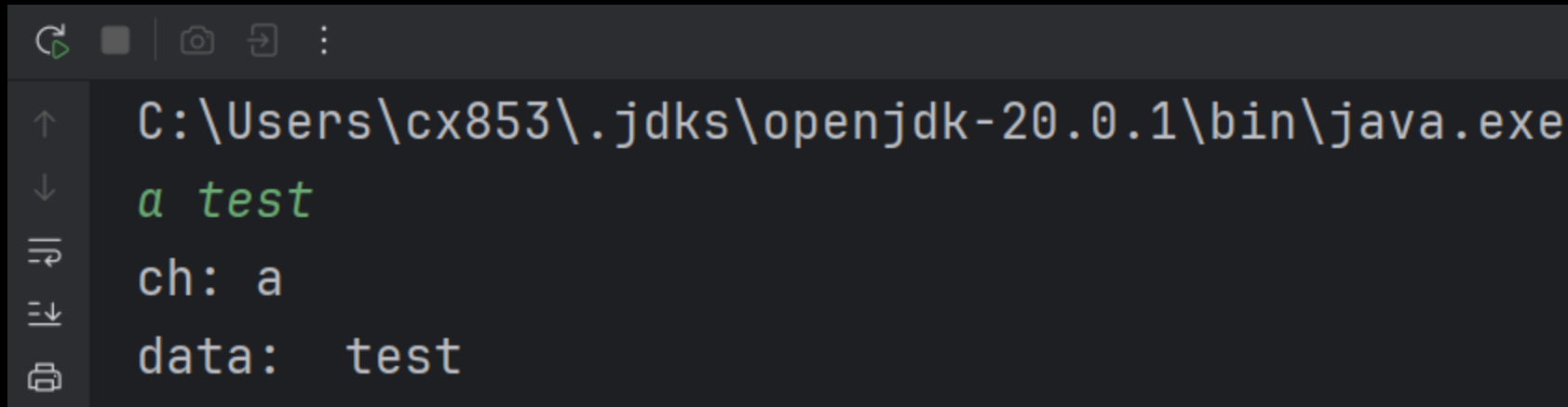


문자, 문자열 입/출력 (08_charStringInput)

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception{
        char ch = (char)System.in.read();
        Scanner s = new Scanner(System.in);
        String data = s.nextLine();

        System.out.println("ch: " + ch);
        System.out.println("data: " + data);
    }
}
```



The screenshot shows a terminal window with the following content:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
a test
ch: a
data: test
```

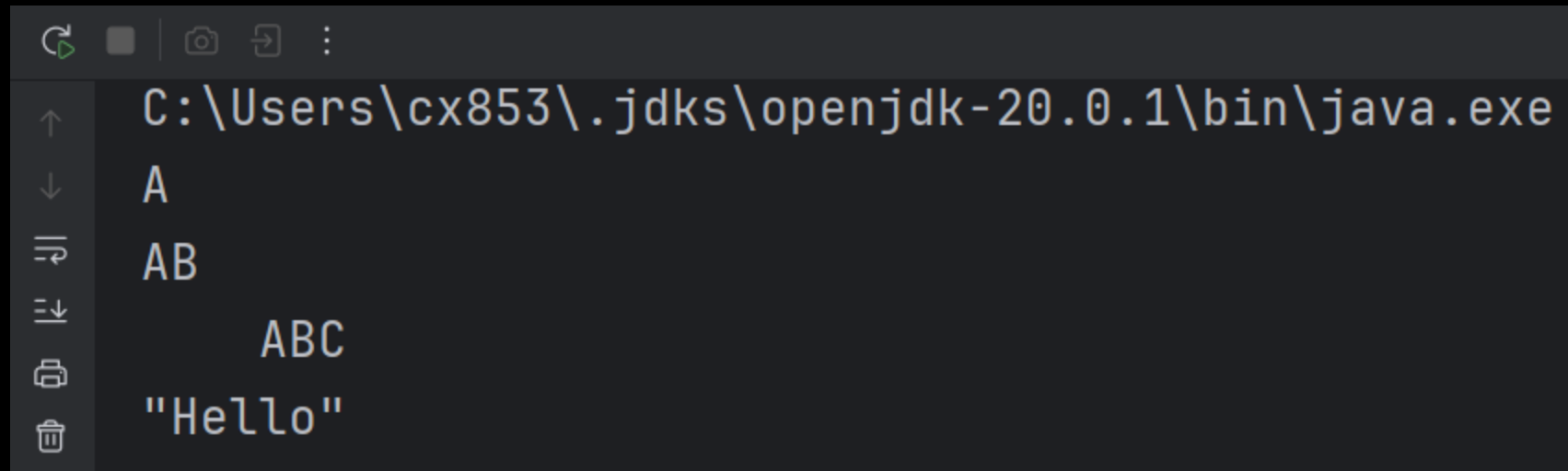
The terminal output demonstrates the program's behavior: it reads the character 'a' and the string 'test' from standard input and prints them back to standard output.

이스케이프 시퀀스 (Escape sequence)

문자	의미
\\	Backslash 문자 자체
\"	
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab

이스케이프 시퀀스 출력 (08_escSeq)

```
public class Main {  
    public static void main(String[] args) {  
        System.out.printf("A\n");  
        System.out.printf("ABC\b\n");  
        System.out.printf("\tABC\n");  
        System.out.printf("\"Hello\"");  
    }  
}
```



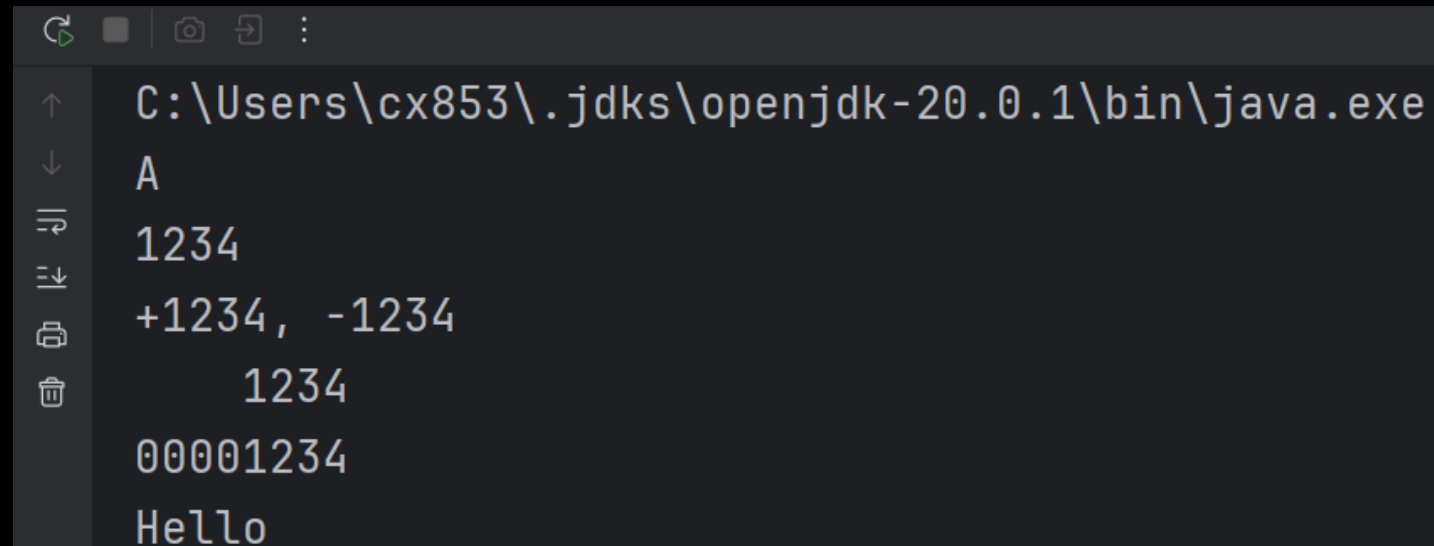
```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
A  
AB  
ABC  
"Hello"
```

형식 문자

- %d 10진수(정수형)
- %c 유니코드 한 문자
- %s 문자열
- %f 실수형
- %t 날짜시간
- %x 16진수

형식문자를 이용한 출력 (08_formatInt)

```
public class Main {  
    public static void main(String[] args) {  
        System.out.printf("%c\n", 65);  
        System.out.printf("%d\n", 1234);  
        System.out.printf("+%d, +%d\n", 1234, -1234);  
        System.out.printf("%8d\n", 1234);  
        System.out.printf("%08d\n", 1234);  
        System.out.printf("%s\n", "Hello");  
    }  
}
```

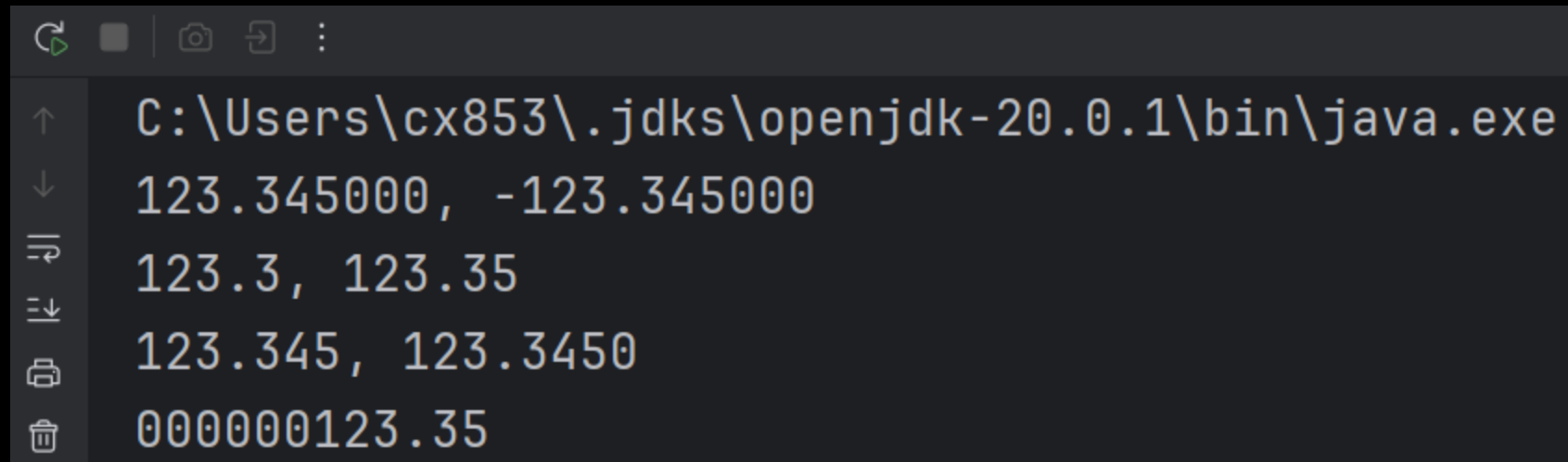


The screenshot shows a Java IDE window with the following output:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
A  
1234  
+1234, -1234  
      1234  
00001234  
Hello
```

형식문자를 이용한 출력 (08_formatDouble)

```
public class Main {  
    public static void main(String[] args) {  
        double data = 123.345;  
        System.out.printf("%f, %f\n", data, -data);  
        System.out.printf("%.1f, %.2f\n", data, data);  
        System.out.printf("%.3f, %.4f\n", data, data);  
        System.out.printf("%012.2f\n", data);  
    }  
}
```



A screenshot of a Java IDE terminal window. The window title bar shows the path C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe. The terminal output displays five lines of formatted double values:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
123.345000, -123.345000  
123.3, 123.35  
123.345, 123.3450  
000000123.35
```

[필수실습 8-1]

나이와 영문 이름 입/출력하기 (20분, 난이도2)

사용자로부터 이름과 나이를 키보드로 입력 받아 출력하는 프로그램을 작성.

Scanner 클래스를 활용해 입력 받은 후 형식문자열을 이용해 출력

나이를 입력하세요: 20

이름을 입력하세요: 철수

당신의 나이는 20살이고 이름은 '철수' 입니다.

9. 연산자 — 첫 번째

연산자

$$a = 3 + 4;$$

- 연산자는 CPU 연산과 직결되는 문법
- 연산자 자체는 하나의 항
- 여러 항을 모아 연산식 작성
 - 여러 항을 소괄호로 묶어 하나의 항처럼 처리
- 연산자와 피연산자로 구성
- 피연산자가 2개 항이면 2항 연산자
(하나는 단항, 셋이면 3항 연산자)

연산자 종류

- 산술 연산자
- 대입 연산자
- 단항 증/감 연산자
- 형변환 연산자
- 비트 (논리) 연산자
- 관계, 논리 연산자, 조건 연산자
- isinstance, new 연산자

연산자 우선 순위와 결합성

순위	연산자	결합성	순위	연산자	결합성
1	() []	L -> R	9	^	L -> R
2	! ++ -- (type) -	L <- R	10		L -> R
3	* % /	L -> R	11	&&	L -> R
4	+ -	L -> R	12		L -> R
5	<< >> >>>	L -> R	13	? :	L <- R
6	< > <= >=	L -> R	14	= += -= *= %= /= &= = ^= ~=	L <- R
7	== !=	L -> R	15	,	L -> R
8	&	L -> R			

연산자 결합성

- 우선순위가 같은 경우 어떤 것을 먼저 연산할 것인지 나타내는 것
- $3 + 4 + 5$ 연산에서 두 덧셈 연산은 우선 순위가 같고 결합성이 $L \rightarrow R$ 이므로 $3 + 4$ 연산을 먼저 수행



산술 연산자

$+$, $-$, $*$, $/$, $\%$

- 대표적인 2항 연산자
- 연산의 결과로 임시결과 발생
- 정수 간 나눗셈의 결과는 반드시 정수가 되며 소수점 이하는 절사

09_addSample

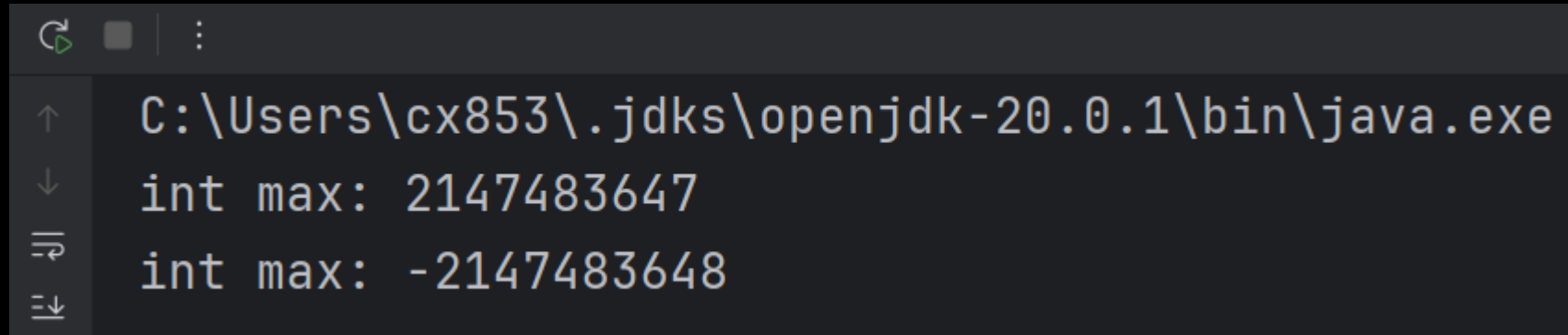
```
public class Main {  
    public static void main(String[] args) {  
        int result = 0;  
        result = 3 + 4;  
        System.out.println("result:" + result);  
    }  
}
```

형승격(Type promotion)

- 임시결과는 피연산자 표현범위 이상의 표현이 가능해야 함
- `char + int` 결과는 `int`
- `double * int` 결과는 `double`
- `4 + 3`과 `4.0 + 3`은 전혀 다른 연산
- 문자열(`String`)과 숫자 연산(`+`)의 결과는 문자열

정수 범위 초과 (09_intMax)

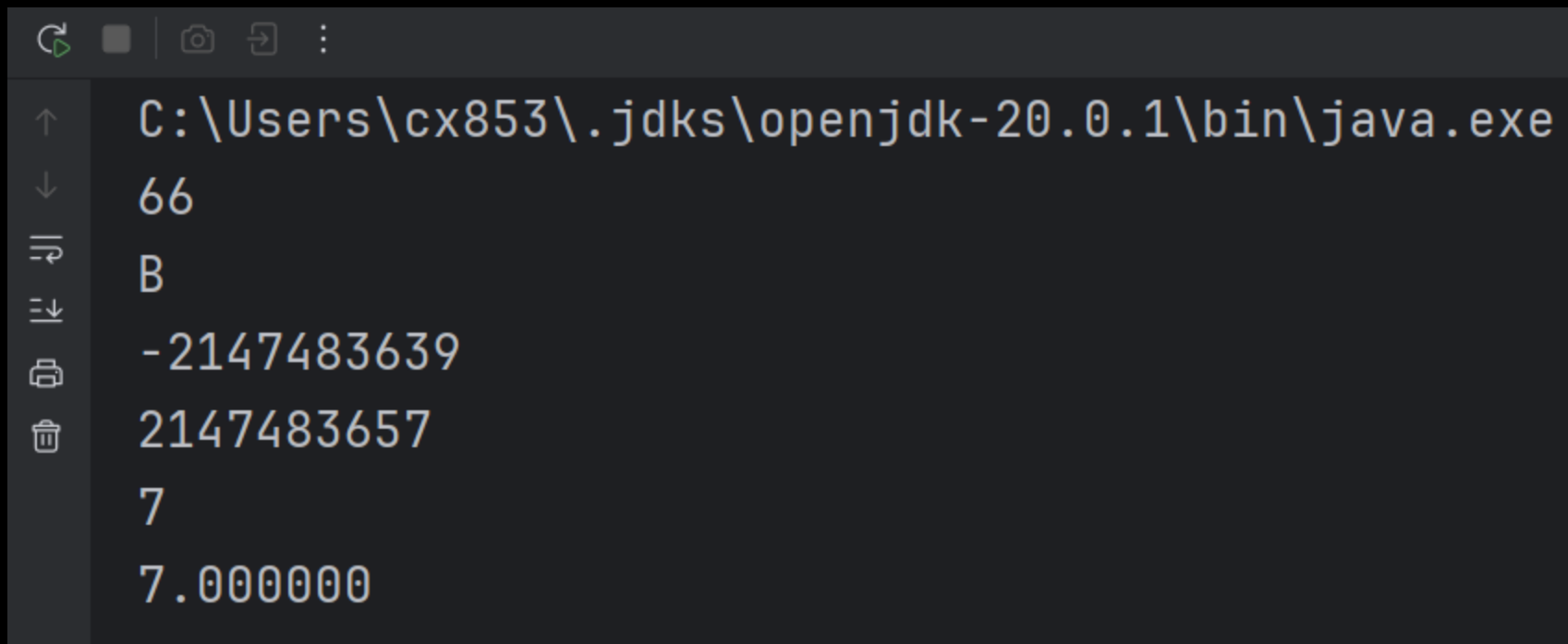
```
public class Main {  
    public static void main(String[] args) {  
        int max = Integer.MAX_VALUE;  
        System.out.println("int max: " + max);  
        System.out.println("int max: " + (max + 1));  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
int max: 2147483647  
int max: -2147483648
```

형승격 (Type promotion)

```
public class Main {  
    public static void main(String[] args) {  
        int maxInt = Integer.MAX_VALUE;  
        long max = Integer.MAX_VALUE;  
        System.out.printf("%d\n", 'A' + 1);  
        System.out.printf("%c\n", 'A' + 1);  
        System.out.printf("%d\n", maxInt + 10);  
        System.out.printf("%d\n", max + 10);  
        System.out.printf("%d\n", 4 + 3);  
        System.out.printf("%f\n", 4.0 + 3);  
    }  
}
```

A terminal window with a dark background. The title bar shows standard window controls (minimize, maximize, close) and a search icon. The command prompt shows the execution of the Java command with various arguments. The output is displayed line by line.

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin>java.exe  
66  
B  
-2147483639  
2147483657  
7  
7.000000
```

문자열 덧셈

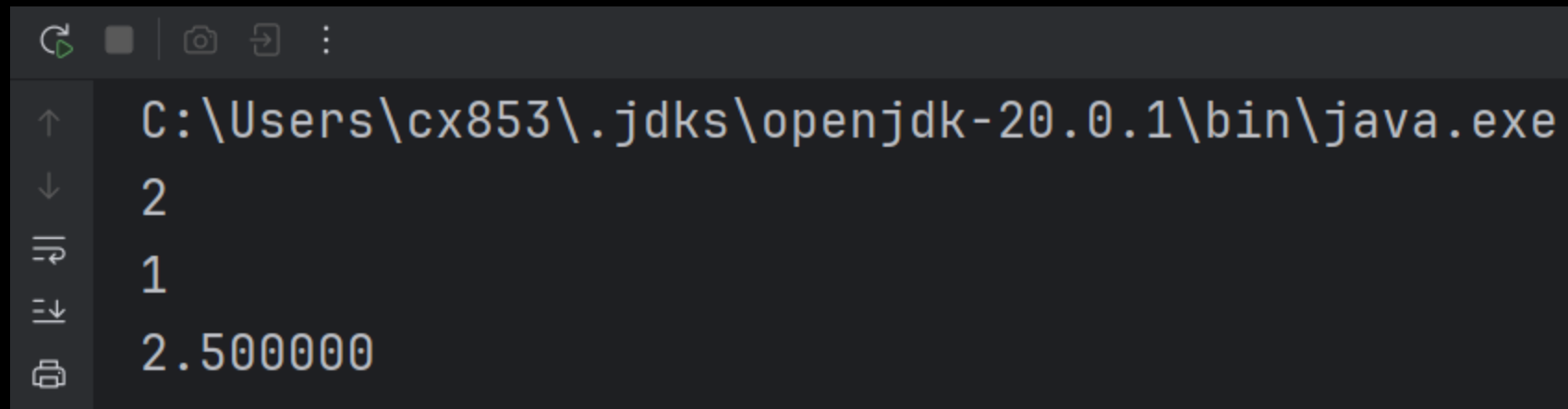
```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello " + "world!");  
        System.out.println("char: " + 'A');  
        System.out.println("char: " + 65);  
        System.out.println("int: " + 10);  
        System.out.println("double: " + 3.3);  
        System.out.println("int: " + 10 + " (test)");  
        System.out.printf("5+5: %d\n", 5+ 5);  
        System.out.println("5+5: " + 5 + 5);  
    }  
}
```

문자열과 정수 연산 결과는 문자열

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
Hello world!
char: A
char: 65
int: 10
double: 3.3
int: 10 (test)
5+5: 10
5+5: 55
```

09_devideSample

```
public class Main {  
    public static void main(String[] args) {  
        System.out.printf("%d\n", 5 / 2);  
        System.out.printf("%d\n", 5 % 2);  
        System.out.printf("%f\n", 5.0 / 2);  
    }  
}
```



A screenshot of a Windows command prompt window. The title bar shows standard Windows icons (refresh, close, maximize, minimize, and a menu icon). The command prompt shows the execution of the Java program. The command entered is `C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe`. The output of the program is displayed on the following lines: `2`, `1`, and `2.500000`. On the left side of the command prompt, there is a vertical toolbar with icons for navigating through the command history (up and down arrows), running the command (a green play button), and other utility icons.

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
2  
1  
2.500000
```

0으로 나누지 못하는 이유

<https://www.youtube.com/watch?v=mZ7pUADoo58>



- 7을 0을 나누면?
- 7에서 0을 빼면 7이고 7은 0보다 큼
- 7에서 계속 0을 빼더라도 결코 7보다 작을 수 없음
- Endless loop

09_divideByZero

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        int input = s.nextInt();
        System.out.println("몫: " + (10 / input));
    }
}
```

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBr
0
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Main.main(Main.java:8)
```

[필수실습 9-1]

평균값 구하기 (15분, 난이도1)

사용자로부터 두 정수를 입력 받아 평균을 출력하는 예제를 작성. 평균값 출력 시 반드시 소수점 둘째 자리까지만 표시.

두 정수를 입력하세요.: 10 20

AVG: 15.00

[필수실습 9-2]

시, 분, 초 계산하기 (30분, 난이도2)

사용자로부터 정수로 초(Second)를 입력 받아 '시:분:초' 형식으로 출력.
시, 분, 초 정보는 모두 두 자리 정수로 표시하고 한 자리 숫자의 경우 앞에 0을 붙여 출력

4000

4000초는 1시간 06분 40초 입니다.

대입 연산자

=

- 단순 대입연산자는 두 피연산자 중 오른쪽 피연산자 (r-value)의 값을 왼쪽 피연산자(l-value)에 저장하는 연산자
- l-value는 Overwrite가 발생하며 기존 값이 사라짐
- 상수는 l-value가 될 수 없음
- 변수 선언 시 사용할 경우 초기값 기술을 위한 문법

대입 연산자 사용예

```
import java.util.Scanner;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        int input = s.nextInt();  
        int result = 0;  
        result = input;  
        System.out.println("result: " + result);
```

```
        3 = 4;  
        final int cutLine = 85;  
        //cutLine = 90;  
        int[] aData = {10, 20, 30};  
        //aData = 40;
```

```
    }
```

```
}
```

F:\독하게 시작하는 Java - Part 1\09_assignError\src\Main.java:

```
java: unexpected type  
    required: variable  
    found:     value
```

final

- 변수이나 한 번 정한 값이 바뀌지 않도록 강제로 상수화
- 심볼릭 상수를 정의하기 위한 문법
- 컴파일러가 코드를 최적화 할 수 있도록 도와주며 코드의 직관성(읽기 좋은 코드) 및 유지보수성을 높여 줌

09_finalSample

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        final int limit = 160;
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();
        if(input >= limit)
            System.out.println("160 이상");
        else
            System.out.println("160 미만");
    }
}
```



Terminal window showing the execution of the Java program. The command 'C:\Users\cx853\jdk\openjdk-20.0.1\bin\java.exe' is entered, followed by the input '150'. The output is '160 미만'.

C:\Users\cx853\jdk\openjdk-20.0.1\bin\java.exe

150

160 미만

[필수실습 9-3]

두 변수 값 교환 (15분, 난이도2)

두 변수의 값을 교환하는 코드를 작성. 사용자로부터 두 정수를 입력 받아 각각을 int형 변수 a, b에 저장하고 임시변수 (tmp)를 활용해 a, b의 값을 교환한 후 출력하는 프로그램을 작성.

3 4

a:4, b:3

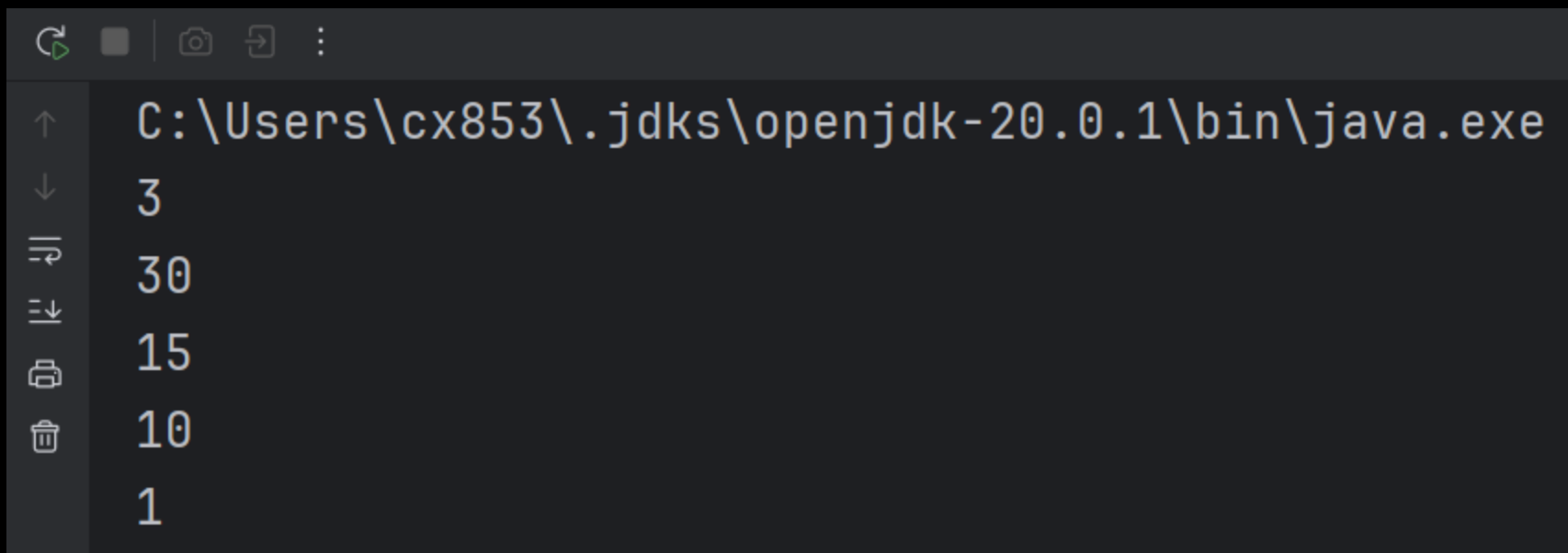
복합 대입 연산자

$+=$ $-=$ $*=$ $\%=$ $/=$ $\&=$ $|=$ $\wedge=$ $\sim=$

- 기능상 단순 대입 연산자와 산술 연산자, 비트 연산자가 조합된 연산자
- 누적 연산 시 $+=$ 연산자 활용

09_comAssign

```
public class Main {  
    public static void main(String[] args) {  
        int result = 0, data = 10;  
        result += 3;  
        System.out.printf("%d\n", result);  
        result *= data;  
        System.out.printf("%d\n", result);  
        result /= 2;  
        System.out.printf("%d\n", result);  
        result -= data - 5;  
        System.out.printf("%d\n", result);  
        result %= 3;  
        System.out.printf("%d\n", result);  
    }  
}
```



A screenshot of a Windows command prompt window. The title bar shows standard Windows window controls (minimize, maximize, close) and a search icon. The command prompt displays the following text:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin>java.exe  
3  
30  
15  
10  
1
```

The command prompt has a vertical toolbar on the left side with icons for navigating the command history (up, down, search, etc.).

[필수실습 9-4]

세 정수 총합 계산하기-누산 (15분, 난이도2)

사용자로부터 세 정수를 입력 받아 총합을 출력. 사용자 입력이 저장되는 변수 하나와 값을 누적하는 변수 하나(총 2개)만 사용해 구현

1

2

3

Total: 6

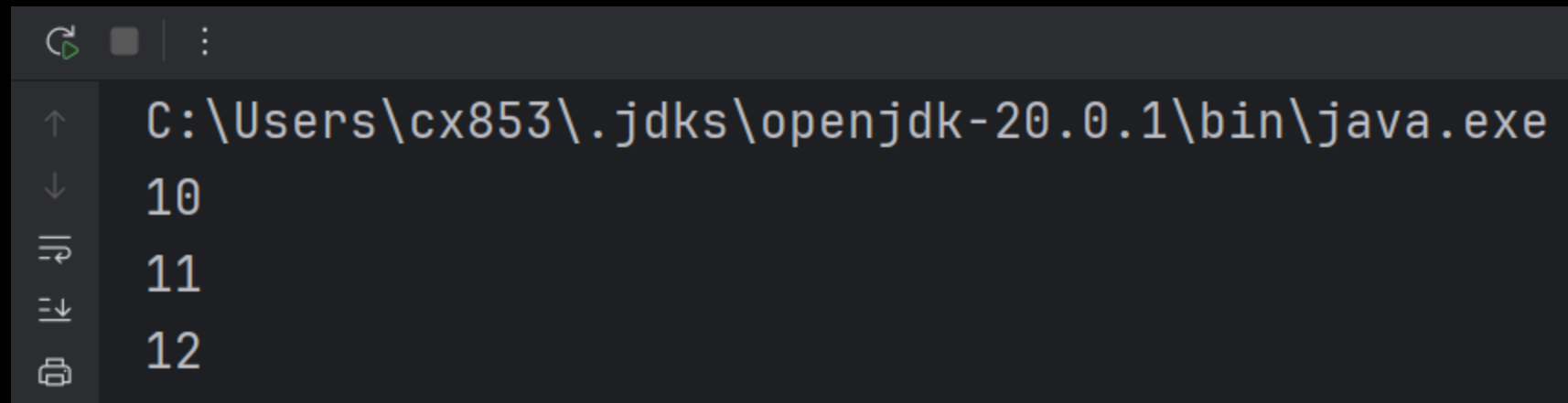
10. 연산자 – 두 번째

단항 증/감 연산자

- 피연산자에 저장된 값을 1씩 증가시키거나 감소시키는 연산
- 피연산자는 반드시 쓰기가 가능한 l-value라야 함
- 전위식, 후위식 표기가 가능하며
- 후위식이면 연산자 우선순위는 전체 식을 평가한 후로 밀림

09_incSample

```
public class Main {  
    public static void main(String[] args) {  
        int x = 10;  
        System.out.println(x);  
        x += 1;  
        System.out.println(x);  
        ++x;  
        System.out.println(x);  
    }  
}
```

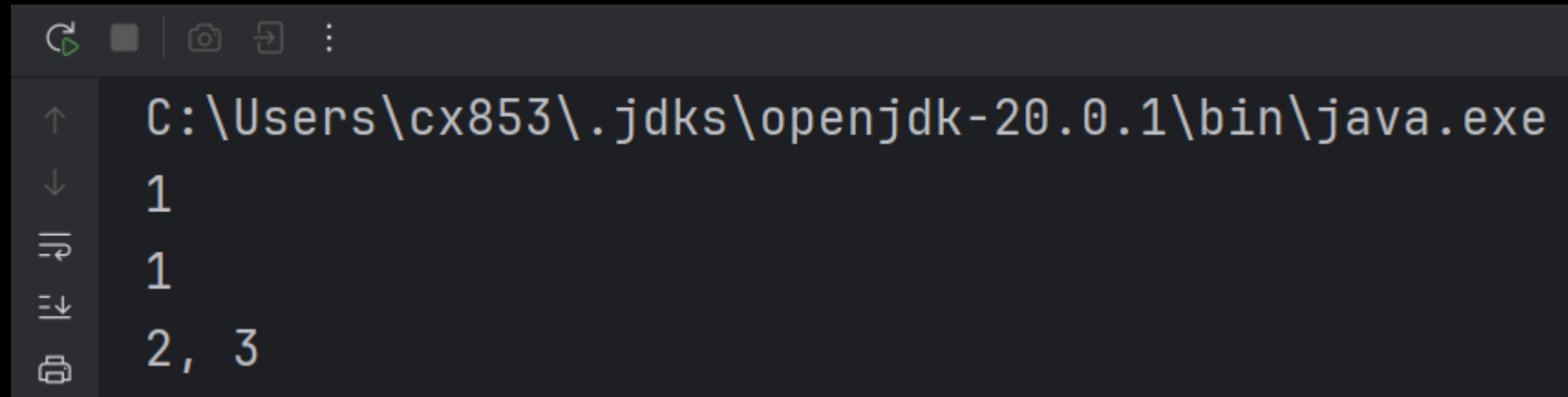


A screenshot of a Java IDE terminal window. The window title bar shows a green play button icon, a gray square icon, and a vertical ellipsis icon. The terminal content shows the command `C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe` being executed, followed by the output `10`, `11`, and `12` on separate lines. On the left side of the terminal, there is a vertical toolbar with icons for running (a green play button), stepping through (a downward arrow), stepping back (a leftward arrow with a double bar), stepping forward (a rightward arrow with a double bar), and copying (a document icon).

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
10  
11  
12
```

09_incPostfix

```
public class Main {  
    public static void main(String[] args) {  
        int x = 0, result = 0;  
        System.out.println(++x);  
        System.out.println(x++);  
        result = x++;  
        System.out.printf("%d, %d\n", result, x);  
    }  
}
```



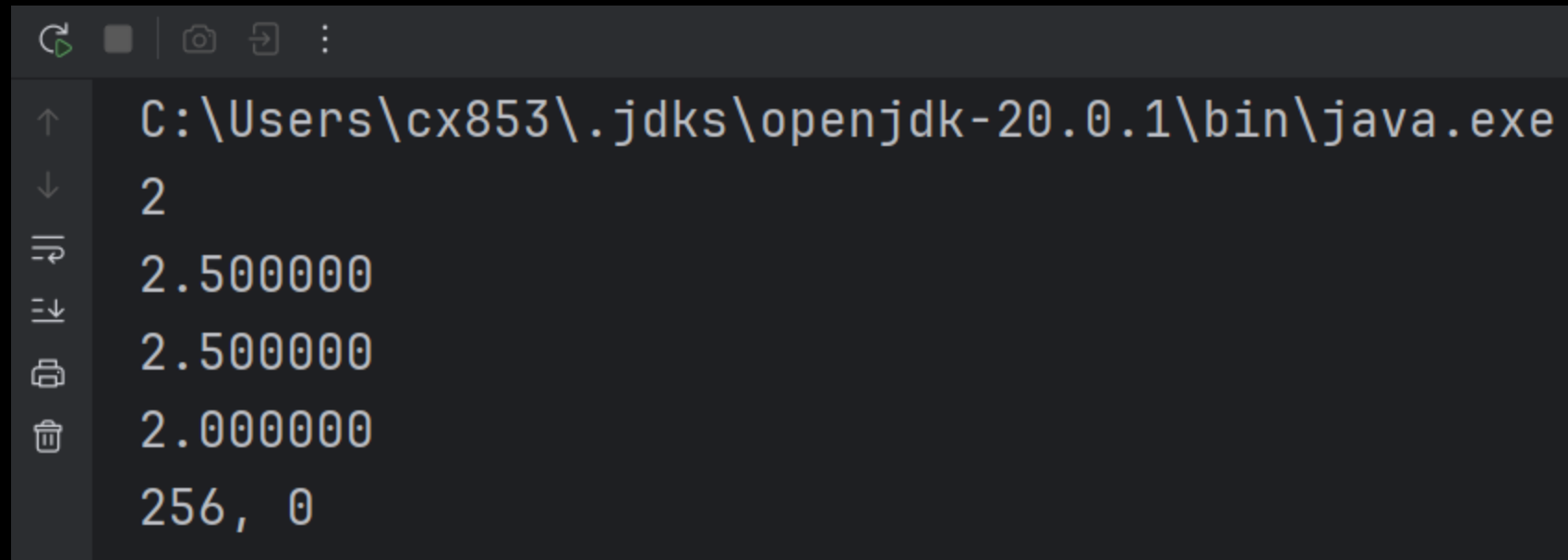
```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
1  
1  
2, 3
```

형변환 연산자

- 피연산자의 형식을 강제로 변경해주는 단항 연산자
 - `double a = 3.3;`
 - `int b = (int)a;`
- 부적절한 변환 시 정보가 유실될 수 있음

10_typecastSample

```
public class Main {  
    public static void main(String[] args) {  
        System.out.printf("%d\n", 5 / 2);  
        System.out.printf("%f\n", 5.0 / 2);  
        System.out.printf("%f\n", (double)5 / 2);  
        System.out.printf("%f\n", (double)(5 / 2));  
        System.out.printf("%d, %d\n", 256, (byte)256);  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
2  
2.500000  
2.500000  
2.000000  
256, 0
```

비트 (논리) 연산자

&, |, ^, ~, <<, >>, >>>

- 자료를 비트 단위로 논리 식을 수행하는 연산
- 보통 2진수로 변환해 판단
- AND, OR, NOT, XOR, Shift left, Shift right
- NOT은 단항, 나머지는 모두 2항 연산자

10_bitOp

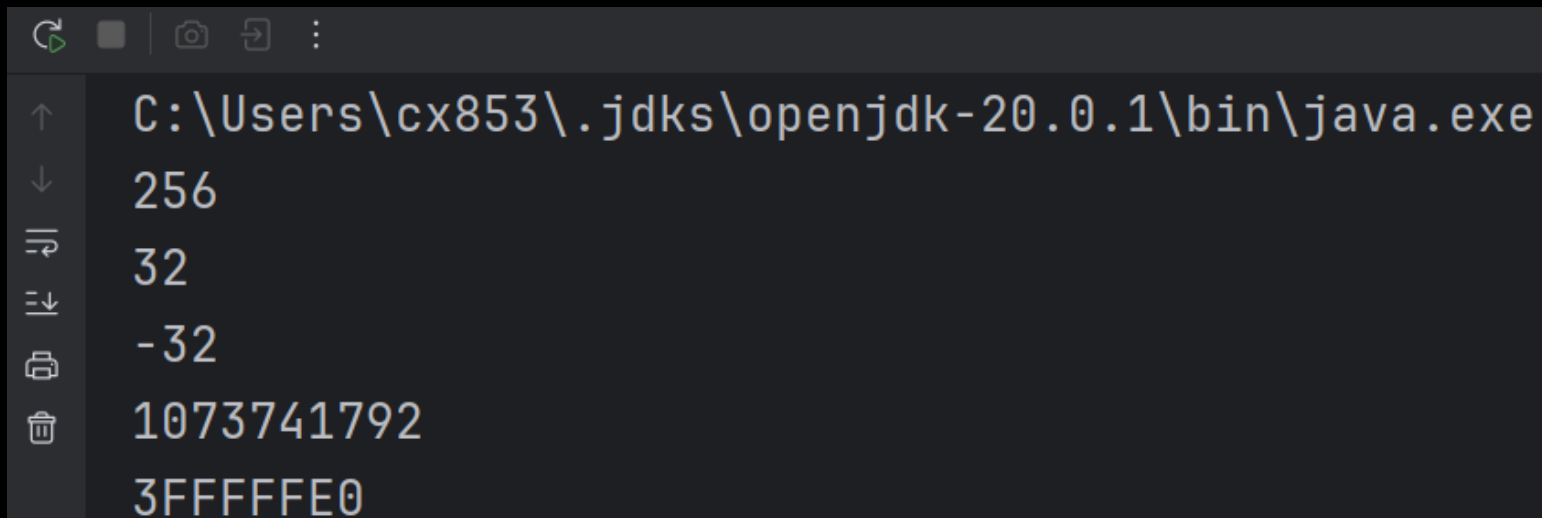
```
public class Main {  
    public static void main(String[] args) {  
        int data = 0x11223344;  
        System.out.printf("%d\n", data);  
        System.out.printf("%08X\n", data & 0xFFFF0000);  
        System.out.printf("%08X\n", data | 0xFFFF0000);  
        System.out.printf("%08X\n", data ^ 0xFFFF0000);  
        System.out.printf("%08X\n", ~data);  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
287454020  
11220000  
FFFF3344  
EEDD3344  
EEDDCCBB
```

10_bitShift

```
public class Main {  
    public static void main(String[] args) {  
        int data = 128;  
        System.out.printf("%d\n", data << 1);  
        System.out.printf("%d\n", data >> 2);  
        System.out.printf("%d\n", -data >> 2);  
        System.out.printf("%d\n", -data >>> 2);  
        System.out.printf("%08X\n", -data >>> 2);  
    }  
}
```



A screenshot of a Java command prompt window. The title bar shows standard Windows window controls. The command prompt displays the execution of the Java program, showing the output of the printf statements. The output is as follows:

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
256  
32  
-32  
1073741792  
3FFFFFFE0
```

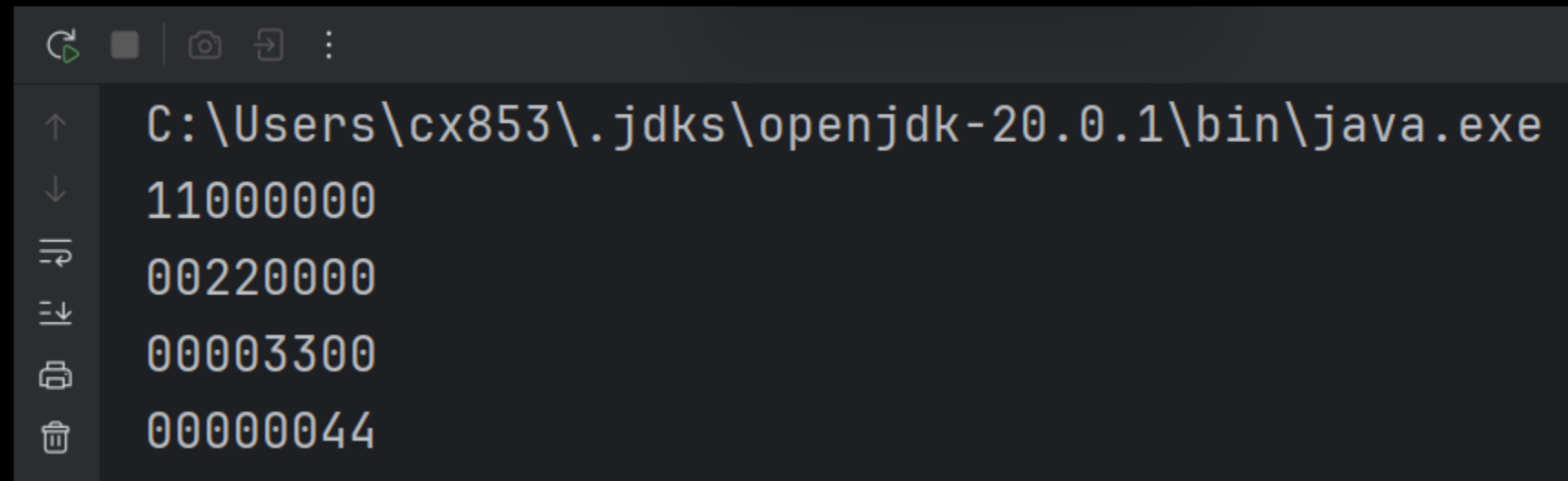
비트 마스크 연산

```
int data = 0x11223344; data & 0x0000FFFF
```

- 데이터에서 특정 영역의 값이 모두 0이 되도록 지우는 연산
- AND의 특징을 이용
- 0과 AND연산을 수행하면 결과는 무조건 0
- JVM은 Big endian 시스템

10_bitMask

```
public class Main {  
    public static void main(String[] args) {  
        int data = 0x11223344;  
        System.out.printf("%08X\n", data & 0xFF000000);  
        System.out.printf("%08X\n", data & 0x00FF0000);  
        System.out.printf("%08X\n", data & 0x0000FF00);  
        System.out.printf("%08X\n", data & 0x000000FF);  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
11000000  
00220000  
00003300  
00000044
```

[필수실습 10-1]

뱀셈 연산 직접 구현하기 (15분, 난이도2)

사용자로부터 두 정수를 입력 받아 뱀셈을 수행하고 결과를 출력하는 프로그램을 작성. 단, 절대로 뱀셈 연산자를 사용하지 말고 비트 연산자를 이용해 구현.

9 6

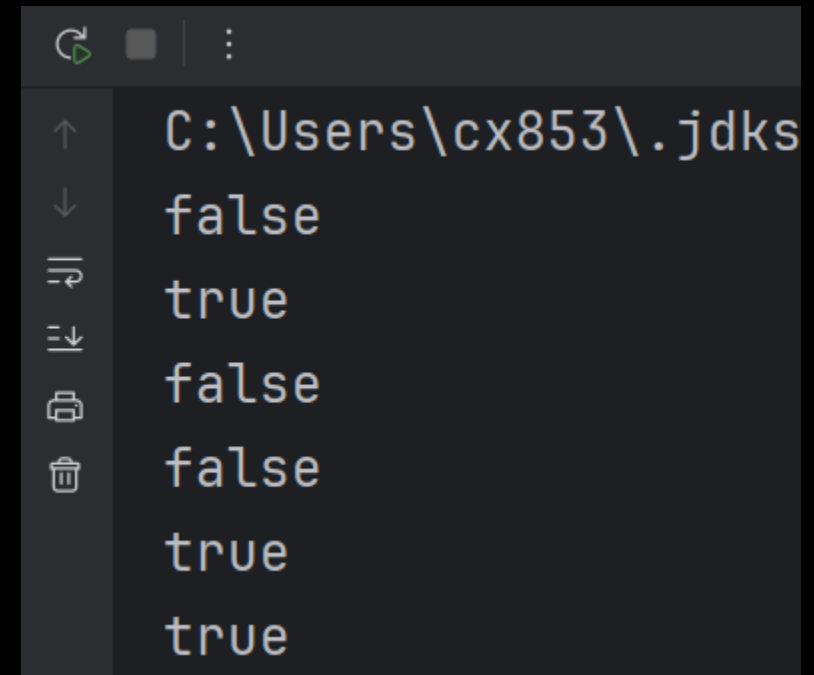
결과: 3

관계 연산자

- 두 피연산자의 값을 비교(뺄셈)해 결과 도출
 - 문자열은 제외
- 상등, 부등, 관계 연산자로 분류
- 상등(==), 부등(!=) 연산은 좌항에서 우항을 뺀 결과가 0이면 true 아니면 false
- 실수형에 대해 상등, 부등연산은 불가
 - 면접에 자주 나오는 질문 중 하나
- >, <, <=, >= (비교연산)

10_relationSample

```
public class Main {  
    public static void main(String[] args) {  
        int a = 5, b = 10;  
        System.out.println(a == b);  
        System.out.println(a != b);  
        System.out.println(a > 5);  
        System.out.println(b < 5);  
        System.out.println(b >= 10);  
        System.out.println(b <= a + 5);  
    }  
}
```

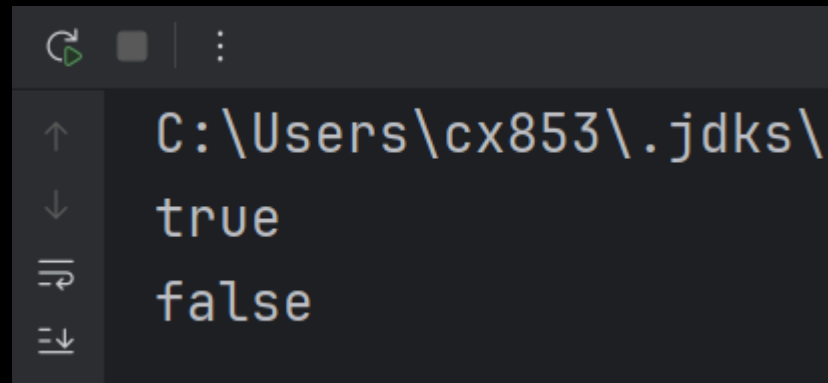


The screenshot shows a Java IDE window with the file path `C:\Users\cx853\.jdk`. The output of the program is displayed in the console, showing the results of the comparison operations performed in the code.

Operation	Result
<code>a == b</code>	false
<code>a != b</code>	true
<code>a > 5</code>	false
<code>b < 5</code>	false
<code>b >= 10</code>	true
<code>b <= a + 5</code>	true

10_relationError

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        System.out.println(300 == 299.999998F);  
        System.out.println(300 == 299.999999);  
        //System.out.println(5 < a < 15);  
    }  
}
```



C:\Users\cx853\.jdk8\bin\java.exe
true
false

```
F:\독하게 시작하는 Java - Part 1\10_relationError\src\Main.java:10: error: bad operand types for binary operator '<'  
        first type:  boolean  
        second type: int
```


논리 연산자

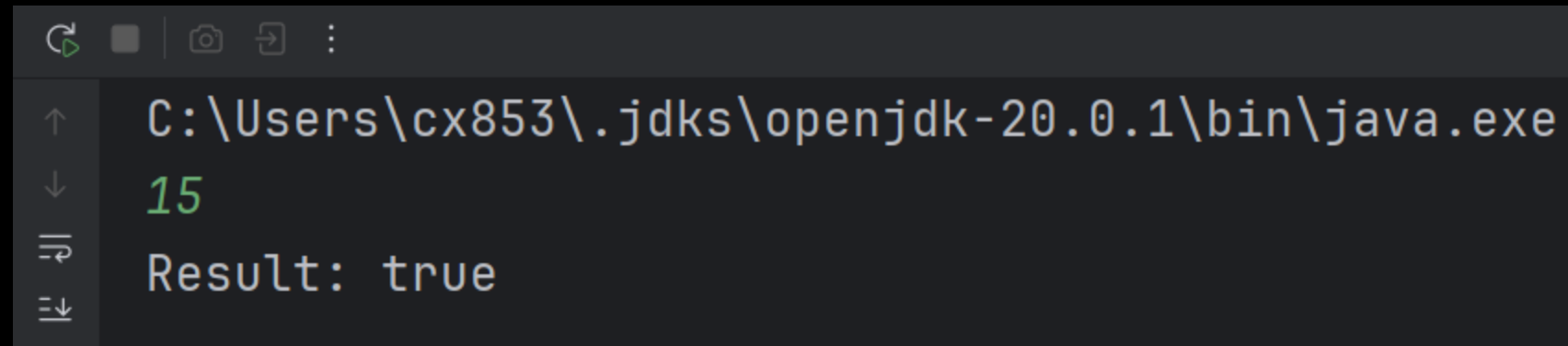
- 항 혹은 연산식을 피연산자로 두는 논리합(\parallel), 논리곱($\&\&$)
2항 연산자
- 논리 부정 연산자(!)는 단항 연산자
- 값의 범위 표현 시 사용되는 것이 보통
- 결합성이 $L \rightarrow R$ 이므로 왼쪽에 등장하는 연산식을 우선 평가하고 이어지는 연산식을 수행할 것인지 판단

10_logicalSample

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();

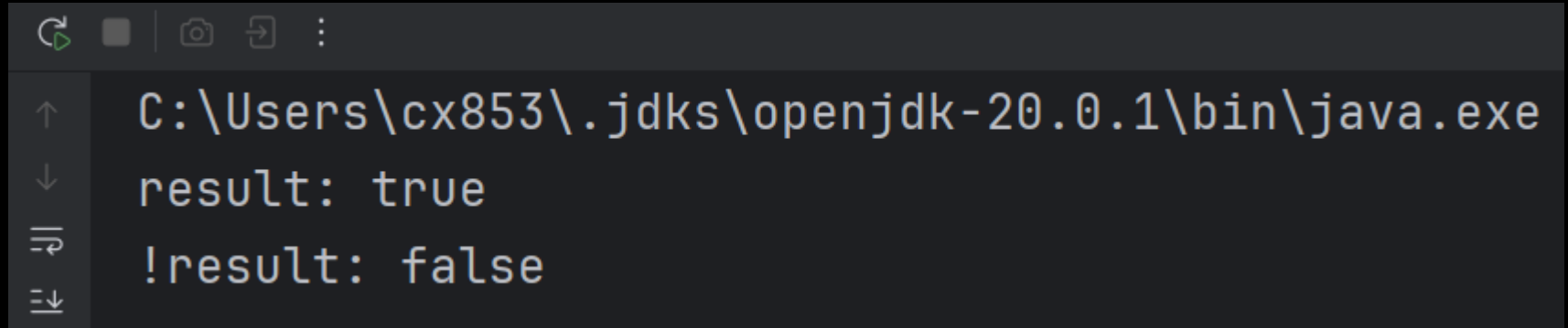
        boolean result = input < 5 || input > 10;
        System.out.println("Result: " + result);
    }
}
```

A screenshot of a Java IDE terminal window. The window has a dark background with a toolbar at the top containing icons for running, stopping, and other actions. The command prompt shows the execution of the Java program: `C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe`. The input `15` is shown in green. The output `Result: true` is shown in white.

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
15
Result: true
```

10_logicalNot

```
public class Main {  
    public static void main(String[] args) {  
        boolean result = true;  
        System.out.println("result: " + result);  
        System.out.println("!result: " + !result);  
    }  
}
```

A screenshot of a Java IDE terminal window. The window has a dark background with a light gray title bar. The title bar contains several icons: a green play button, a gray square, a camera icon, a magnifying glass icon, and a vertical ellipsis. The terminal content shows the command path 'C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe' followed by the output 'result: true' and '!result: false'. On the left side of the terminal, there are four icons: an upward arrow, a downward arrow, a double horizontal line, and a double horizontal line with a downward arrow.

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
result: true  
!result: false
```

Short circuit

- 논리 식은 왼쪽부터 실행
- 피연산자 항이 식일 경우 식부터 평가
- 논리합의 경우 왼쪽 조건이 만족되면 이후 식은 연산하지 않음
- 논리곱의 경우 마지막 식까지 모두 평가해 모든 결과가 참인지 확인

10_shortCircuit

```
import java.util.Scanner;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        System.out.print("나이: ");  
        int age = s.nextInt();  
        System.out.print("키: ");  
        int height = s.nextInt();  
  
        System.out.printf("결과: %b\n",  
            age >= 20 && age <= 30 && height >= 150 ||  
            age >= 100 ||  
            height >= 200);  
    }  
}
```



C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe

나이: 120

키: 3

결과: true

조건 (3항) 연산자

1-value = 조건식 ? 항A : 항B

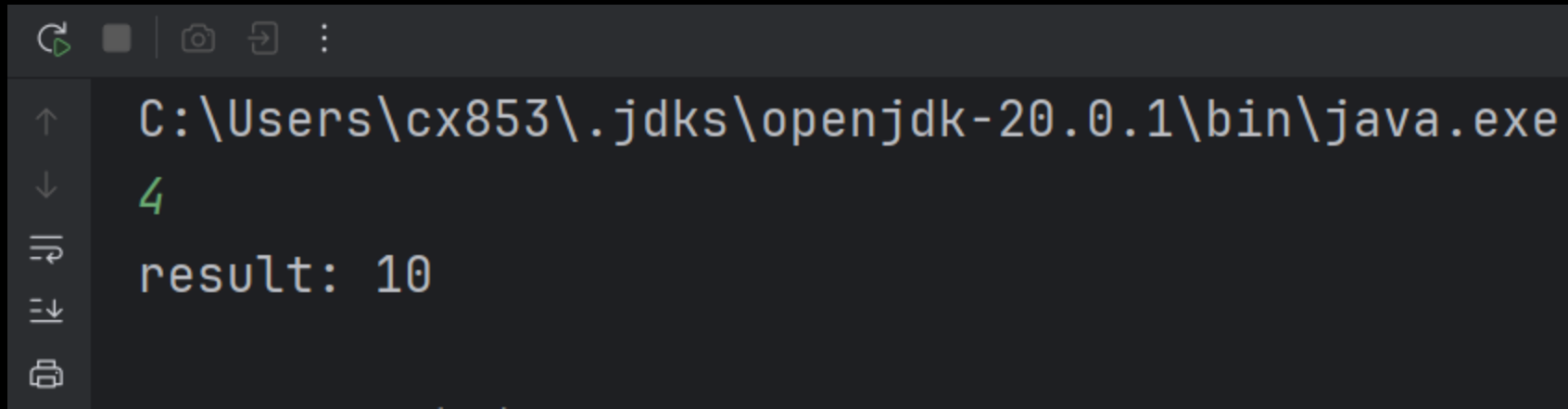
- 유일한 3항 연산자
- 조건식 ? 항A : 항B
- 논리적 오류를 피하려면 선택 대상 항은 괄호로 묶어 표기
 - 조건식 ? (항A) : (항B)

10_condiSample

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();

        int result = input <= 10 ? 10 : 20;
        System.out.println("result: " + result);
    }
}
```

A screenshot of a terminal window with a dark background. The command prompt shows the path 'C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe' followed by the input '4'. The output of the program is 'result: 10'. The terminal has a standard toolbar at the top with icons for running, stopping, and other actions.

```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
4
result: 10
```

조건 연산자 잘못된 활용 예 (10_condiError)

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();

        int result = 0, result2;
        result2 = (input <= 10) ? (result = 10) : (result = 20);
        System.out.println("result: " + result);
    }
}
```


[필수실습 10-2]

합격/불합격 판단하기 (20분, 난이도2)

사용자로부터 점수(0~100)를 입력 받아 80점 이상이면 '합격' 그렇지 않으면 '불합격'이라고 출력하는 프로그램을 작성.

반드시 3항 연산자를 사용

점수를 입력하세요: 80

결과: 합격

[필수실습 10-3]

최댓값 구하기 - 서바이벌 방식 (30분, 난이도3)

사용자로부터 입력 받은 정수 중 가장 큰 수를 출력하는 프로그램을 작성.
정수는 부호가 있는 32비트 정수로 한정하며 정수는 한 번에 한 값만 입력
받고 내부적으로 최댓값을 계속 갱신하는 구조로 작성.

10

20

30

MAX: 30

[필수실습 10-4]

최댓값 구하기 - 토너먼트 방식 (30분, 난이도3)

사용자로부터 입력 받은 정수 중 가장 큰 수를 출력하는 프로그램을 작성.
32비트 정수를 한 행에서 한 번에 세 값을 모두 입력 받아야 함. 최댓값은
세 값을 비교한 후 출력

10 20 30

MAX: 30

11. 기본 제어문

if

```
if(조건식)    구문1;
```

```
if(조건식)  
    구문2;
```

```
if(조건식)  
{  
    구문3;  
    구문4;  
}
```

- 분기문(Branching statement)
- 조건식의 결과에 따라 절차상 수행할 구문이 달라질 수 있는 제어문
 - 추가 코드 수행
- 3항 연산자와 논리적으로 유사

if문 실습 시 주의사항

- 조건식은 괄호로 묶이며 뒤에 세미콜론을 붙이지 않음
- 실행할 구문이 여러 행이면 중괄호로 묶어(Scope)주며 반드시 들여 쓰기를 일치시켜 주어야 함

11_ifAndScope

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("나이를 입력하세요: ");
        int age = s.nextInt();

        if(age >= 20) {
            System.out.printf("처리전, 당신의 나이는 %d세 입니다.\n", age);
            age = 20;
        }

        System.out.println("최종 당신의 나이는 " + age + "세 입니다.");
    }
}
```

```
↑ C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe
↓
⇨ 나이를 입력하세요: 30
⇩
⇩ 처리전, 당신의 나이는 30세 입니다.
⇩
⇩ 최종 당신의 나이는 20세 입니다.
```

IntelliJ 디버깅 기능

- Position breakpoint

- Ctrl + F8
- Shift + F9(Debug mode run), F8(Step over)
- 디버그 모드 실행 시에만 작동
- 식평가 지원

- Data breakpoint

- 변수 값에 대해 조건식을 적용

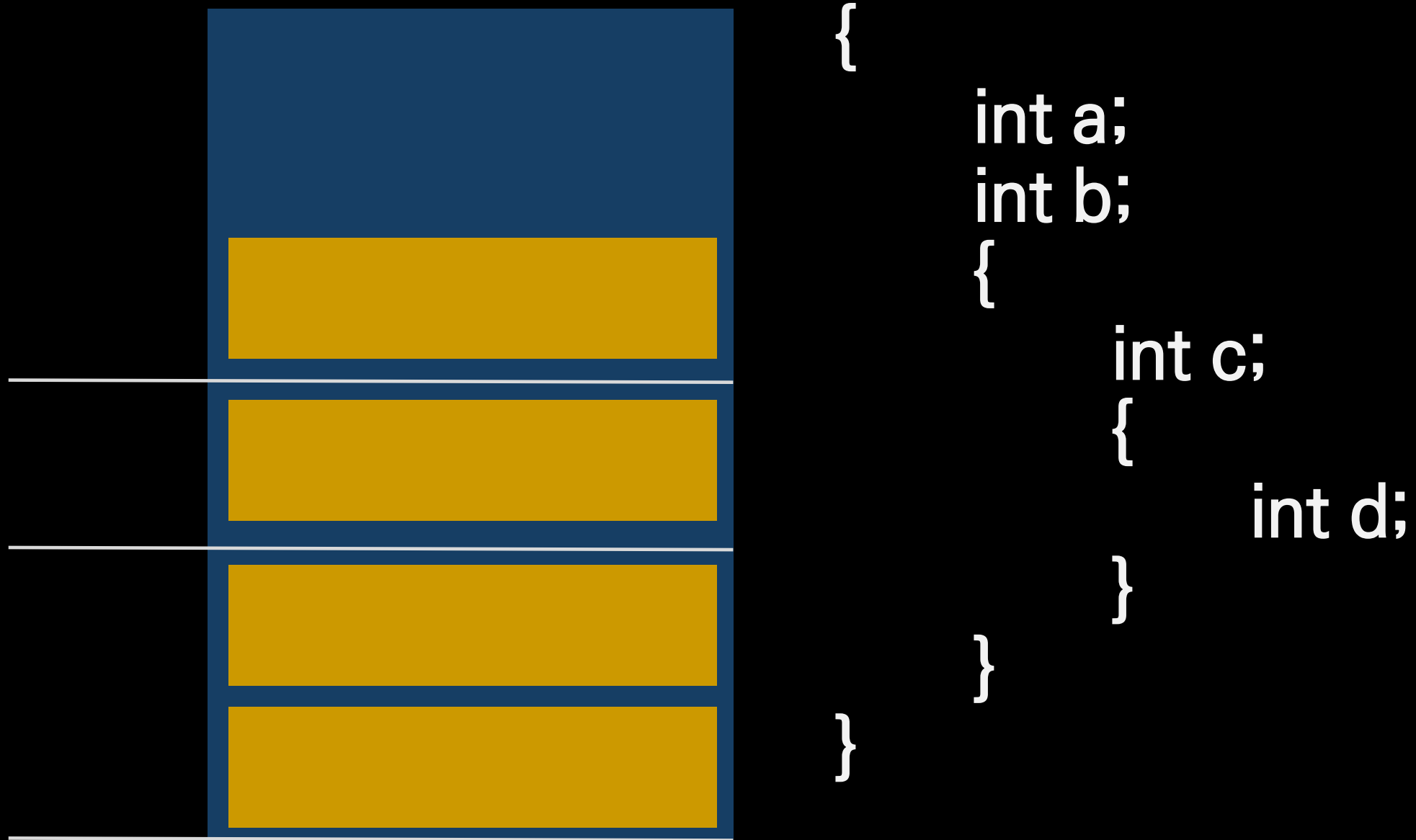
- ‘객체@숫자’ 형식에서 숫자는 ObjectID

- Java Debug Wire Protocol

Scope

- 여러 구문을 한 덩어리로 묶어주기 위해 사용
- 중괄호 {} 사용
- 지역 변수로 선언된 식별자의 유효 범위는 스코프 내부로 제한됨
- 스코프를 벗어난 식별자는 접근 할 수 없음
- 변수 선언 시 속한 스코프 수준에서 접근 가능한 변수와 이름이 같은 변수는 선언 할 수 없음

다시 살펴보는 자동변수와 스택



[필수실습 11-1]

최댓값 구하기 - if문 버전 (10분, 난이도2)

사용자로부터 입력 받은 정수 중 가장 큰 수를 출력하는 프로그램을 작성. 앞서 작성한 실습코드와 논리적으로 동일하며 한 번에 한 값만 입력 받고 내부적으로 최댓값을 계속 갱신하는 구조로 작성. 반드시 if문을 사용할 것!

10

20

30

MAX: 30

[필수실습 11-2]

버스요금 계산 (20분, 난이도2)

버스 기본 요금을 1000원으로 가정하고 20세 미만인 경우에는 요금을 25% 할인하고 20세 이상 성인은 할인하지 않음. 단, 나이가 20을 넘는 경우 20으로 강제 조정해 출력할 것.

15

나이: 15, 최종요금: 750원

25

나이: 20, 최종요금: 1000원

if else

```
if(조건식)
{
    구문1;
    구문2;
}
else
    구문3;

구문4;
```

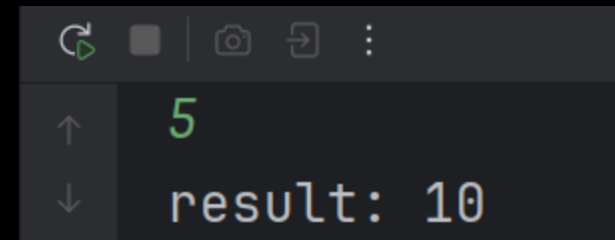
- 조건식을 만족하는 경우와 그렇지 않은 경우 수행되는 식을 배타적으로 기술할 수 있는 제어문
- else 오른쪽에는 조건식도 세미콜론도 없음에 주의

11_ifElseSample

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();

        //10_condiSample
        //int result = input <= 10 ? 10 : 20;
        int result = 0;
        if(input <= 10)
            result = 10;
        else
            result = 20;
        System.out.println("result: " + result);
    }
}
```



if else 중첩

```
if(조건식1)
{
    구문1;
    if(조건식2)
        구문2;
    else
        구문3;
}
else
    구문4;
```

- 각종 제어문 내부에 다시 제어문을 넣을 수 있음
- 여러 발생가능한 경우의 수를 나열한 후 피라미드 식으로 분류하는 경우 유용
- 조건에 의한 분류와 선택 구조

if else문 중첩 (11_ifNested)

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = s.nextInt();
        int result = 0;
        if(input <= 10) {
            if(input < 0)
                result = 0;
            else
                result = 10;
        }
        else
            result = 20;

        System.out.println("result: " + result);
    }
}
```


[필수실습 11-3]

연령별 버스요금 계산 (30분, 난이도3)

버스 기본 요금을 1000원으로 가정하고 나이에 따라 다음과 같이 요금을 할인율 적용해 최종 요금을 출력. if else문을 중첩해 구현할 것.

0~3	영유아	100% (무료)
4~13	어린이	50%
14~19	청소년	25%
20 이상	성인	0%

다중 if

```
if(조건식1)
    구문1;
else if(조건식2)
    구문2;
else if(조건식3)
    구문3;
else
    구문4;
```

- 마치 if else를 여럿 연이어 기술한 것과 같은 형식
- 다수의 조건식이 등장
- 생각해볼 문제. 정렬을 하는 이유는 무엇일까?

11_classify01

```
int score = s.nextInt();  
char ch = 'A';
```

```
if(score >= 90)  
    ch = 'A';  
else if (score >= 80)  
    ch = 'B';  
else if (score >= 70)  
    ch = 'C';  
else if (score >= 60)  
    ch = 'D';  
else  
    ch = 'F';
```

```
System.out.println("점수: " + score + " 학점: " + ch);
```

11_classify02

```
if(score >= 80) {  
    if(score >= 90)  
        ch = 'A';  
    else  
        ch = 'B';  
}  
else {  
    if (score >= 70)  
        ch = 'C';  
    else {  
        ch = 'D';  
        if (score < 60)  
            ch = 'F';  
    }  
}
```

switch-case

- if문처럼 정보를 분류하고 경우를 선택하는 제어문
- 조건식의 결과는 반드시 정수 혹은 문자열
- 각 case는 결국 상등 연산 시 일치하는 값에 해당하는 정수 혹은 문자열
- case는 콜론으로 끝남
 - 예) case 3:
- 한 case에 대해 break문 조합 (생략가능)

11_switchCase01

```
import java.util.Scanner;

public class Main {
    public static void main(String[]
args) {
        Scanner s = new
Scanner(System.in);

        int left = s.nextInt();
        char operator =
s.next().charAt(0);
        int right = s.nextInt();
        int result = 0;

        switch (operator)
        {
            case '+':
                result = left + right;
                break;

            case '-':
                result = left - right;
                break;

            case '*':
                result = left * right;
                break;

            case '/':
                result = left / right;
                break;
        }

        System.out.println("result: " +
result);
    }
}
```

11_switchCase02

```
Scanner s = new Scanner(System.in);
int score = s.nextInt();
char ch = 'A';

switch (score / 10) {
    case 10:
    case 9:
        ch = 'A';
        break;

    case 8:
        ch = 'B';
        break;

    ...
    default:
        ch = 'F';
}

System.out.println("result: " + ch);
```

12. 반복문

기본 개념 및 종류

- 특정 구간의 코드(구문)를 반복해 실행하는 문법
- 종류
 - while
 - do while
 - for
 - foreach (확장 for문)

while

```
while(조건식1)
    구문1;
구문2;
```

```
while(조건식2)
{
    구문3;
    구문4;
}
```

- if문과 유사한 구조
- 조건식이 참일 경우 구간 코드를 계속 반복해서 수행
- 반복 수행 구간코드 내부에서 반복을 멈출 수 있는 연산이 반드시 있어야 함
- 무한루프는 최악의 논리 오류 중 하나

12_whileInput

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception {
        int input = 0;
        while((input = System.in.read()) != 'q') {
            System.out.println("input: " + input);
        }
        System.out.print("Exit");
    }
}
```

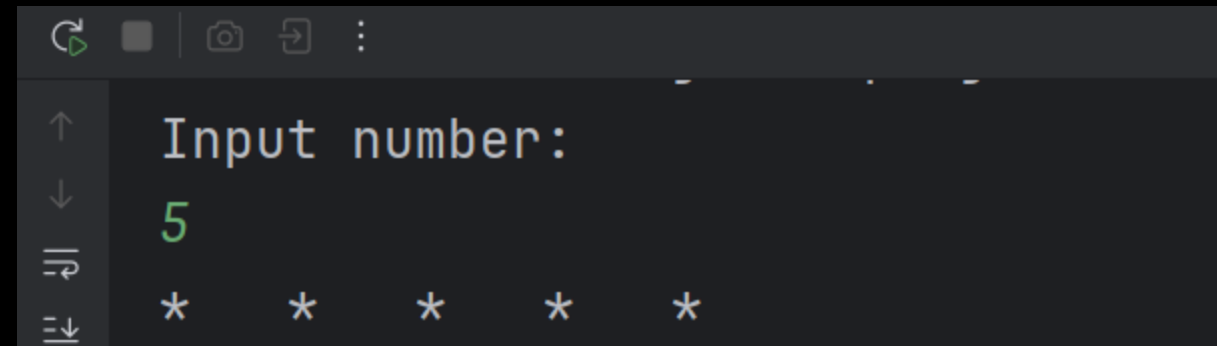
12_whilePrintStar

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Input number: ");
        int input = s.nextInt();

        if(input < 0)        input = 1;
        else if(input > 9)    input = 9;

        int cnt = 0;
        while(cnt < input) {
            System.out.print("*\t");
            cnt++;
        }
    }
}
```

A screenshot of a Java IDE window. The title bar shows standard window controls. The main area displays the output of the program. It starts with the prompt "Input number:" followed by the user input "5" on the next line. Below the input, there is a row of five asterisks, each separated by a tab character, resulting in a wide row of stars.

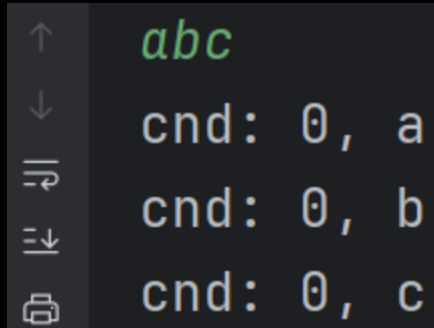
```
Input number:
5
*      *      *      *      *
```

12_variableInLoop

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception{
        Scanner s = new Scanner(System.in);

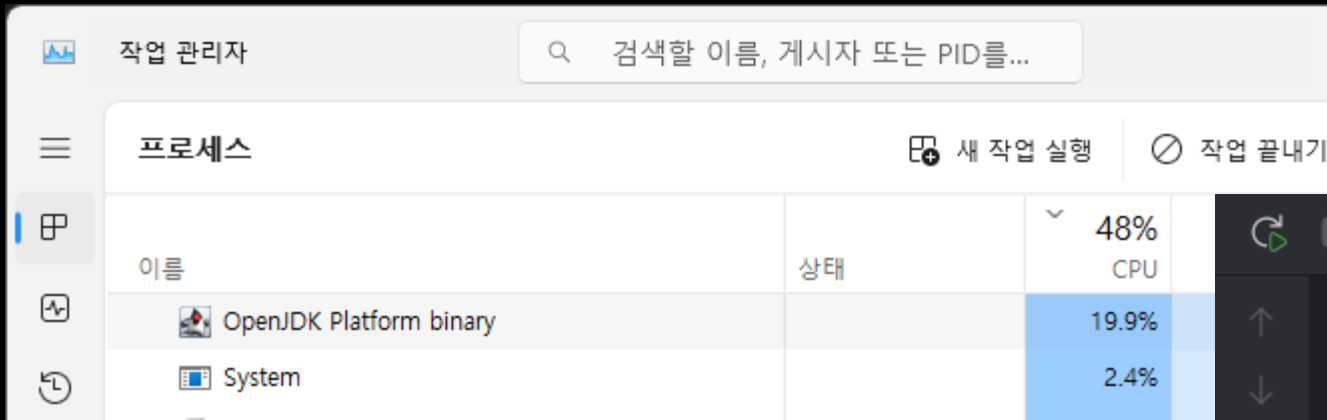
        int ch = 0;
        while((ch = System.in.read()) != 'q') {
            int cnt = 0;
            System.out.printf("cnd: %d, %c\n", cnt, ch);
            cnt++;
        }
    }
}
```



↑ *abc*
↓ cnd: 0, a
↵ cnd: 0, b
⇓ cnd: 0, c
🖨

13_endlessLoop

```
public class Main {  
    public static void main(String[] args) {  
        int cnt = 0;  
        while(cnt >= 0) {  
            cnt = cnt + 1 - 1;  
            //cnt++;  
        }  
        System.out.println("cnt: " + cnt);  
    }  
}
```



cnt: -2147483648

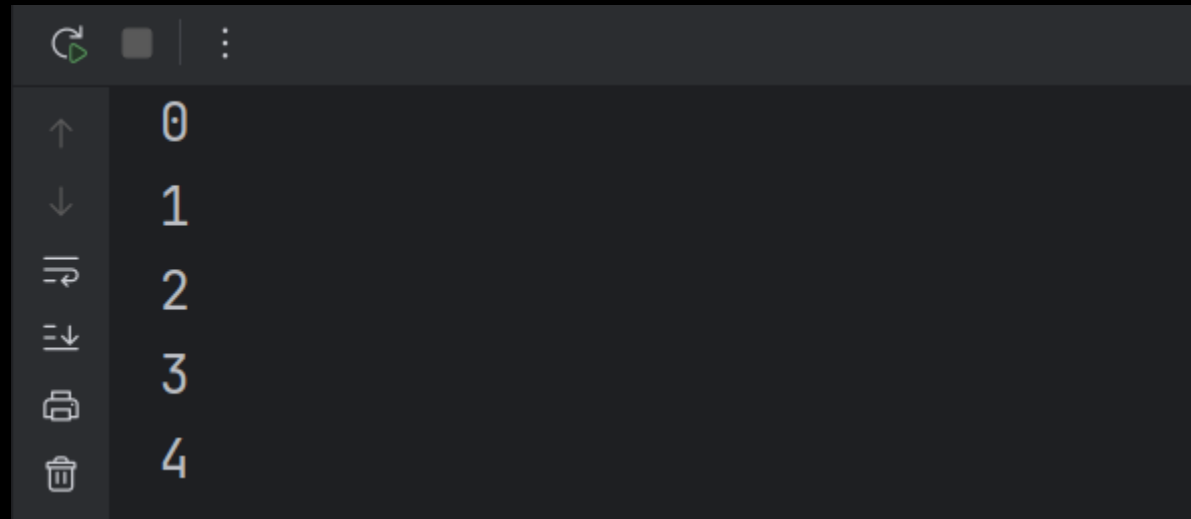
for

```
for(초기화; 조건식; 증감)
{
    구문1;
    구문2;
}
```

- **계수 기반 반복문**
- **반복 회수 조절**에 관련된 코드를 한 행에서 볼 수 있어 while문에 비해 상대적으로 실수 가능성이 적음
- 구구단 출력같은 코드가 필요한 상황에서 유리

12_forSample

```
public class Main {  
    public static void main(String[] args) {  
        for(int i = 0; i < 5; ++i) {  
            System.out.println(i);  
        }  
    }  
}
```



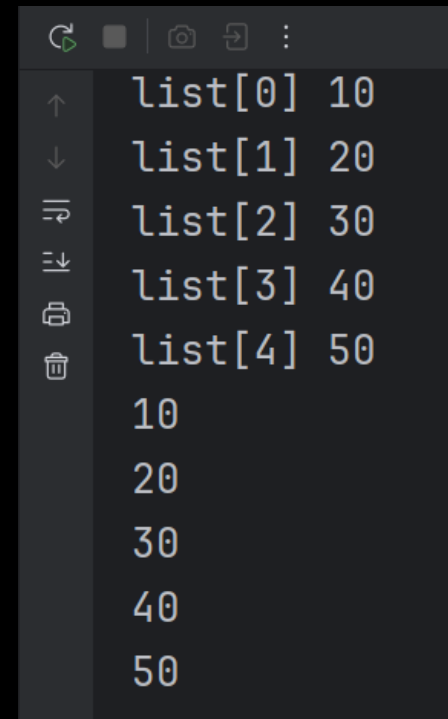
확장 for문(foreach)

```
for(타입 변수 : 배열)
{
    구문1;
    구문2;
}
```

- 배열 요소 기반 반복문이나 개수 및 인덱스를 계산할 필요가 없음
- 유지보수성이 매우 뛰어남

13_forEach

```
public class Main {  
    public static void main(String[] args) {  
        int[] list = {10, 20, 30, 40, 50};  
        for(int i = 0; i < list.length; ++i) {  
            System.out.println("list[" + i + "] " + list[i]);  
        }  
  
        for (int i : list) {  
            System.out.println(i);  
        }  
    }  
}
```



```
list[0] 10  
list[1] 20  
list[2] 30  
list[3] 40  
list[4] 50  
10  
20  
30  
40  
50
```

[필수실습 12-1]

총합 계산하기 (20분, 난이도3)

1~10까지 총합을 출력하는 프로그램을 작성. while문과 for문으로 각각 작성하며 반복회수는 10회로 제한. 반드시 총합을 누적하는 변수를 활용할 것

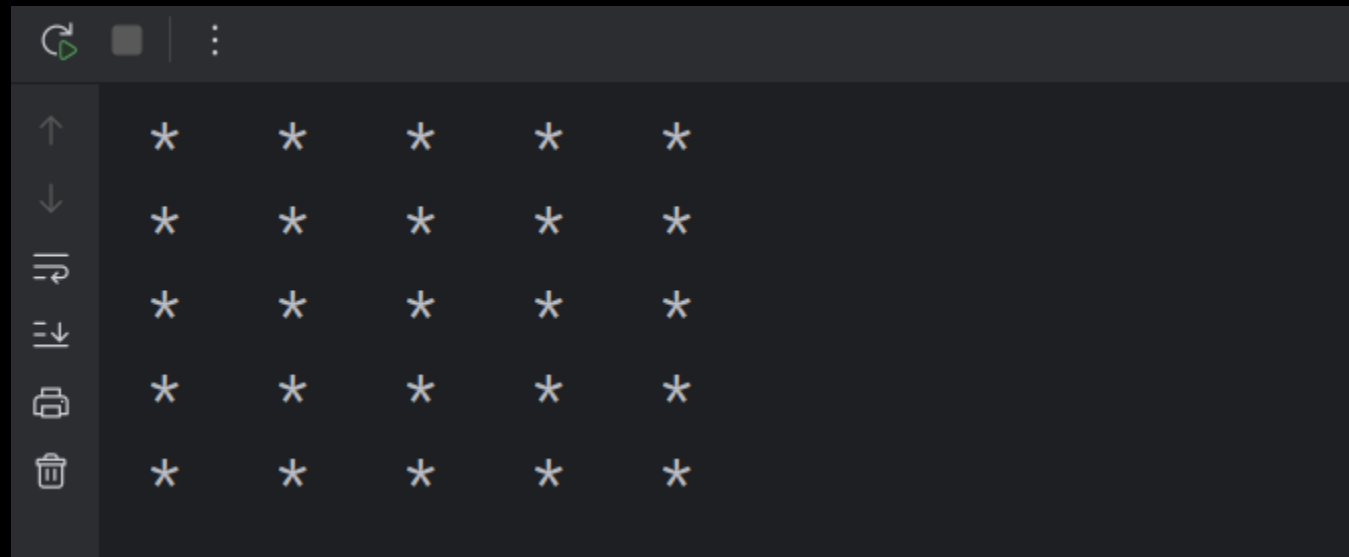
Total: 55

반복문의 중첩

- 1차원적 반복 출력 코드를 다시 반복문으로 묶어 2차원적 구조를 출력
- 반복에 대한 반복이 발생해 논리적으로 한 층 더 복잡

12_nestedFor

```
public class Main {  
    public static void main(String[] args) {  
        for(int i = 0; i < 5; ++i) {  
            for(int j = 0; j < 5; ++j)  
                System.out.print("*\t");  
            System.out.print('\n');  
        }  
    }  
}
```



[필수실습 12-2]

*출력 놀이 - 삼각형 기본 (20분, 난이도3)

아래 예처럼 삼각형을 *로 출력하는 프로그램을 작성. 반드시 for문을 중첩하는 구조로 개발.

```
*  
* *  
* * *  
* * * *  
* * * * *
```

[필수실습 12-3]

***출력 놀이 응용 (30분, 난이도3)**

아래 예처럼 삼각형을 *로 출력하는 프로그램을 작성.
'*' 왼쪽 여백은 Tab으로 조절

```

                *
            *   *
        *   *   *
    *   *   *   *
*   *   *   *   *
```

[필수실습 12-4]

***출력 놀이 응용 (30분, 난이도3)**

아래 예처럼 삼각형을 *로 출력하는 프로그램을 작성.
'*' 왼쪽 여백은 Tab으로 조절

```

                *
            *   *   *
        *   *   *   *   *
    *   *   *   *   *   *   *
*   *   *   *   *   *   *   *
```

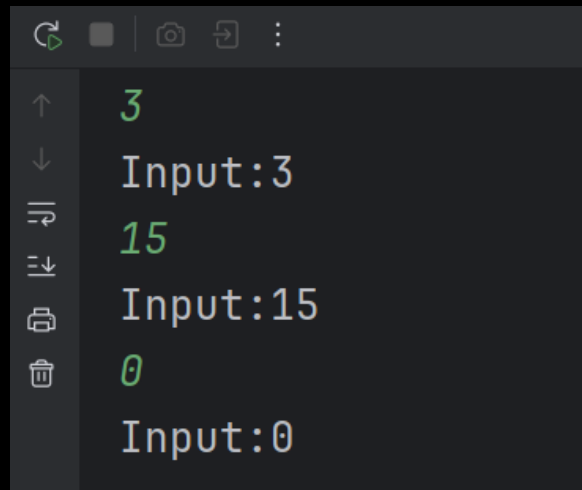

do while

- 반복 대상 구간코드가 조건식의 만족 여부와 관련 없이 무조건 한 번은 실행하는 반복문
- 기존 while문과 달리 조건식 뒤에 세미콜론이 있음

12_doWhileSample

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int input = 0;
        do {
            input = s.nextInt();
            System.out.println("Input:" + input);
        } while(input != 0);
    }
}
```

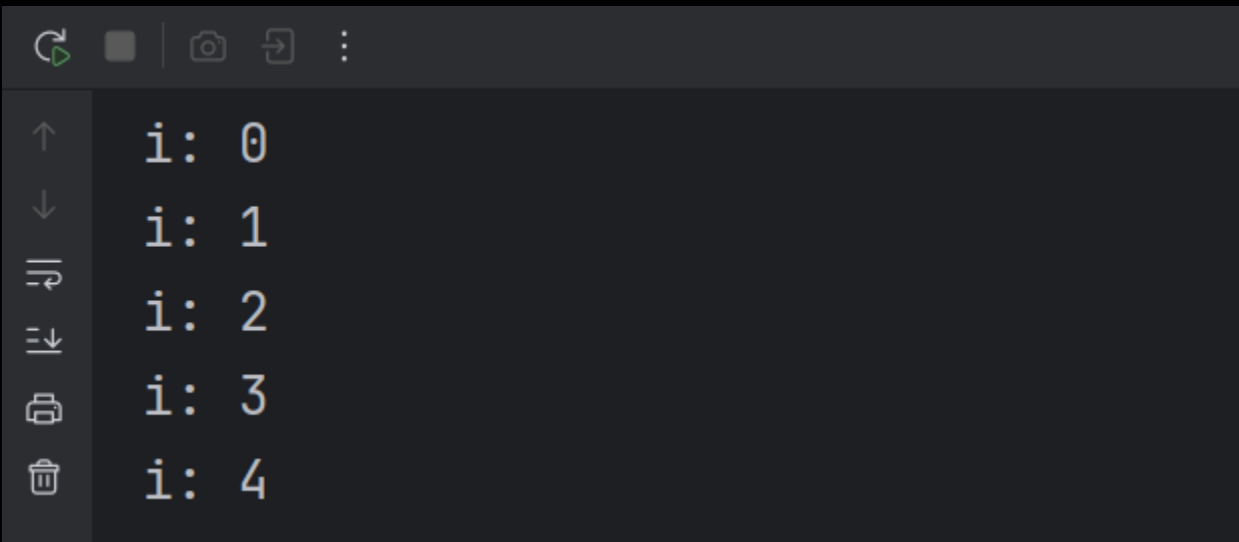


break

- 반복문과 switch-case문에 사용되며 수행 시 **스코프를 즉시 벗어나 그 다음 구문으로 이동**
- 반복문 내부에서 break 수행 시 조건과 상관 없이 **반복문 종료**

12_breakSample

```
public class Main {  
    public static void main(String[] args) {  
        for(int i = 0; i < 10; ++i) {  
            if(i > 4)  
                break;  
            System.out.println("i: " + i);  
        }  
    }  
}
```



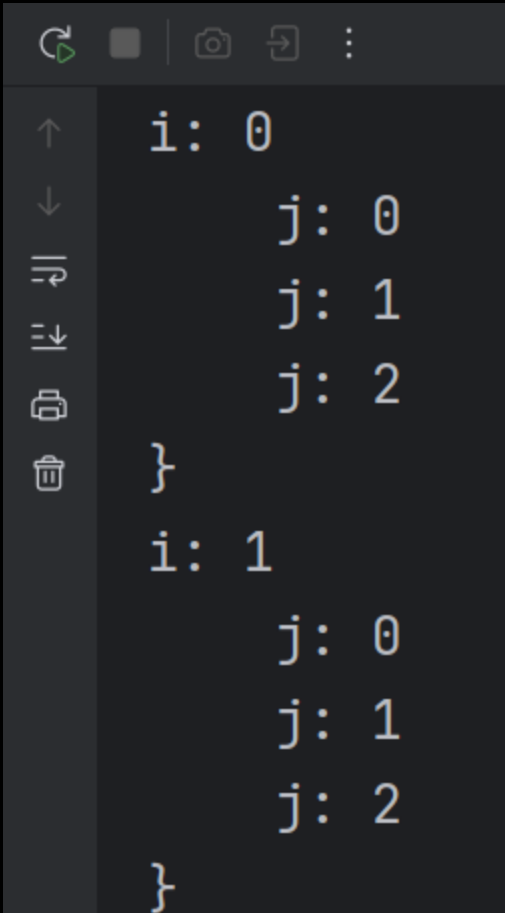
```
↑ i: 0  
↓ i: 1  
↺ i: 2  
↻ i: 3  
🗑 i: 4
```

break와 중첩 Loop

- 반복문이 중첩됐을 경우 내부 반복문에서 break문이 수행되면 내부 반복만 멈춤
- 외부 반복문은 계속 수행

12_nestedForBreak

```
public class Main {  
    public static void main(String[] args) {  
        for(int i = 0; i < 2; ++i) {  
            System.out.println("i: " + i);  
            for(int j = 0; j < 5; ++j) {  
                if (j > 2)  
                    break;  
                System.out.println("\tj: " + j);  
            }  
            System.out.println("{}");  
        }  
    }  
}
```



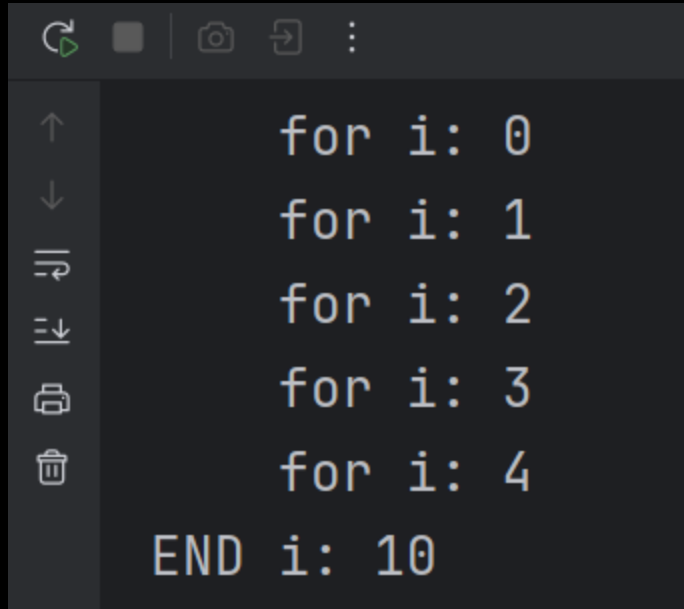
```
i: 0  
    j: 0  
    j: 1  
    j: 2  
}  
i: 1  
    j: 0  
    j: 1  
    j: 2  
}
```

continue

- break문과 달리 반복을 멈추지 않으며 반복문 내부에서 continue문 이후 코드를 수행하지 않고 그냥 넘어가 반복 시작 조건식 비교
- 반복은 계속되며 논리적으로 복잡해지는 원인이 될 수 있음

12_continueSample

```
public class Main {  
    public static void main(String[] args) {  
        int i = 0;  
        for(i = 0; i < 10; ++i) {  
            if(i > 4)  
                continue;  
            System.out.println("\tfor i: " + i);  
        }  
  
        System.out.println("END i: " + i);  
    }  
}
```



A terminal window with a dark background and light gray text. The window has a title bar with standard OS icons (back, forward, search, etc.). The output of the program is displayed as follows:

```
↑      for i: 0  
↓      for i: 1  
↶      for i: 2  
↷      for i: 3  
🖨      for i: 4  
🗑  
END i: 10
```


13. 배열과 프로그래밍

배열 (Array)

- 형식이 같은 자료 여러 개를 모아 한 덩어리로 관리하는 방법
- 여러 요소를 식별하기 위해 **인덱스**를 사용하며 0부터 시작 (**Zero-based index**)하고 최대 인덱스는 요소 개수-1
 - 배열의 경계를 벗어날 경우 오류 발생
- **배열의 이름**은 배열 객체에 대한 참조
- Java에서 배열은 객체(클래스)로 구현되어 있음
 - length
 - 지금은 깊은 이론적 이해보다는 실습이 중요

배열이 필요한 이유 (13_arrayNeeds)

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int a, b, c;
        a = s.nextInt();
        b = s.nextInt();
        c = s.nextInt();
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

배열 (Array) 객체

byte[] array



char array[]

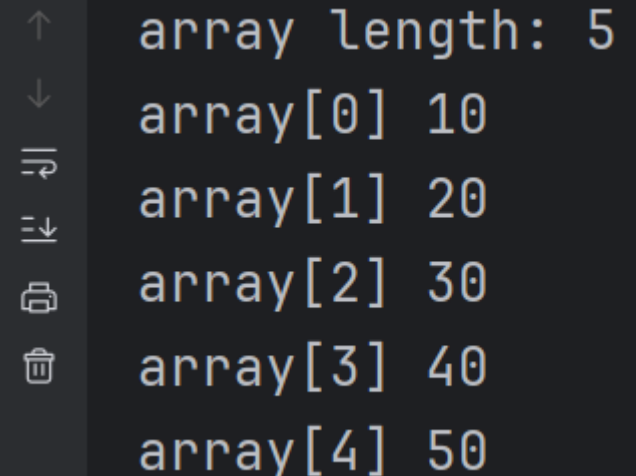


int[] array



13_arraySample

```
public class Main {  
    public static void main(String[] args) {  
        //int aData[] = {10, 20, 30, 40, 50};  
        int[] array = {10, 20, 30, 40, 50};  
        int idx = 0;  
        System.out.println("array length: " + array.length);  
        for(int data : array) {  
            System.out.println("array[" + idx + "]" + data);  
            idx++;  
        }  
    }  
}
```

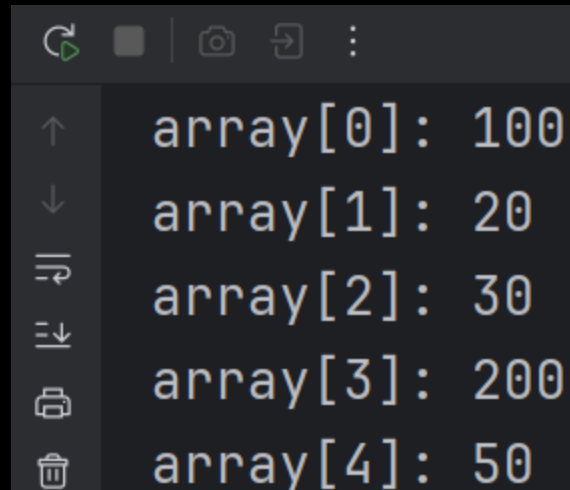


The image shows a terminal window with a dark background. On the left side of the terminal, there is a vertical toolbar containing several icons: an upward arrow, a downward arrow, a double arrow pointing right, a double arrow pointing left, a printer icon, and a trash can icon. To the right of these icons, the output of the Java program is displayed in a monospaced font. The output consists of six lines: "array length: 5", "array[0] 10", "array[1] 20", "array[2] 30", "array[3] 40", and "array[4] 50".

```
↑ array length: 5  
↓ array[0] 10  
⇨ array[1] 20  
⇩ array[2] 30  
🖨 array[3] 40  
🗑 array[4] 50
```

배열 인덱스에 유의 (13_arrayAccess)

```
public class Main {  
    public static void main(String[] args) {  
        int[] array = new int[] {10, 20, 30, 40, 50};  
  
        array[0] = 100;  
        array[3] = 200;  
        for (int i = 0; i < array.length; i++) {  
            System.out.printf("array[%d]: %d\n", i, array[i]);  
        }  
    }  
}
```



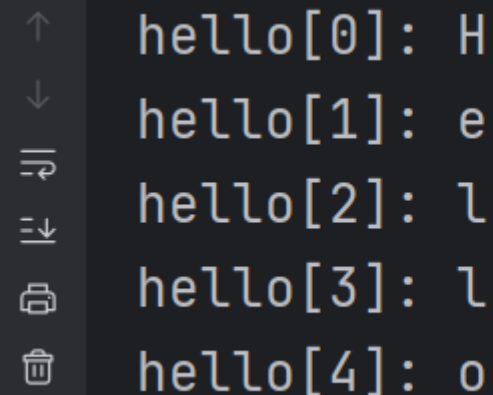
```
array[0]: 100  
array[1]: 20  
array[2]: 30  
array[3]: 200  
array[4]: 50
```

문자 배열 (문자열)

- 문자열의 실체는 `char[]`
- 문자열 상수란 쓰기가 허용되지 않는 이름이 없는 `char[]`로 이해할 수 있음
(통상 String 클래스로 처리)

13_charArray

```
public class Main {  
    public static void main(String[] args) {  
        char[] hello = new char[5];  
        String helloString = "Hello";  
        for (int i = 0; i < 5; i++) {  
            hello[i] = helloString.charAt(i);  
            System.out.printf("hello[%d]: %c\n", i, hello[i]);  
        }  
    }  
}
```



```
↑ hello[0]: H  
↓ hello[1]: e  
⇌ hello[2]: l  
⇌ hello[3]: l  
🖨 hello[4]: o  
🗑
```


[필수실습 13-1]

배열에서 최댓값 찾기 (20분, 난이도2)

for문을 사용해 int[5] 배열에 저장된 값 중 가장 큰 값을 찾아 출력하는 프로그램을 작성. 최댓값이 저장된 변수는 int max로 선언.

50 40 10 30 20

MAX: 50



[필수실습 13-2]

버블정렬 (30분, 난이도 3.5)

for문을 사용해 int[5] 배열에 저장된 값들을 오름차순으로 정렬하는 프로그램을 작성. 정렬 방식은 버블정렬 알고리즘을 사용



[필수실습 13-3]

선택정렬 (30분, 난이도 4)

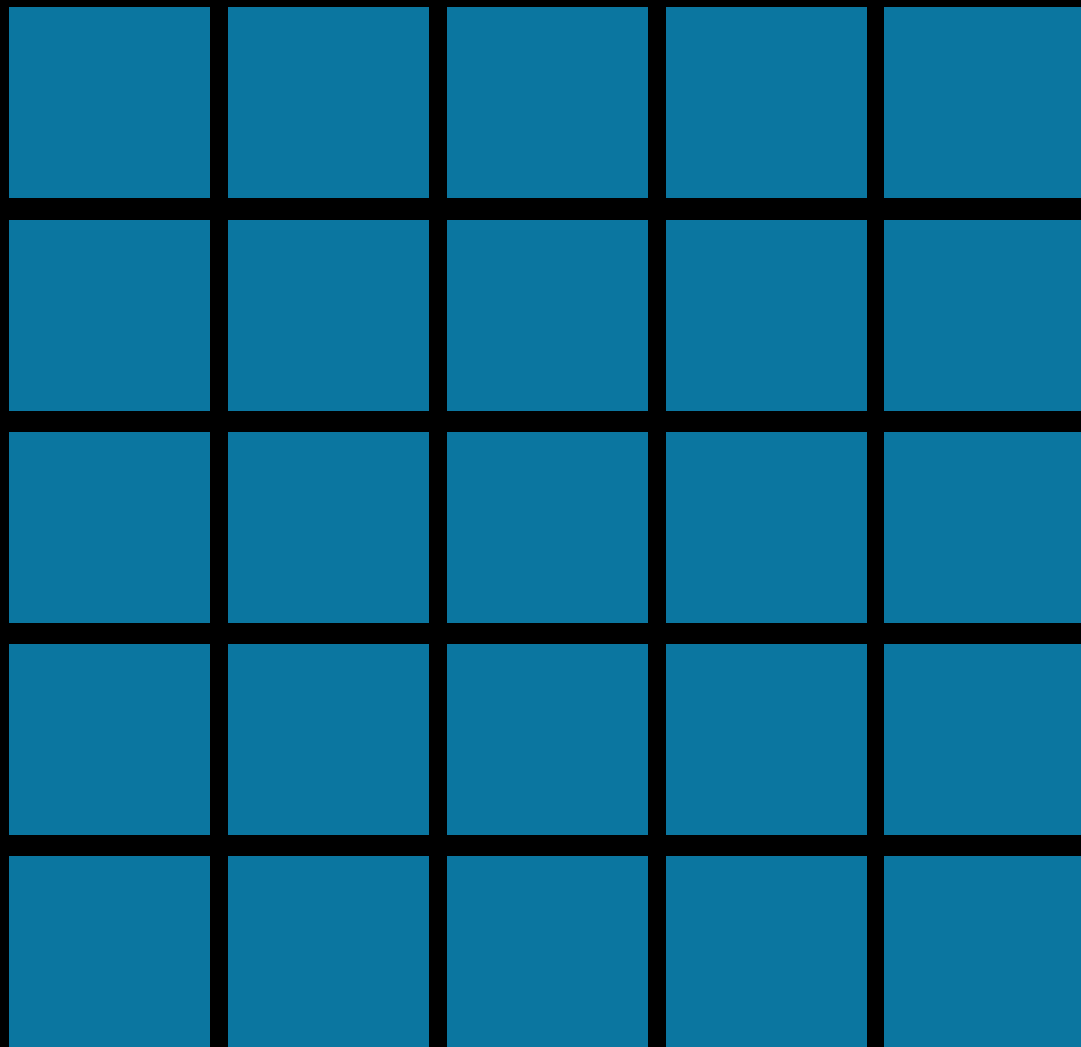
for문을 사용해 int[5] 배열에 저장된 값들을 오름차순으로 정렬하는 프로그램을 작성. 정렬 방식은 선택정렬 알고리즘을 사용.



다차원 배열

- 배열의 요소가 배열인 배열
- 2차원 구조일 경우 [행][열]
- 3차원 구조일 경우 [면][행][열]

2차원 배열



13_multiArray

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[][] array = {
            {10, 20, 30, 40},
            {50, 60, 70, 80},
            {90, 100, 110, 120}
        };

        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.printf("%d\t", array[i][j]);
            }
            System.out.print('\n');
        }
    }
}
```

↓	10	20	30	40
⇌	50	60	70	80
⇌	90	100	110	120
🖨				

[필수실습 13-4]

행과 열의 합 구하기 (30분, 난이도 3)

```
int [][]array = {  
    {10, 20, 30, 0},  
    {40, 50, 60, 0},  
    { 0,  0,  0,  0}  
};
```

상기 배열 요소의 행, 열 총합을 다음과 같이 출력하는 프로그램 작성.

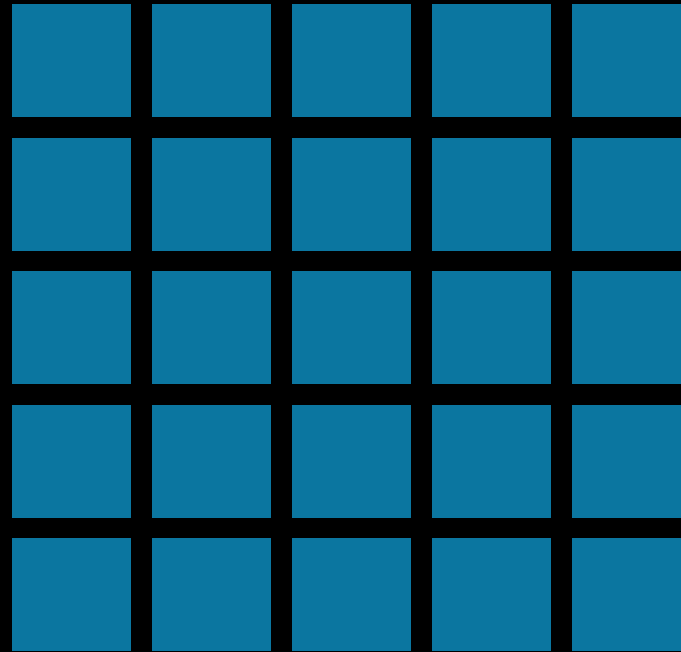
```
10 20 30 60  
40 50 60 150  
50 70 90 210
```

[필수실습 13-5]

2차원 배열 순차 채우기 (30분, 난이도 3)

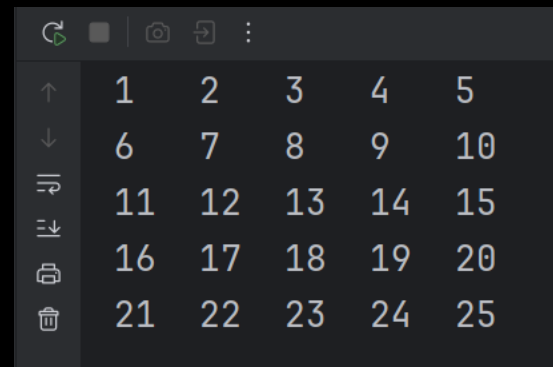
for문을 사용해 int[5][5] 배열에 다음과 같이 저장하는 프로그램을 작성.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



2차원 배열 순차 채우기 (13_multiArrayCounter)

```
public static void main(String[] args) {  
    int[][] array = new int[5][5];  
  
    int cnt = 0;  
    for (int i = 0; i < array.length; i++) {  
        for (int j = 0; j < array[i].length; j++)  
            array[i][j] = ++cnt;  
    }  
  
    for (int i = 0; i < array.length; i++) {  
        for (int j = 0; j < array[i].length; j++) {  
            System.out.printf("%d\t", array[i][j]);  
        }  
        System.out.print('\n');  
    }  
}
```



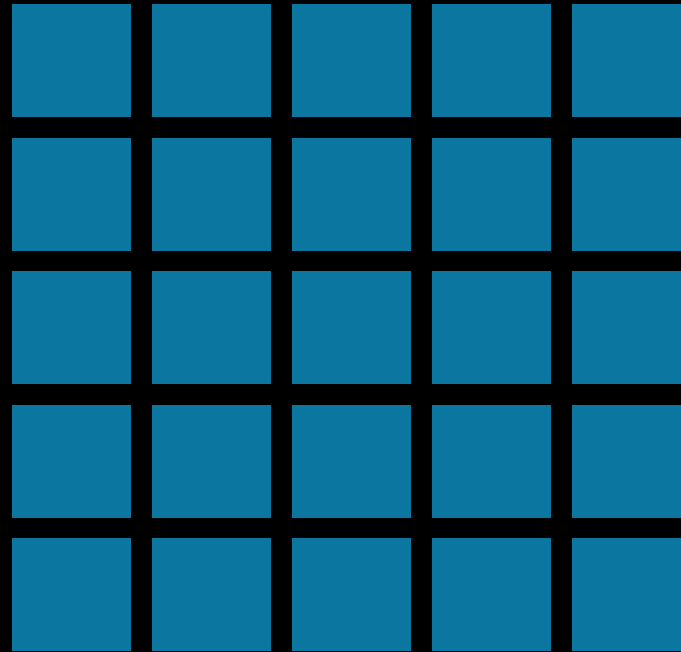
↑	1	2	3	4	5
↓	6	7	8	9	10
↺	11	12	13	14	15
↻	16	17	18	19	20
↵	21	22	23	24	25

[필수실습 13-6]

교차 (30분, 난이도 3.5)

for문을 사용해 int[5][5] 배열에 다음과 같이 저장하는 프로그램을 작성.

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16
21	22	23	24	25

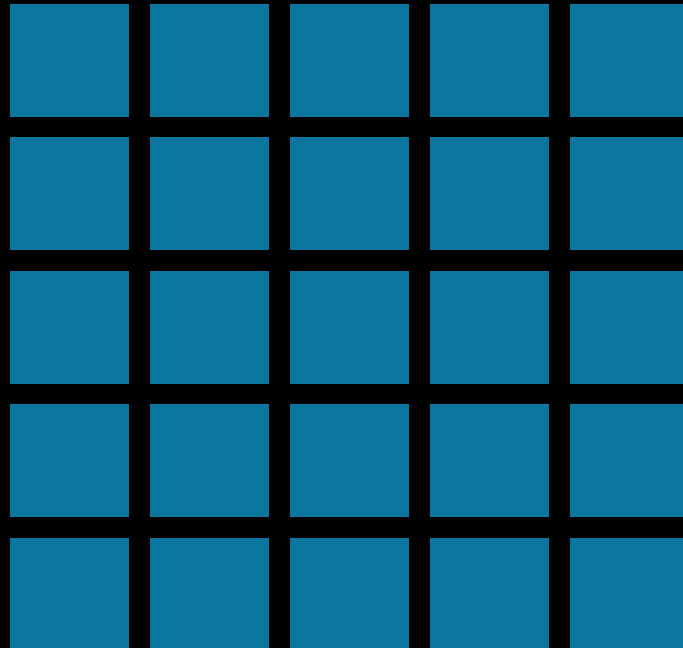


[필수실습 13-7]

좌절 금지 달팽이 (60분+?, 난이도 5)

for문을 사용해 int[5][5] 배열에 다음과 같이 저장하는 프로그램을 작성.
(달팽이 배열)

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9



Lookup 배열 (중요)

- 데이터를 검색 혹은 참조하기 위해 사용하는 배열
- 기준에 따라 인덱스 값을 계산하고 식별한 요소를 활용하는 구조
- 배열의 한 요소가 처리할 하나의 경우로 활용
- switch-case 구조의 단점을 획기적으로 개선할 수 있는 방법

필수실습 11-3 연령별 버스요금 계산 코드

```
if(age < 14) {  
    if(age < 4)  
        rate = 0.0;  
    else  
        rate = 0.5;  
}  
else {  
    if(age >= 20)  
        rate = 1.0;  
    else  
        rate = 0.75;  
}
```

```
double fee = 1000 * rate;  
System.out.printf("나이: %d, 최종요금: %d", age, (int)fee);
```

13_lookupArray

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        double[] rateArray = {
            0.0, 0.1, 0.25,
            0.5, 0.5,
            0.6, 0.65,
            0.8, 0.82, 0.97
        };
    }
}
```

```
final int fee = 1000;
System.out.println("요금표");
for (int i = 1; i <= 10; i++) {
    System.out.printf("%d세 요금: \t%d원\n",
        i, (int)(fee * rateArray[i - 1]));
}
```

```
System.out.print("나이를 입력하세요: ");
int age = s.nextInt();
if(age < 1)
    age = 1;
else if(age > 10)
    age = 10;
```

```
System.out.printf("요금: %d원\n",
    (int)(fee * rateArray[age - 1]));
```

```
}
```

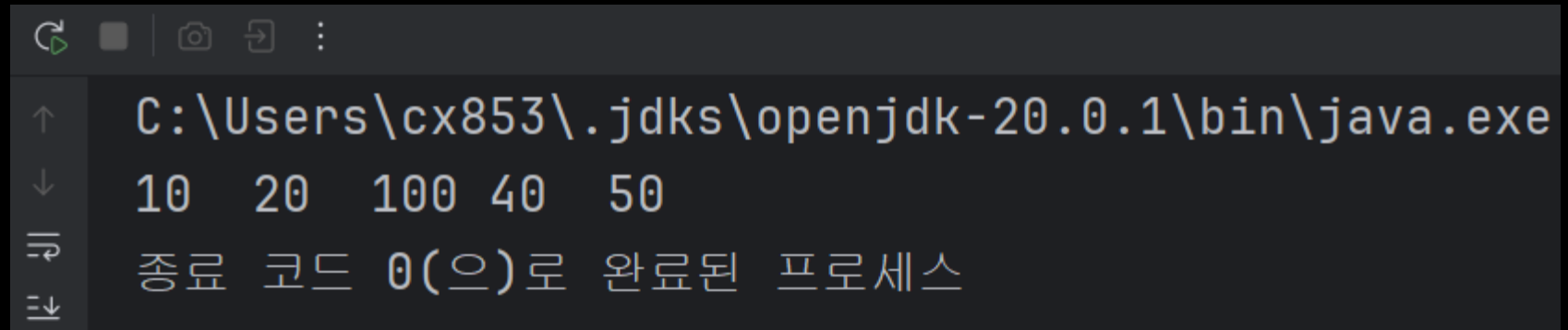
```
}
```

배열의 복사

- Java에서 배열은 객체(클래스)로 구현되어 있음
- 배열 객체에 대한 식별자는 결국 참조자
- 한 객체에 대해 여러 참조자가 존재할 수 있음
- Deep copy, Shallow copy 문제에 주의!

배열 얕은 복사 문제 (13_arrayShallowCopy)

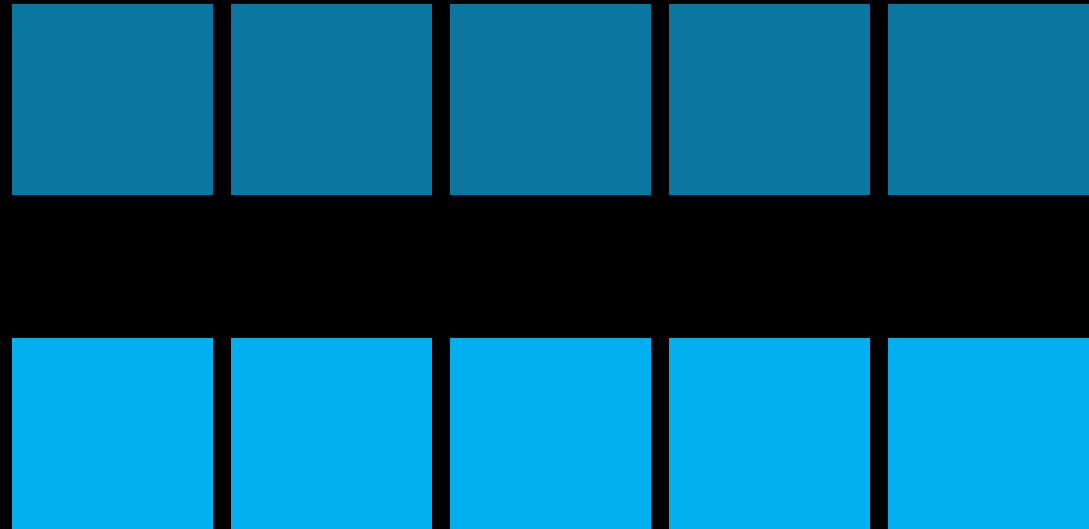
```
public class Main {  
    public static void main(String[] args) {  
        int[] aSrc = {10, 20, 30, 40, 50};  
        int[] aDst = aSrc;  
  
        aDst[2] = 100;  
        for(int data : aSrc)  
            System.out.print(data + "\t");  
    }  
}
```



```
C:\Users\cx853\.jdk\openjdk-20.0.1\bin\java.exe  
10 20 100 40 50  
종료 코드 0(으)로 완료된 프로세스
```

배열 깊은 복사 (13_arrayDeepCopy)

```
public class Main {  
    public static void main(String[] args) {  
        int[] aSrc = {10, 20, 30, 40, 50};  
  
        int[] aDst01 = new int[aSrc.length];  
        for (int i = 0; i < aSrc.length; i++) {  
            aDst01[i] = aSrc[i];  
        }  
  
        aDst01[2] = 100;  
        ...  
    }  
}
```

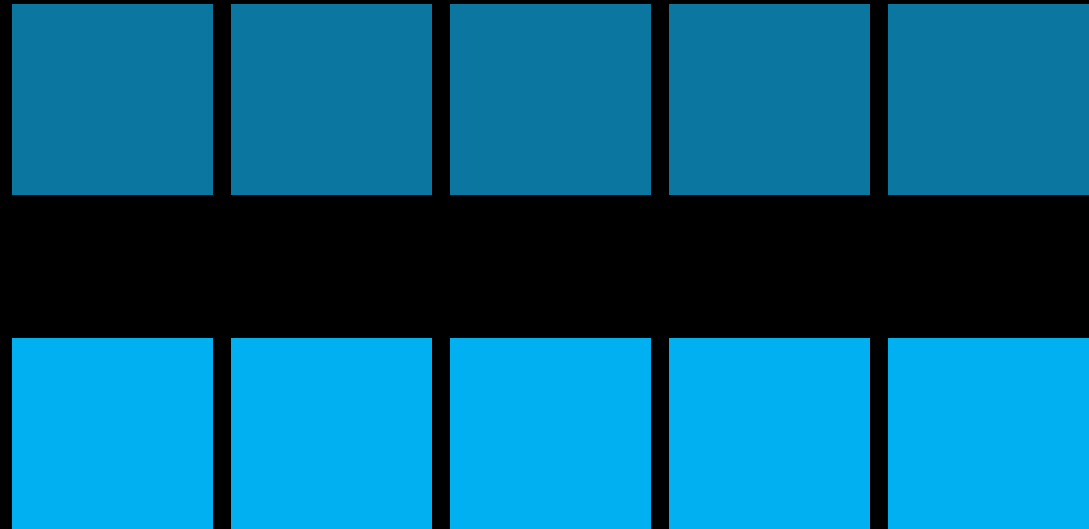


```
int[] aDst02 = aSrc.clone();
aDst02[3] = 300;
System.out.print("\n  aSrc[]: ");
for(int data : aSrc)
    System.out.print(data + "\t");

System.out.print("\naDst02[]: ");
for(int data : aDst02)
    System.out.print(data + "\t");
```

```
}
```

```
}
```



배열을 Deep copy하는 다른 방법

- `Array.clone()`과 비슷한 `Arrays.copyOf()` 활용
 - `Arrays.copyOfRange()`
- `System.arraycopy()` 활용

System.arraycopy()

```
public class Main {  
    public static void main(String[] args) {  
        int[] aSrc = {10, 20, 30, 40, 50};  
        int[] aDst = new int[aSrc.length];  
  
        System.arraycopy(aSrc, 0, aDst, 0, aDst.length);  
  
        aSrc[2] = 100;  
        System.out.print("aSrc[]: ");  
        for(int data : aSrc)  
            System.out.print(data + "\t");  
  
        System.out.print("\naDst[]: ");  
        for(int data : aDst)  
            System.out.print(data + "\t");  
    }  
}
```

```
↑  
↓  
⇒ aSrc[]: 10 20 100 40 50  
⇒ aDst[]: 10 20 30 40 50  
⇒ 종료 코드 0(으)로 완료된 프로세스
```

14. 메서드 (함수)

(사용자 정의) 메서드 (함수)

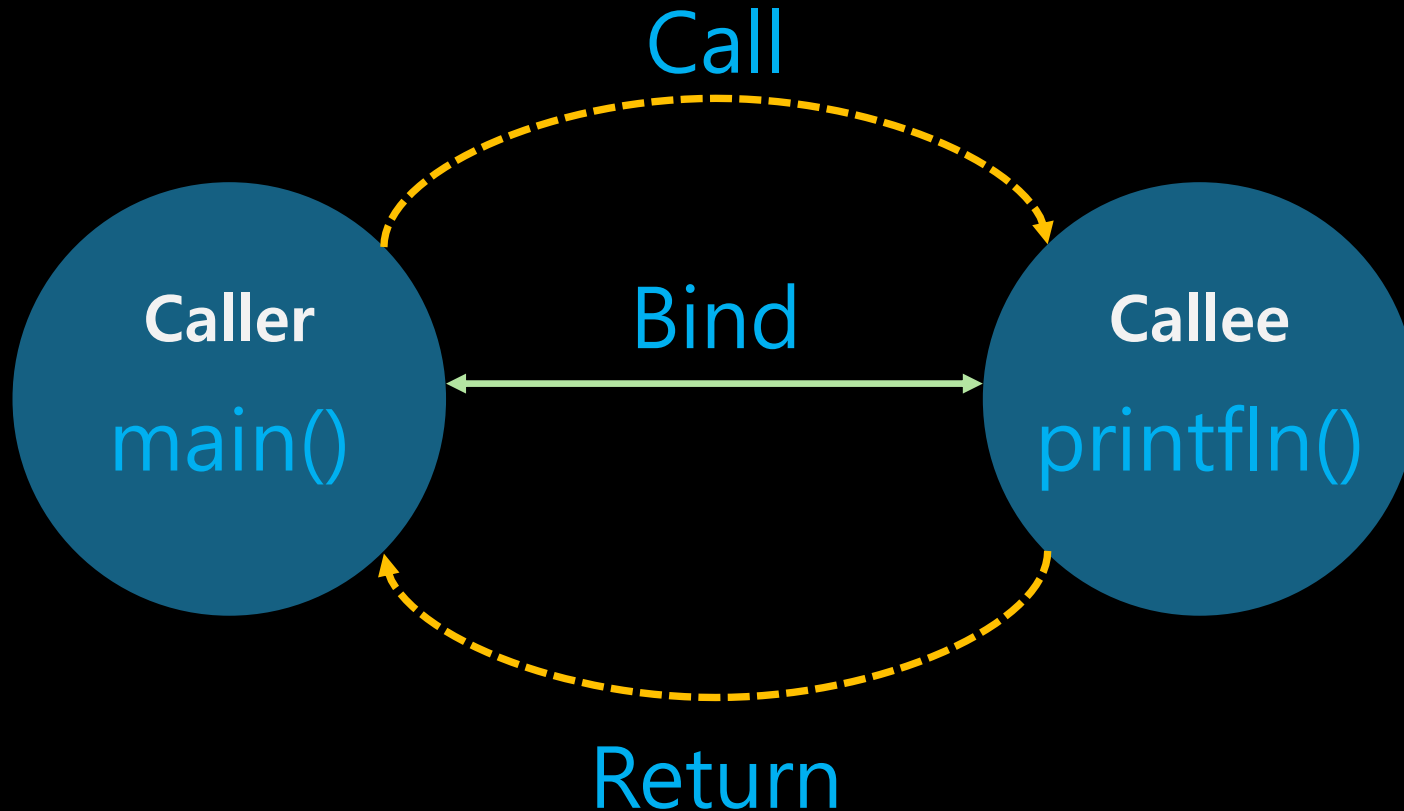
- 절차적 흐름을 갖는 여러 구문을 하나로 묶어({ }) 사용하는 단위 코드
- 클래스를 구성하는 요소 중 하나
- Main 클래스 main() 함수의 시작과 끝이 곧 프로그램의 시작과 끝
- 여러 메서드가 존재 할 경우 호출을 통해 연결 (bind)

문법 특성

- 함수는 반환자료형 이름 (매개변수 목록) 형식으로 기술
- 호출자 함수와 피호출자 함수로 관계를 규정 할 수 있음 (Binding)
- 호출자는 피호출자 함수의 매개변수 초깃값을 기술해야 할 의무가 있음
- 피호출자 함수는 호출자 함수에게 값을 반환
 - return문으로 반환 시점 변경가능
 - 함수가 끝나면 return문이 없어도 반환

기초 이론과 문법

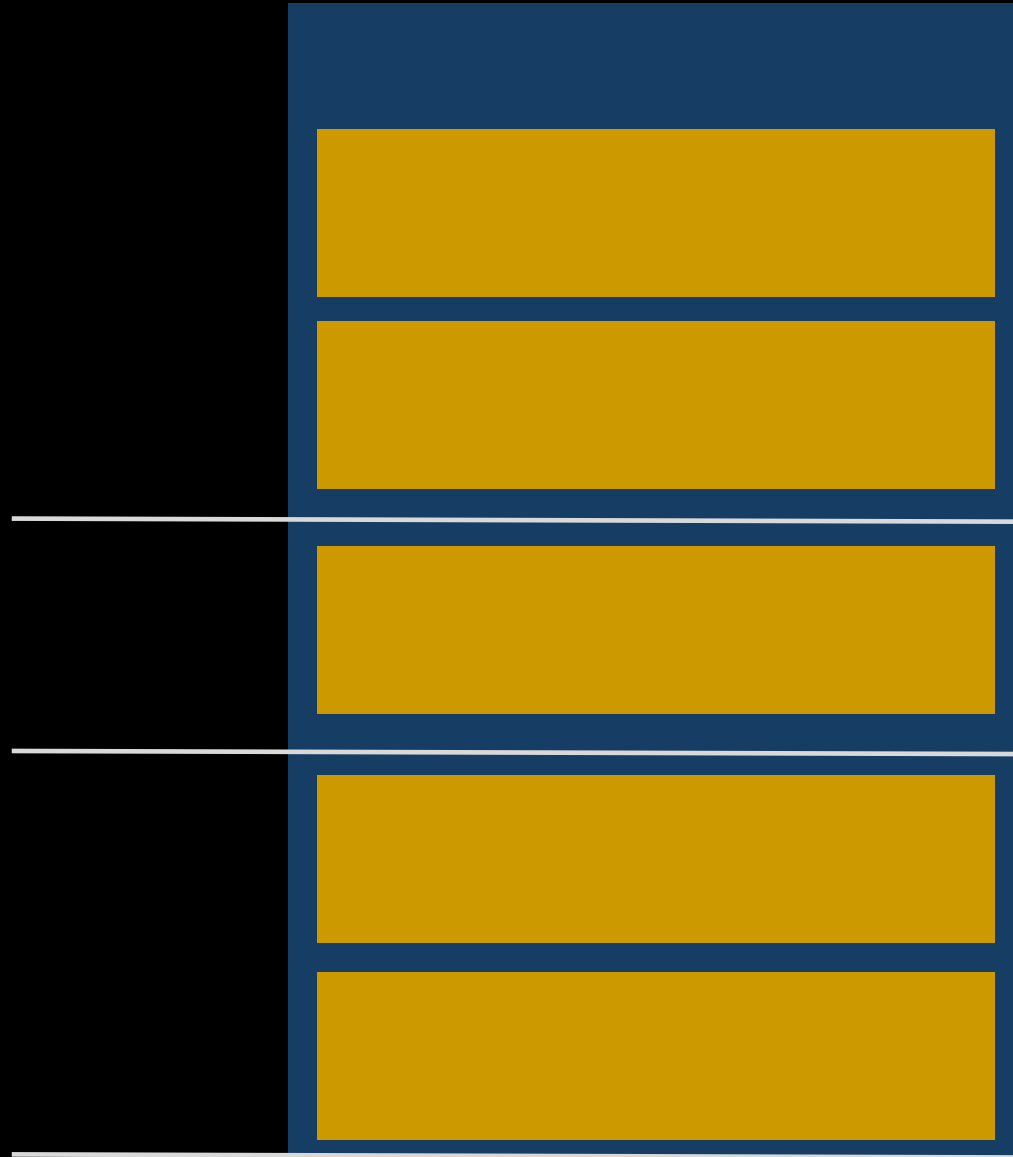
```
System.out.println("Hello");
```



14_functionSample

```
public class Main {  
    public static void main(String[] args) {  
        int result = add(3, 4);  
        System.out.println("result: " + result);  
    }  
  
    public static int add(int a, int b) {  
        int data = 0;  
        data = a + b;  
        return data;  
    }  
}
```

함수 호출과 스택 변화



main()

int a, b;

testFunc()

testFunc(int c)

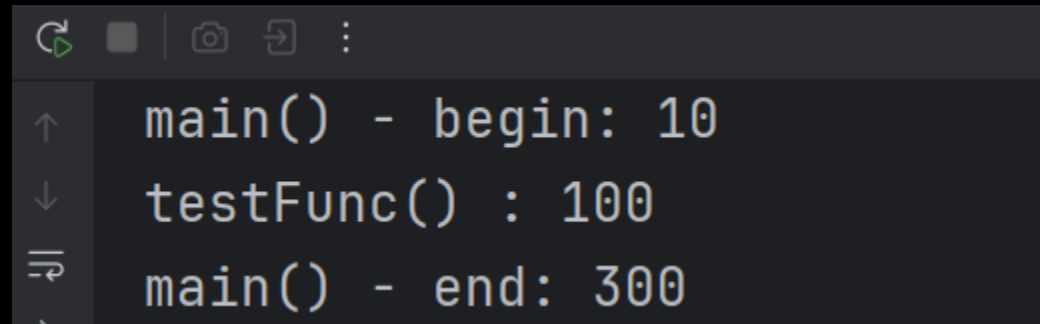
add(1, 2)

멤버 변수와 식별자 검색 순서

- **멤버 변수**는 메서드처럼 클래스를 이루는 멤버
- 클래스에 속한 모든 메서드가 접근 가능하며 서로 데이터를 공유 할 수 있음
- 식별자 검색 순서
 1. 지역 **스코프**
 2. 최대 **함수 바디**
 3. **클래스 멤버**까지 확장

메서드간 자료 공유(14_memberData)

```
public class Main {  
    static int testData = 10;  
  
    public static void main(String[] args) {  
        System.out.println("main() - begin: " + testData);  
        Main.testFunc();  
        System.out.println("main() - end: " + testData);  
    }  
  
    public static void testFunc() {  
        int testData = 100;  
        Main.testData = 300;  
        System.out.println("testFunc() : " + testData);  
    }  
}
```

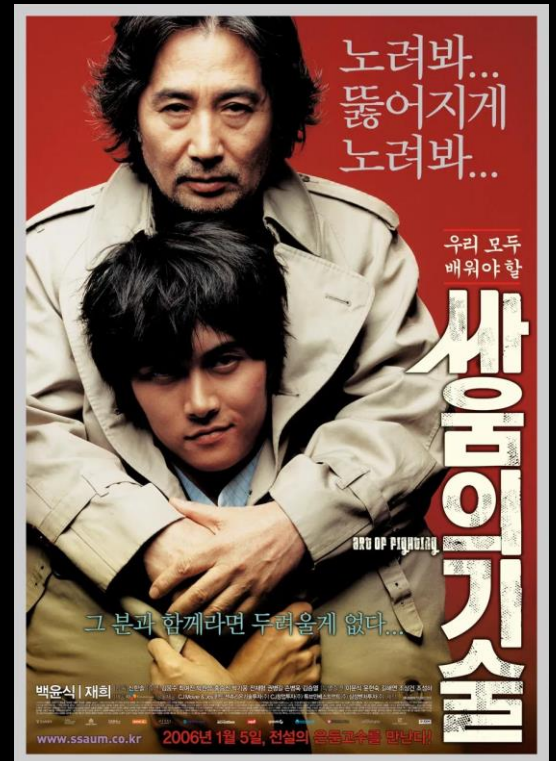
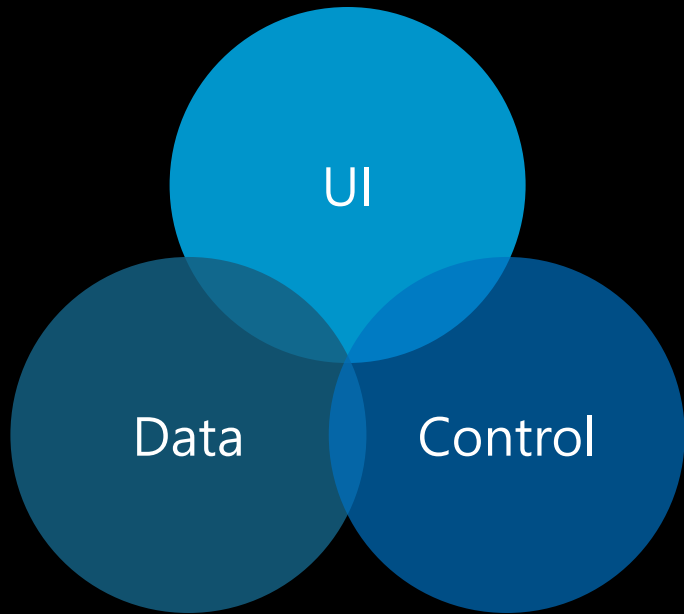


The screenshot shows a console window with the following output:

```
main() - begin: 10  
testFunc() : 100  
main() - end: 300
```

기초적인 두 가지 함수 설계 원칙

- UI와 기능은 반드시 분리
- 재사용 가능한 단위 코드는 함수로 구현 (DRY 원칙)



너 피동싸고 기저귀 찬다?

UI와 기능이 혼합된 경우(14_funcDesignBefore)

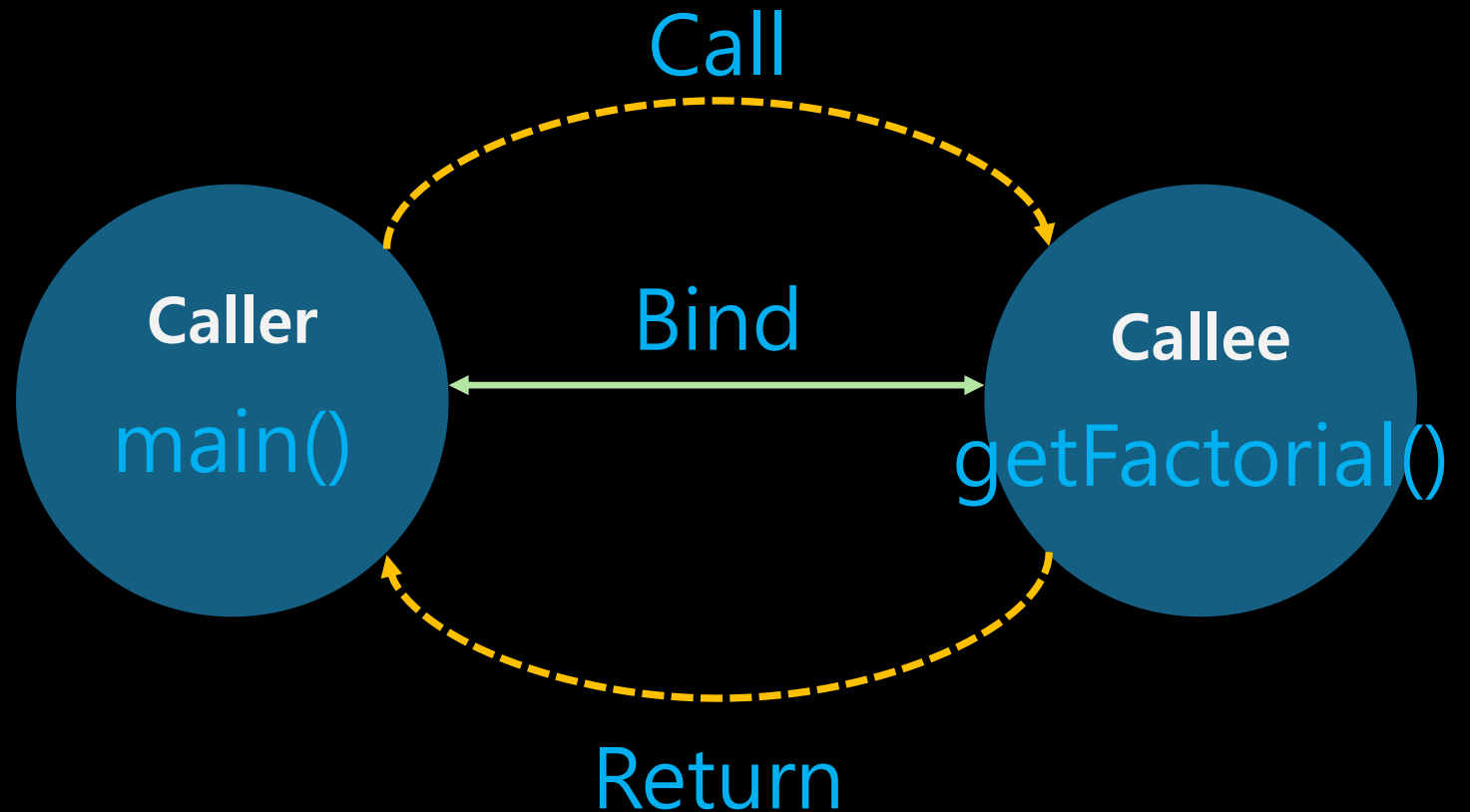
```
public static int getFactorial(int param) {  
    if(param < 1 || param > 10) {  
        System.out.println("ERROR: 1~10 사이 정수를 입력하세요.");  
        return 0;  
    }  
  
    int result = 1;  
    for (int i = 1; i <= param; i++)  
        result *= i;  
    return result;  
}
```

UI와 기능이 분리된 경우(14_funcDesignAfter)

```
public static void main(String[] args) {  
    Scanner s = new Scanner(System.in);  
    System.out.print("계승계산을 위한 값(1~10): ");  
    int input = s.nextInt();  
    if(input < 1 || input > 10) {  
        System.out.println("ERROR: 1~10 사이 정수를 입력하세요.");  
        return;  
    }  
  
    System.out.println("factorial: "+ getFactorial(input));  
}
```



```
public static int getFactorial(int param) {  
    if(param < 1 || param > 10)  
        return 0;  
  
    int result = 1;  
    for (int i = 1; i <= param; i++)  
        result *= i;  
    return result;  
}
```



14_funcDesignGrade

```
public class Main {  
    public static void main(String[] args) {  
        int score = getScore();  
        if(score < 0 )  
            return;  
  
        char ch = getGrade(score);  
        System.out.println("학점: " + ch);  
    }  
}
```

```
public static int getScore() {  
    System.out.println("점수: ");  
    Scanner s = new Scanner(System.in);  
    int score = s.nextInt();  
    if(score < 0 || score > 100) {  
        System.out.println("ERROR: 점수범위가 부적절합니다.");  
        return -1;  
    }  
    return score;  
}
```

```
public static char getGrade(int score) {  
    if (score >= 90) return 'A';  
    else if (score >= 80) return 'B';  
    else if (score >= 70) return 'C';  
    else if (score >= 60) return 'D';  
    return 'F';  
}
```

[필수실습 14-1]

최댓값을 반환하는 함수(메서드) (20분, 난이도 3)

사용자로부터 세 정수를 입력 받아 최댓값을 반환하는 함수(getMax())를 작성. 사용자 입력을 받는 부분(main())과 최댓값을 계산하는 코드는 반드시 별도 함수로 분리.

이벤트 루프

- `main()` 함수에서 사용자 인터페이스 출력 및 사용자 입력을 반복하는 구조
- 보통 메뉴출력과 사용자 선택을 확인
- 메뉴 선택에 따라 기능 수행
- 대부분의 응용 프로그램이 채택하는 일반적 구조

14_eventLoop

```
public static void main(String[] args) {  
    int input = 0;  
  
    for(printMenu(); (input = getCmdNumber()) != 0; printMenu())  
    {  
        if(input == 1) {  
            System.out.println("New menu");  
        }  
        else if(input == 2) {  
            System.out.println("Search menu");  
        }  
    }  
    System.out.println("Bye");  
}
```

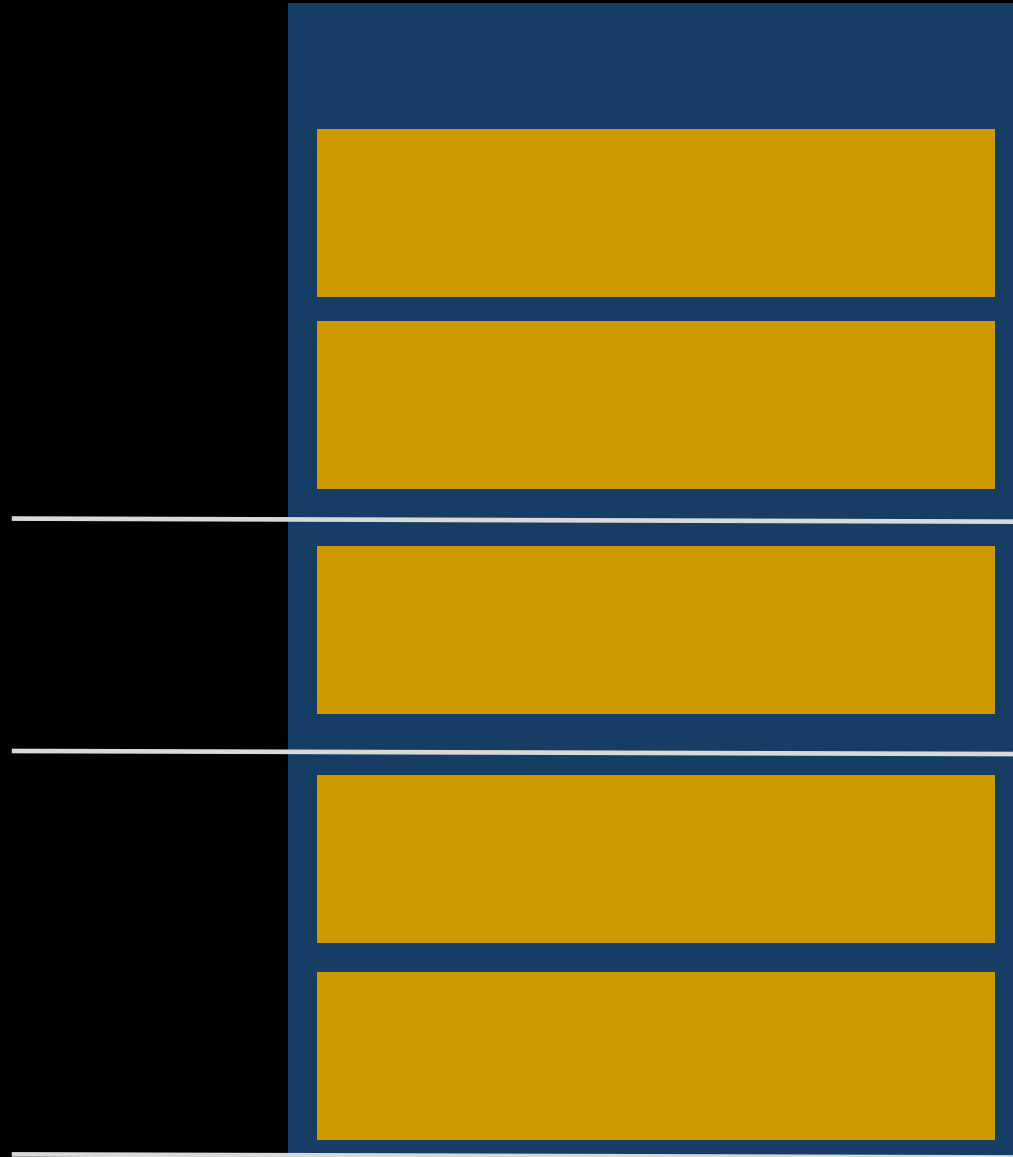
```
public static void printMenu() {  
    System.out.println("[1]New\t[2]Search\t[3]Print\t[0]Exit");  
    System.out.print("Command(0~3): ");  
}
```

```
public static int getCmdNumber() {  
    Scanner s = new Scanner(System.in);  
    int cmd = s.nextInt();  
    return cmd;  
}
```

매개변수 전달 기법

- Call by value
- Call by reference
 - C언어에서는 참조형을 포인터로 구현
- 인수, 매개변수, 파라미터, 아규먼트 등은 다 같은 말
- 매개변수는 Stack 영역을 사용하는 지역변수
- new 연산으로 생성한 객체는 모두 Heap 영역 인스턴스
- Java에서 String은 기본형식처럼 취급되는 경향이 있음에 유의

함수 호출과 스택 변화



main()

int a, b;

testFunc()

testFunc(int c)

add(1, 2)

Call by value

```
public class Main {  
    public static void main(String[] args) {  
        int result = add(3, 4);  
        System.out.println("result: " + result);  
    }  
  
    public static int add(int a, int b) {  
        return a + b;  
    }  
}
```

Call by reference

```
public static void main(String[] args) {  
    int[] array = new int[] {10, 20, 30, 40, 50};  
    printArray(array);  
    modifyArray(array);  
    printArray(array);  
}
```

```
public static void modifyArray(int[] param) {  
    for (int i = 0; i < param.length; i++) {  
        param[i] = (i + 1) * 100;  
    }  
}
```

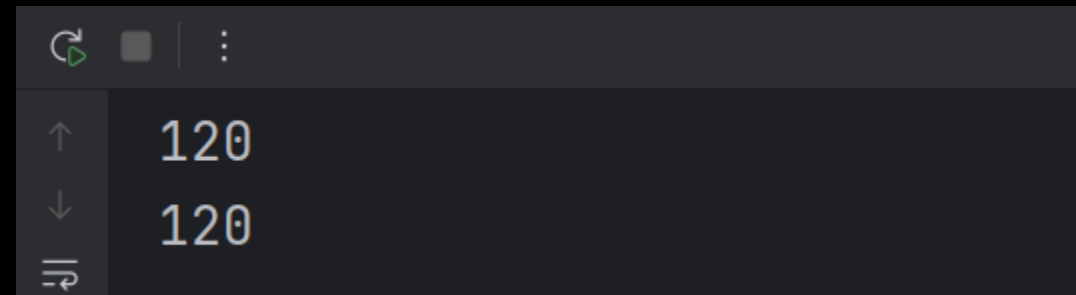


재귀 호출

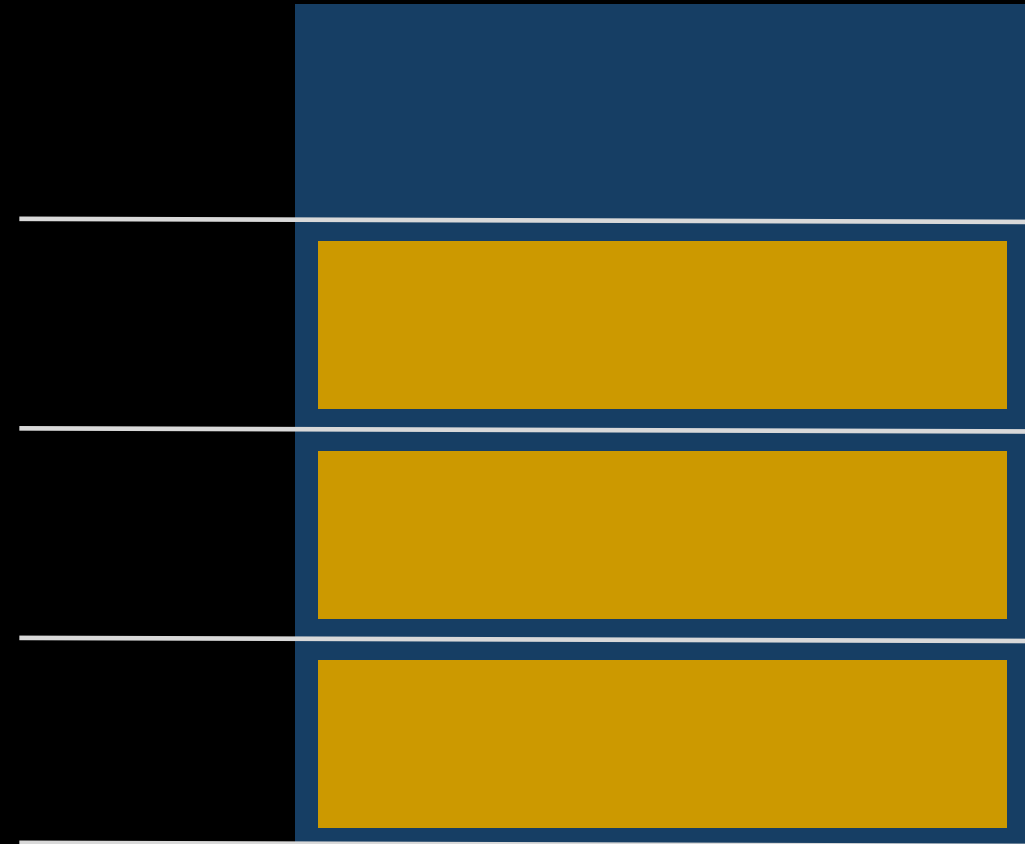
- 함수 코드 내부에서 다시 자신을 호출하는 것
- 반복문과 Stack 자료구조를 합친 것
- 비선형 자료구조에서 매우 중요하게 활용
- 함수 호출 오버헤드는 감수
- 논리 오류 발생 시 Stack overflow 발생
 - 무한루프와 같은 원리

14_recursiveCall

```
public static void main(String[] args) {  
    System.out.println(getFactorialLoop(5));  
    System.out.println(getFactorial(5));  
}  
  
public static int getFactorialLoop(int param) {  
    int total = 1;  
    for (int i = 1; i <= param; i++)  
        total *= i;  
    return total;  
}
```



```
public static int getFactorial(int param) {  
    if(param == 1)  
        return 1;  
  
    int result = param * getFactorial(param - 1);  
    return result;  
}
```



[필수실습 14-2]

문자열을 뒤집어 출력하는 메서드 (30분, 난이도 4)

사용자로부터 (영문)문자열을 입력 받은 후 한 글자씩 출력하되 입력한 문자열을 거꾸로 뒤집어 출력. 한 글자씩 출력하는 메서드는 반드시 재귀호출 구조로 구현 (charAt())

Hello

olleH

다중 정의 (Overloading)

- public static int
add(int a, int b)
- public static int
add(double a, double b)
- 메서드의 이름은 같지만 매개변수 구성이 다름
 - 반환형식은 고려하지 않음
- 다양한 형식에 대해 같은 기능을 제공하는 상황에서 자주 사용

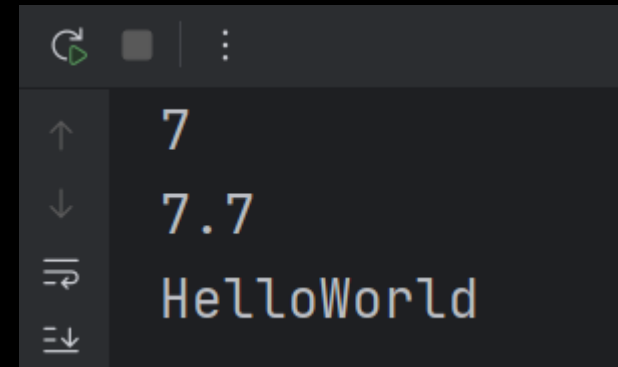
add() 메서드 다중 정의

```
public static void main(String[] args) {  
    System.out.println(add(3, 4));  
    System.out.println(add(3.3, 4.4));  
    System.out.println(add("Hello", "World"));  
}
```

```
public static int add(int a, int b) {  
    return a + b;  
}
```

```
public static double add(double a, double b) {  
    return a + b;  
}
```

```
public static String add(String a, String b) {  
    return a + b;  
}
```



학습을 마무리 하면서...

- 절차를 기술하는 (문법이 있는) 글쓰기
 - 변수, 상수, 경우의 수
 - 각종 연산
 - 흐름 제어(조건, 계수, 반복)
 - 배열 (자료)구조 (Lookup)
 - 함수 (메서드)
- **이제는 객체를 다룰 준비가 끝났음!**

감사합니다!