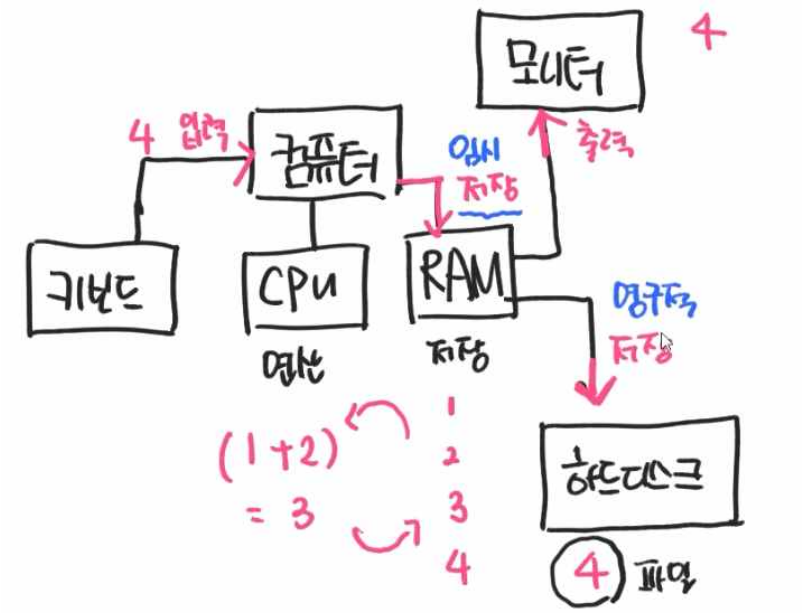
Chapter 1. 자바 JVM 이해하기

1장. JDK 란

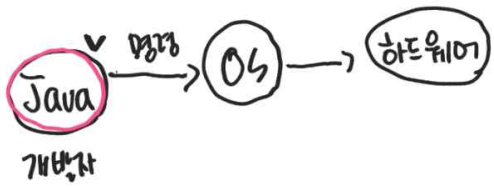
ⓐ 컴퓨터의 구조



▷ RAM : 전류이용 데이터를 저장 → 컴퓨터가 종료되면 데이터가 소멸

▷ HDD : 스크레치를 이용해 기록한다. → 컴퓨터가 종료되도 데이터가 소멸되지 않는다.

ⓑ 키보드, CPU, RAM, 하드디스크 등 하드웨어를 사람이 직접 건드리는 것이 불가능하기 때문에 운영체제가 중간에서 인터페이스 역할을 함.



※ OS는 사용자의 명령을 받아서 하드웨어를 직접 제어한다.

→ 개발자는 자바라는 언어를 이용해서 OS(운영체제)에게 명령을 내린다.

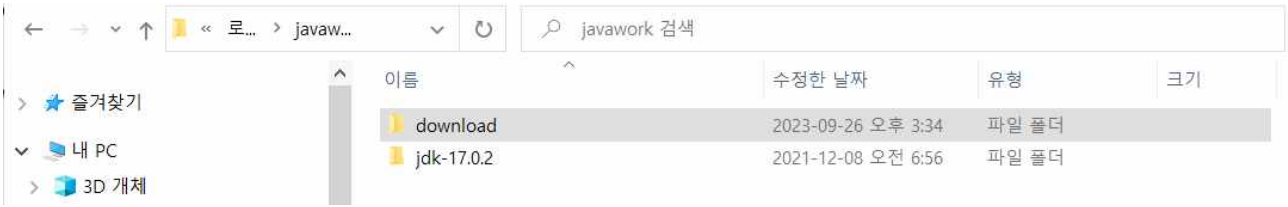
→ 명령은 JDK(Java Development Kit)를 사용해서 운영체제에 명령을 내린다.

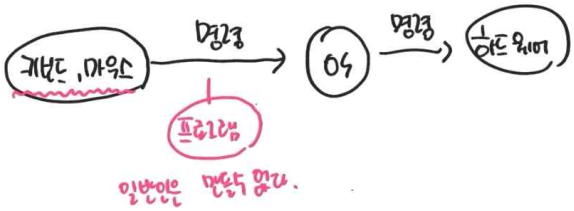
2장. JDK 설치(jdk.java.net/archive)

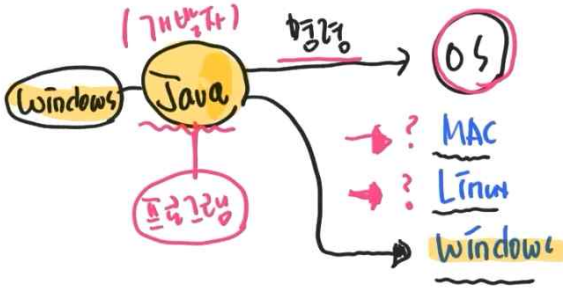
ⓐ c:/javawork 디렉토리 생성

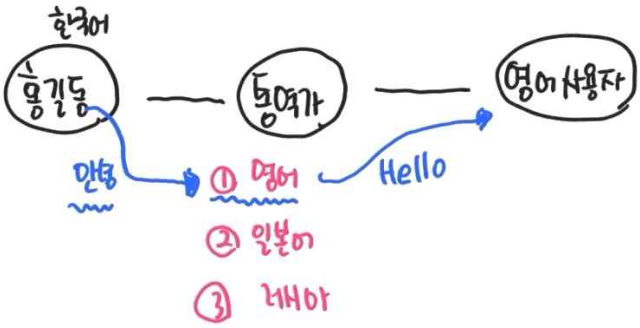
ⓑ 다운로드 받은 파일을 해당 폴더에 압축을 푼다.

ⓒ c:/javawork/download 폴더 생성 후 .zip 파일을 옮겨 둔다. → 나중에 또 쓸 수 있으니까...









3장. JDK 환경변수 설정

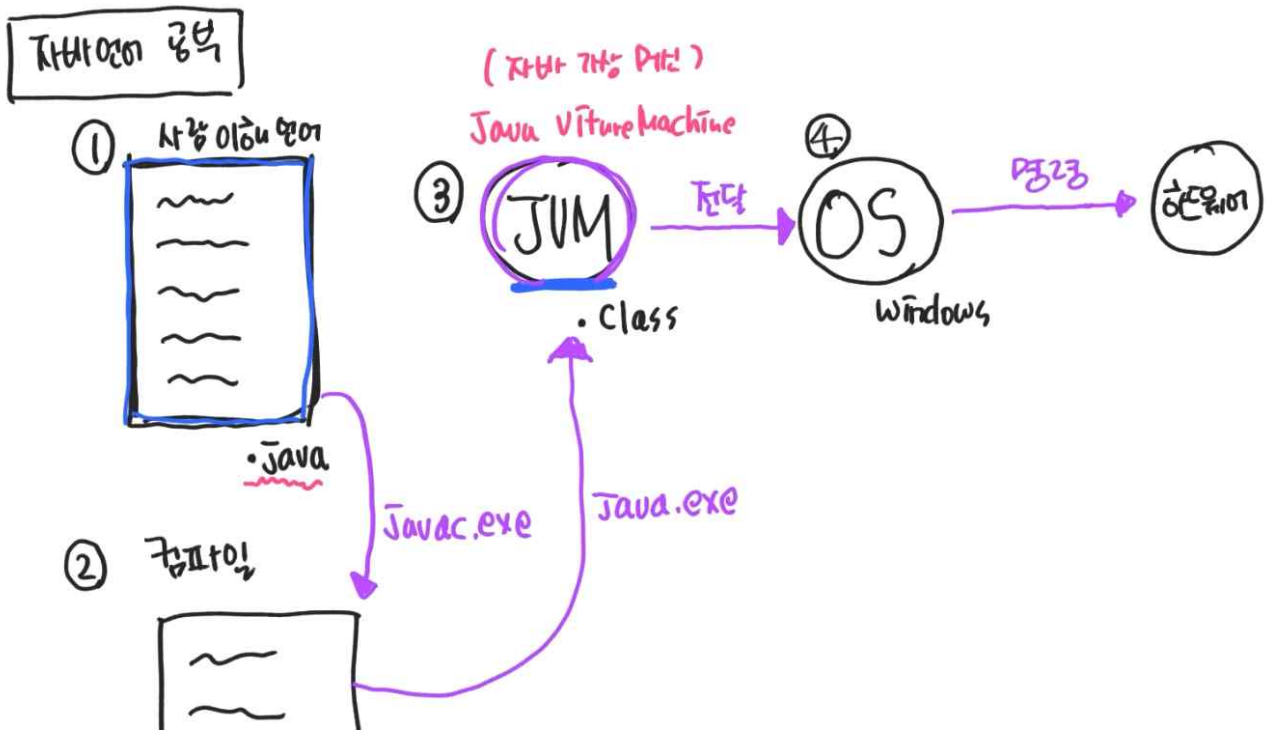
▶ 환경변수란?

→ 컴퓨터 내 어느 디렉토리든지 해당하는 프로그램을 실행할 수 있는 경로를 지정해 주는 것.

ⓐ C:₩javawork₩jdk-17.0.2₩bin 폴더 복사

ⓑ 내 PC → 마우스 우클릭 → 고급시스템설정 → 시스템 속성 → 환경변수 → 시스템 변수 → 새로만들기 → Path 더블클릭 → 새로만들기 → 붙여넣기 → cmd 종료 후 재시작

ⓒ java 를 설치하면 JVM(Java Virture Machine)이 만들어 진다.



ⓓ .java → javac.exe가 .class 파일을 생성 → Java.exe로 실행시키면 JVM이 받는다. → JVM은 윈도우 운영체제가 이해할 수 있는 파일형식으로 변환해서 전달함. → OS가 받아서 하드웨어에게 명령하여 결과를 보여준다.

1. 프로그램 개발은 어떠한 운영체제에서 개발해도 상관없다
2. 컴파일은 반드시 해당 운영체제에 맞는 곳에서 해야 한다.
   → Mac에 설치된 JVM으로 컴파일하면 Mac에서 돌아가는 프로그램이 된다.
   → 리눅스에 설치된 JVM으로 컴파일하면 리눅스에서 돌아가는 프로그램이 만들어 진다.

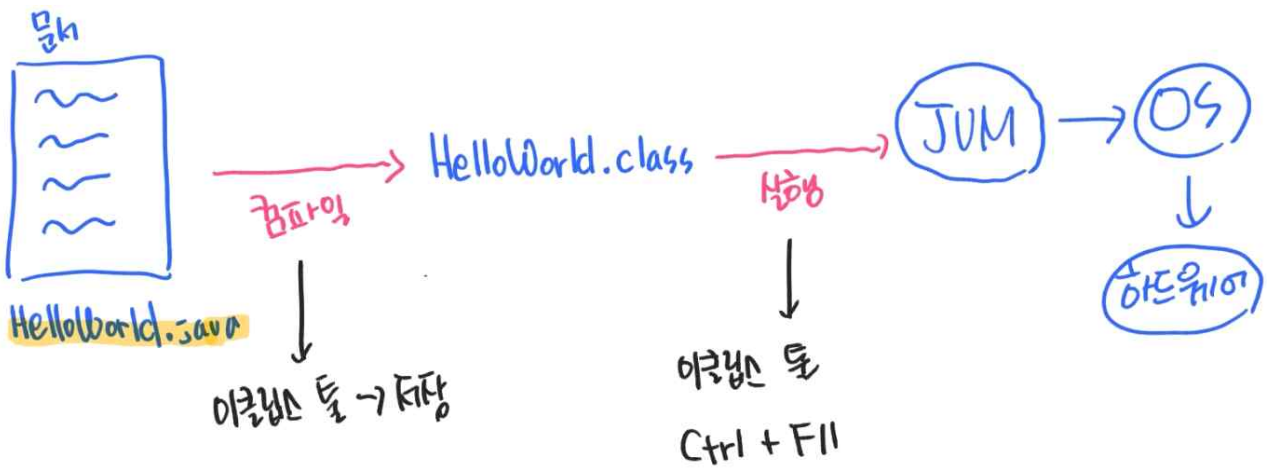
   즉, 어떠한 운영체제와 관계없이 해당 운영체제에 맞는 JVM이 설치되었다면, 어느 운영체제에서도 돌아가는 프로그램을 생성할 수 있다.

4장. 이클립스 또는 IntelliJ 설치

▷ 해당 사이트에 접속 후 download 폴더에 내려받고 설치.
→ Eclipse IED for Java Developers 설치 또는 IntelliJ Community 설치
▷ c:₩javawork₩java-project 폴더를 기본 폴더로 지정
▷ 프로젝트 테스트
ⓐ 이름 : test
ⓑ 폰트 : Verdana -- 확인해 보고 괜찮은 폰트면 자주 사용하자!

5장. IDE의 구성과 실행원리

▷ 해당 프로젝트의 폴더에 가보면 javawork → java-project → test → bin → src → HelloWorld.class
▷ HelloWorld.class 파일을 삭제해 보자

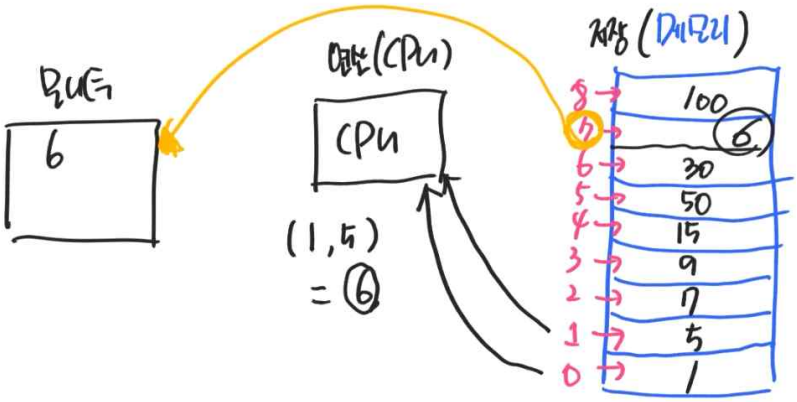


▶ 직접 위 단계를 실행해 보자
① java.class 파일 복사
② /javawork/java-test 폴더 생성 후 붙여넣기
③ 마우스 우클릭 편집 → package 명을 지우자(얘는 지금 파일이 어느어느 폴더에 있어... 라는 의미) → 저장 → cmd실행 → cd /javawork/java-test → javac HelloWorld.java → HelloWorld.class 생성 → 실행할 때는 java HelloWorld 만 치면 실행 됨... 파일의 확장자는 실행시에 입력 안해도 된다.
④ 위의 단계를 간단히 진행해 주는 것이 IDE이다.

※ 통합 개발 환경(Integrated Development Environment, IDE)
→ 공통된 개발자 툴을 하나의 그래픽 사용자 인터페이스(Graphical User Interface, GUI)로 결합하는 애플리케이션을 구축하기 위한 소프트웨어이다.

6장. Static, Heap, Stack이란

▷ 메모리 영역에 대한 개념



→ 연산을 위해서 메모리에 저장되어 있는 번지에 접근해서 데이터를 CPU로 갖고 온 다음 결과를 메모리 특정 번지에 저장하는 형태임.

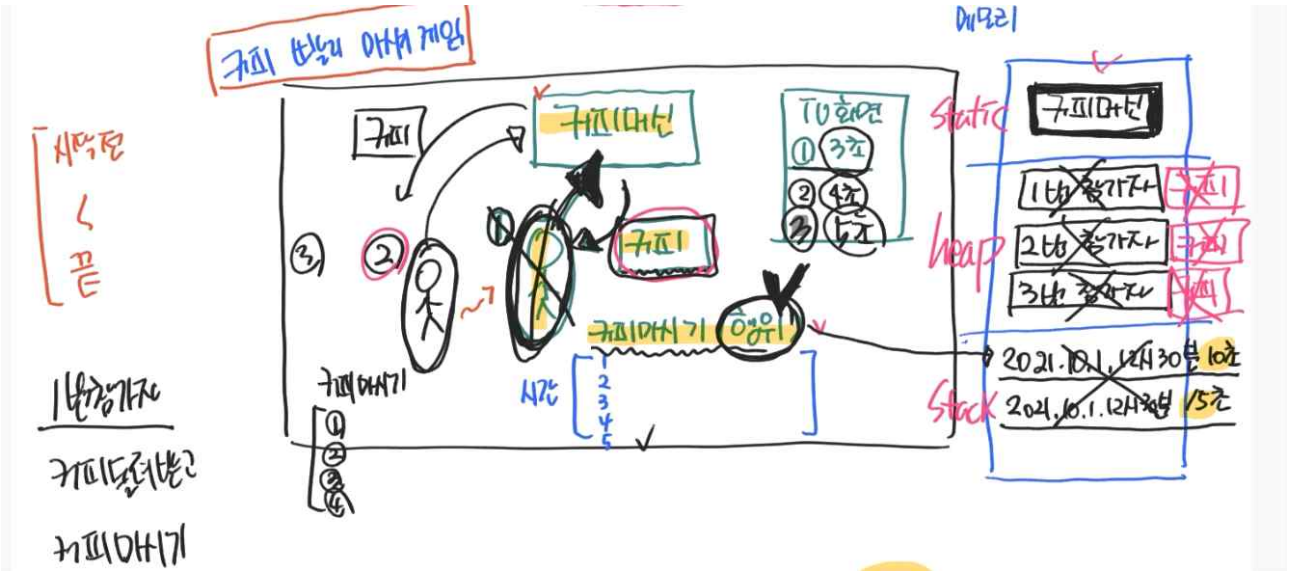
▷ 프로그래밍 언어는 메모리를 Static, heap, Stack 공간을 구분하여 사용함.

→ Static : 프로그램 시작 전부터 끝까지 메모리에 존재할 필요가 있는 것을 올려 놓음.(메모리 부하가 크다)

→ heap : 프로그램 운영 중에 동적으로 필요했다가 사라지는 데이터가 놓임.

→ Stack : 행위의 정보들이 스택에 저장된다. : 연산에 활용된다.

ex) 커피빨리 마시기 대회에서 커피머신은 프로그램이 시작된 후부터 종료될 때까지 메모리에 떠 있어야 하므로 Static 영역에 할당. 참가자와 커피는 동적으로 생성되었다가 사라져야 하므로 이들은 heap에 떠 있다. 스택은 커피마시는 행위의 시간을 계산하기 위해, 커피마시는 시작시간을 스택에 저장해 놓았다가 커피를 마시는 종료시간까지의 차이를 계산하기 위해 두 시간을 스택에 잠시 기억해 놨다가 연산에 활용함.



7장. 자료형이란

▷ 8가지 자료형 중 기본 자료형

① boolean : 1bit → 0 or 1

② int : 32bit → 약 -21억 ~ +21억

③ double : 64bit → -900경 ~ +900경

④ char : 16bit → 2^{16} → 65536

※ package : java 파일이 모여 있는 폴더

▶ package : ch01

```
package ch01;

public class VarEx01{
   public static void main(String[] args){
      boolean b1 = true;
      boolean b2 = false;
      int n1 = 1000;
      double d1 = 1000.1;
      char c1 = '가';
   }
}
```

8장. 자료형 메모리구조와 변수

▷ 메모리 구조

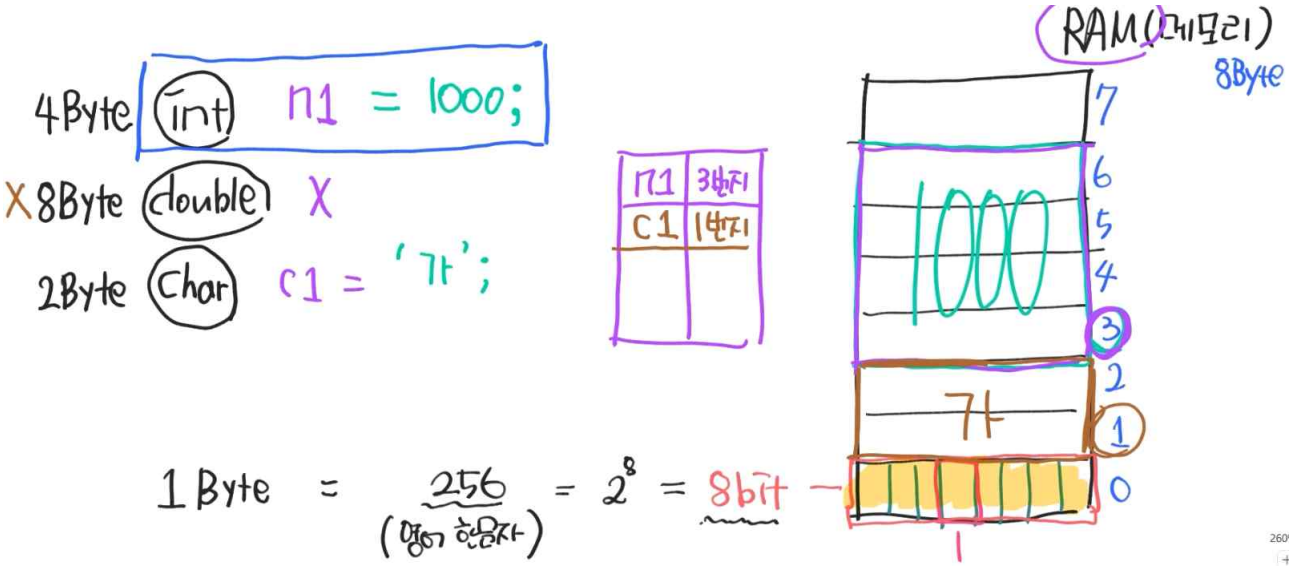
① RAM(메모리)은 0번지부터 시작해서 주소체계로 구성되어 있음.

② 각각의 주소는 8bit(1Byte)로 구성되어 있음.

③ 각각의 주소 1Byte는 2^8으로 총 256개의 문자를 표현할 수 있다. → ASCII 코드로 표현

④ 변수선언을 통해 메모리에 적절한 영역을 할당 함.

※ 변수란 변할 수 있는 데이터를 저장할 수 있는 공간




```
package ch01;

public class VarEx01{
   public static void main(String[] args){
      boolean b1 = true;  // 나 메모리에 1Byte 공간이 필요해
      boolean b2 = false;
      int n1 = 1000;  // 나 메모리에 4Byte 공간이 필요해
      double d1 = 1000.1;   // 나 메모리에 8Byte 공간이 필요해
      char c1 = '가';  // 나 메모리에 2Byte 공간이 필요해
      System.out.println(c1);
      System.out.println(n1);

      n1 = 50; //값을 변경
      System.out.println(n1);
   }
}
```

9장. 자바코드 실행원리

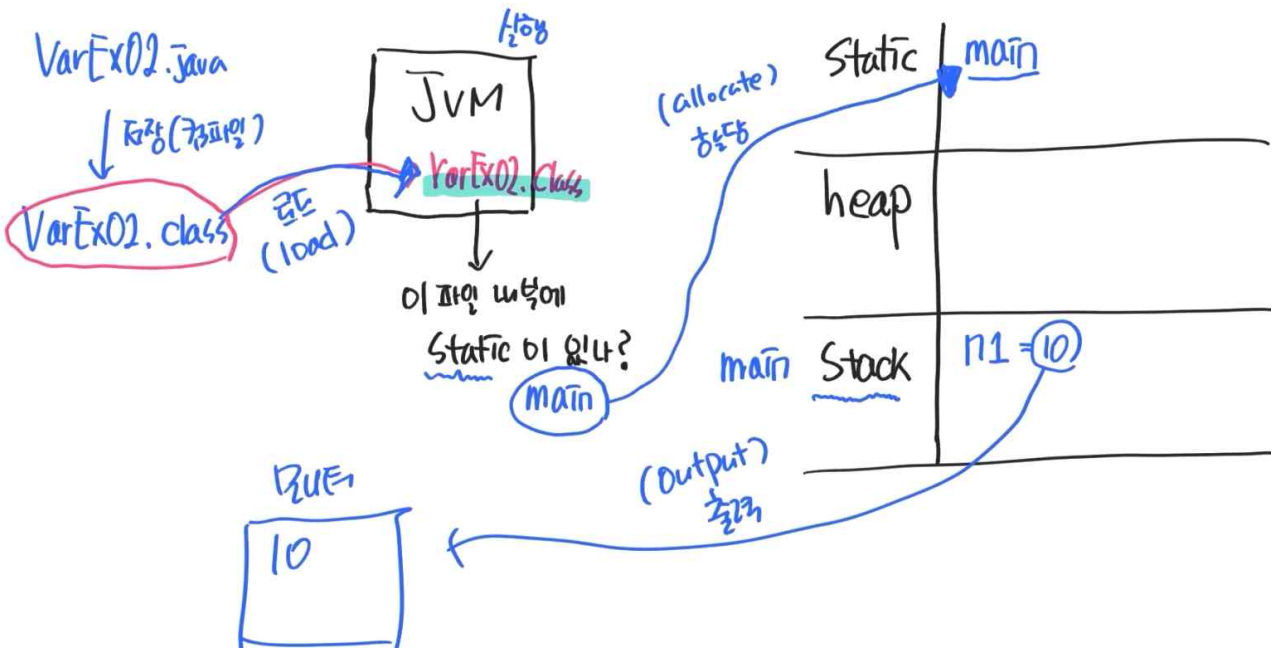
▷ 우리가 작성한 실행 가능한 코드는 main이라는 함수에 들어가 있음.

```
package ch01;

//int n3 = 30;   //Syntax Error

public class VarEx02{
   int n2 = 20;      → static int n2 = 20; //VarEx02라는 static 공간의 n2로 할당
   public static void main(String[] args){
      int n1 = 10; //해당 라인이 실행될 때 10이라는 값이 n1이라는 메모리에 할당 된다.
      System.out.println(n1);
      //System.out.println(n2);   //non-static field error
   }
}
```

① 자바의 모든 코드는 클래스 내부에 존재해야 한다.
② 자바는 실행전에 static 이라는 키워드를 찾아서 걔네들을 static 메모리 영역에 올려놓는다.
→ static 을 지우고 컴파일을 하면 static영역에 main이 없기 때문에 실행이 안된다는 메시지가 보임.
→ int n2 = 20; 은 실행을 안하기 때문에 메모리에 올려지지 않는다.
③ 자바를 실행하면 main이라는 친구의 내부를 실행한다.
④ main 내부 { } 가 끝나면 종료된다.(자바프로그램)



▶ 질문 : 그럼 int n2 = 20; 가 실행되려면 어떻게 해야 할까?

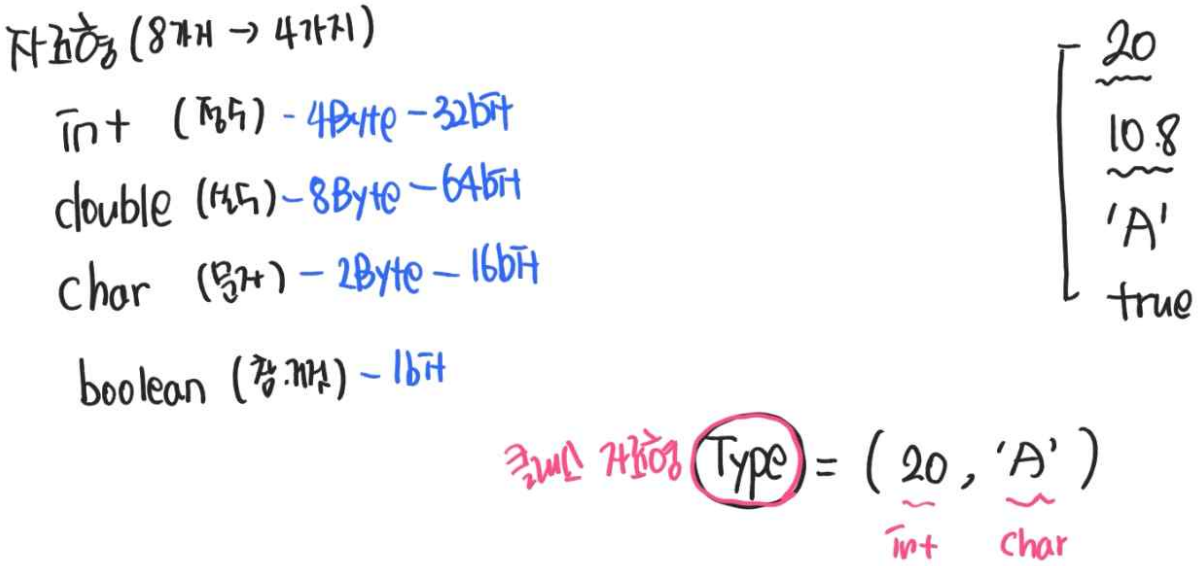
답> 두가지 방법이 존재.

ⓐ int n2 = 20; 를 스택공간에 할당하기 위해 main 함수 안으로 이동

ⓑ 프로그램이 시작될 때 static 공간에 할당하기

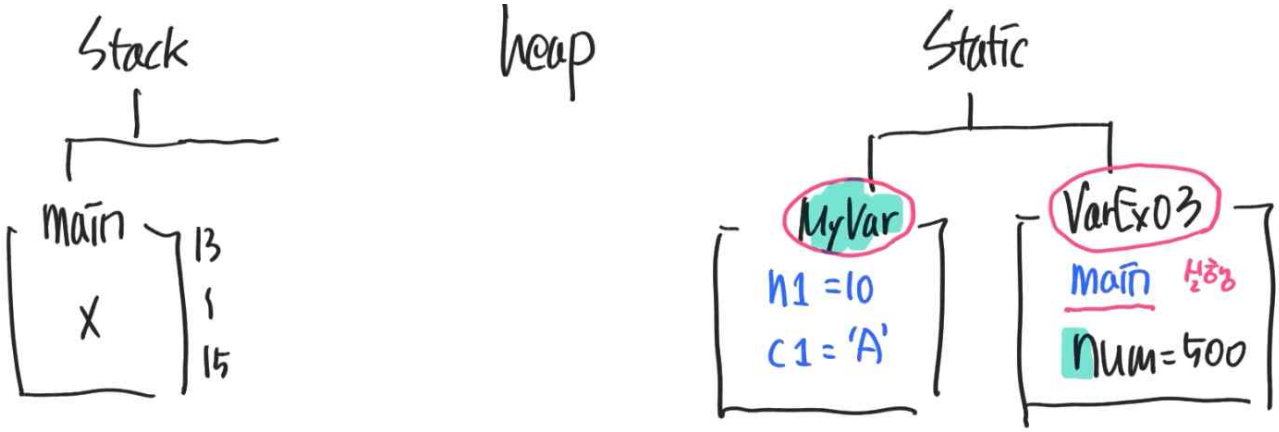
```
static int n2 = 20;
```

10장. 클래스 자료형(Beans)




```
package ch01;
// MyVar은 클래스 자료형 = 개발자가 만드는 커스텀 자료형
// 여러가지 데이터를 가지고 있는 클래스를 Beans라고 부름
class MyVar{
  //int n1 = 10;
  static int n1 = 10;
  //char c1 = 'A';
  static char c1 = 'A';
}

public class VarEx03{
  public static void main(String[] args){
    static int num = 500;
    //System.out.println(n1);  → 오류 : MyVar 클래스가 메모리에 안올라왔기 때문
                               → 그래서 MyVar 클래스 내부 변수를 static 으로 선언해야 함.
    System.out.println(MyVar.n1);  // . 은 연결연산자
    System.out.println(MyVar.c1);  // . 은 연결연산자
    System.out.println(VarEx03.num);  // VarEx03 생략 가능하나 써주는 것이 정확한 표현
  }
}
```


Stack
Heap
Static
main
13
X
1
15
MyVar
n1 =10
c1 = 'A'
VarEx03
main
num=500

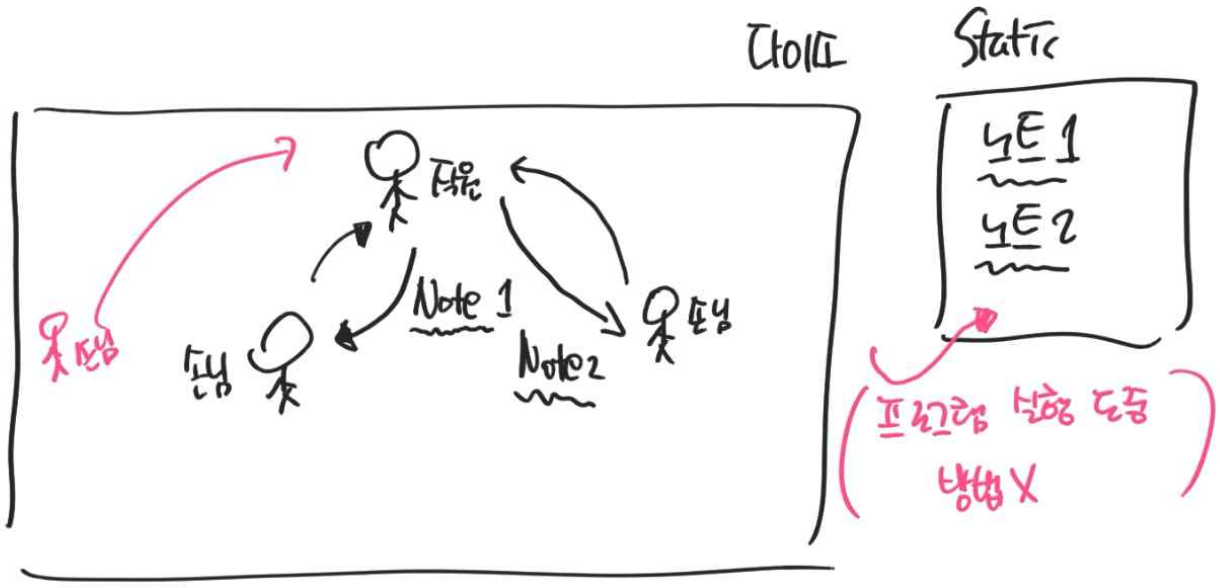
11장. 클래스 자료형 - heap 저장

▷ beans 클래스를 static 으로 저장할 때의 문제점

① 프로그램 시작전부터 종료시까지 메모리에 존재하게 된다.

② 많은 데이터를 저장하고 싶으면 클래스 자료형을 여러개 만들어야 한다.

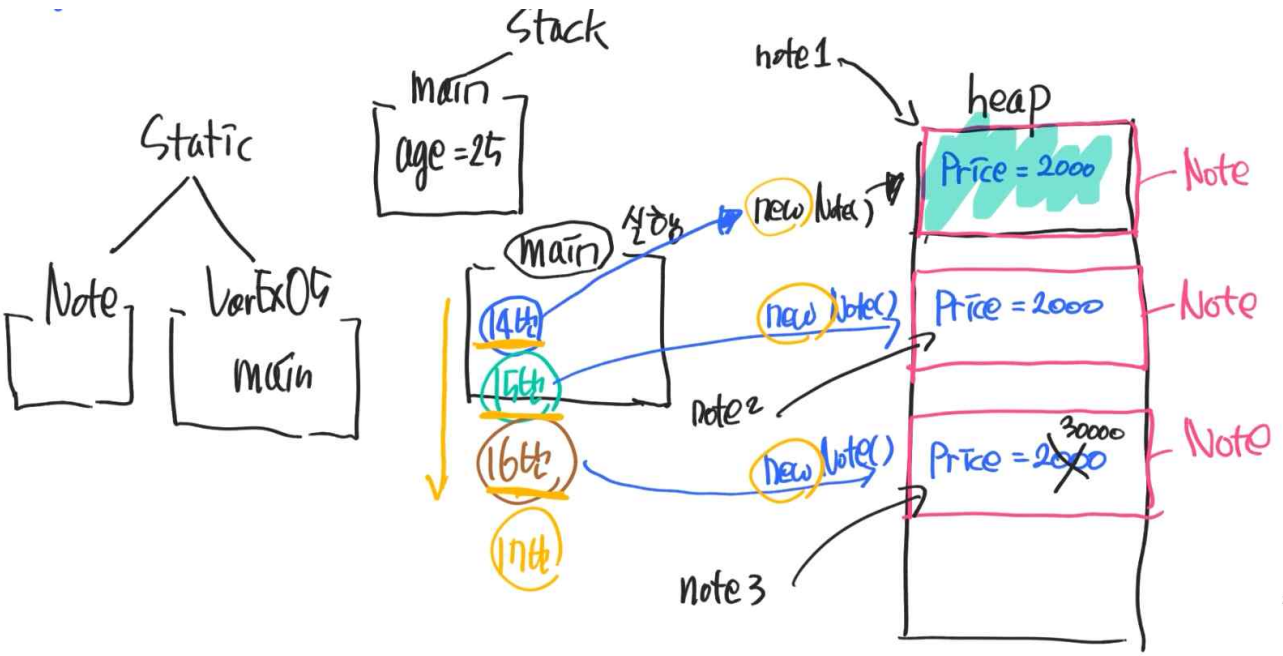
```
package ch01;
// MyVar은 클래스 자료형 = 개발자가 만드는 커스텀 자료형
// 여러가지 데이터를 가지고 있는 클래스를 Beans라고 부름
class Note{
   static int num = 1;
   static int time = 1015;
   static int price = 3000;
}
// 노트 100개가 필요? - 클래스 자료형 100개를 만들어야 한다. → 문제임.
public class VarEx04{
   public static void main(String[] args){
      System.out.println(Note.num);
      System.out.println(Note.time);
      System.out.println(Note.price);
   }
}
```



▷ 서비스 시작 시 노트가 2개 static에 생성되기 때문에 3번째부터는 서비스가 불가능하다. 왜? static은 실행 중간에 메모리에 추가할 수 없다. → static은 정적인 형태의 메모리이다.

▷ 그래서 동적인 처리가 필요하다. → 동적 저장 공간인 heap에 할당한다.

```
package ch01;
class Note{
   int price = 2000;
}
// 현재 메모리에는 VarEx05는 떠있고, Note는 메모리 할당을 못 받은 상태
public class VarEx05{
   public static void main(String[] args){
      new Note(); // heap공간에 Note 클래스가 갖고 있는 모든 변수를 할당해!!(단, static 빼고)
      new Note();
      new Note(); // new 라는 연산자는 동적으로 메모리에 할당할 수 있다.
      // 위 세가지 중 첫번째 자료를 찾을 수 있는 방법이 없다. 그래서 각각 타입과 이름을 부여
      Note note1 = new Note();
      Note note2 = new Note();
      Note note3 = new Note();
      int age = 25;  // main의 stack 공간에 생성
      System.out.println(age);
      note3.price = 30000;
      System.out.println(note1.price);
      System.out.println(note2.price);
      System.out.println(note3.price);
   }
}
```



▶ Chapter 1 연습문제

1) 붕어빵을 표현하기 위한 커스텀자료형(heap)을 만드시오.

- 클래스명 : 붕어빵
- 필드

붕어빵의 가격 : 1000
맛 : 달콤함
색깔 : 노랑

```
package ch01;
class 붕어빵{
   int price = 1000;
   String taste = "달콤함";
   String color = "노란색";
}
```

2) 메모장에 코드를 작성하시오.

- 파일명 : HelloWorld.java
- 폴더 : c:₩javawork₩ch1₩HelloWorld.java
- Hello World 를 출력하는 프로그램 만들기
- CMD에서 컴파일하고 실행하시오.

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

```
c:₩javawork₩ch1> javac HelloWorld.java
c:₩javawork₩ch1> java HelloWorld
Hello World!
c:₩javawork₩ch1>
```