비전공자 웹프로그램을 위한 자바문법책

발행일 : 2023.01.01

변경일 : 2023.01.01

저자 : 박수민

이메일 : foxman12@hanmail.net

발행처 : 휴먼교육

java -

목차

01. 자바 기초 시작하기

> 01. 최신 eclipse 설치하기

"이클립스"는 소프트웨어 개발을 위한 통합 개발 환경(IDE)중 하나로 Java 개발에 가장 널리 사용됩니다. java 개발을 편하게 하기위한 에디터(editor)이다.

에디터는 한글, 워드, 엑셀처럼 문서를 편집할 수있는 소프트 웨어이다. 이클립스는 자바문서를 편집하는 에디터이다.

다음 01장에서는 최신 eclipse를 설치하는 방법을 다룬다.

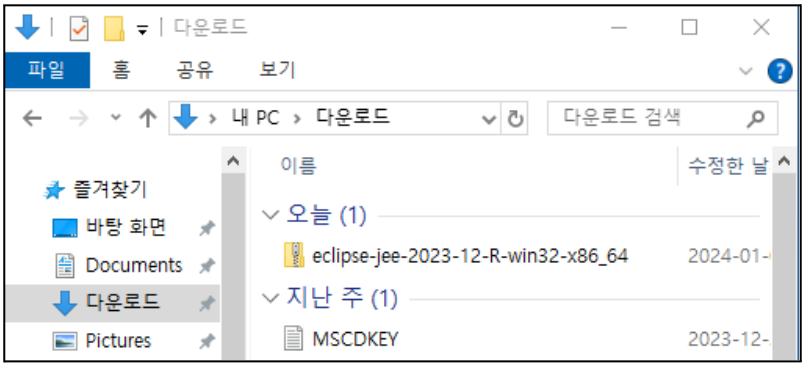
1. www.eclipse.org 방문하여 오른쪽 상단 다운로드(download)를 클릭해 보자.

2. 왼쪽 이미지에서 download packages 부분을 클릭해 보자.

3. Eclipse ide for enterprise java and web developers 툴을 다운로드 받을 예정이다. 윈도우즈를 사용할 예정이므로 다음 이미지 windows x86_64 문자열을 찾아 클릭해 보자.

4. 이동한 페이지에서 왼쪽 이미지 커서 부분 문자열을 클릭하고 조금 기다리면 해당 파일을 다운로드 할 수 있다.

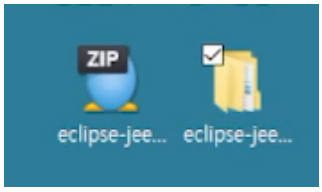
5. 다운로드가 완료될 때까지 기다리자.

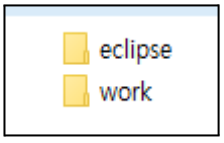


6. 내 컴퓨터 다운로드 폴더로 이동한다. 작업 창에서

와 같은 내 컴퓨터 아이콘을 누르거나 아무 폴더를 누른 다음 왼쪽 이미지 처럼 다운로드를 클릭하면 방금 다운로드 된 파일을 확인할 수 있다.

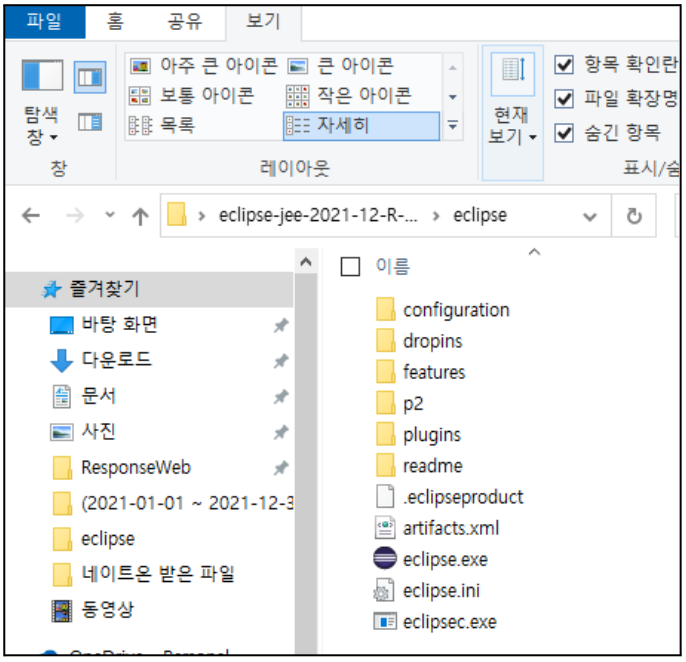
7. 해당 파일을 실행하는 방법은 다른 윈도우즈 프로그램 처럼 설치하는 과정이 필요 없이 다운로드한 압축된 파일을 풀어서 원하는 위치에 옮겨 놓으면 된다.





8. 압축 푼 폴더에 들어가면 eclipse폴더가 있는데 마우스 오른쪽 클릭 >> 새로 만들기 >> 폴더 를 선택해서 work폴더를 만들자.

f2를 누르면 폴더 이름을 변경할 수 있다. work폴더는 eclipse에서 본인이 생성한 데이터를 저장할 폴더이다. 이클립스 실행시 작업폴더를 요청 하면 이 work폴더를 선택 하면 된다.

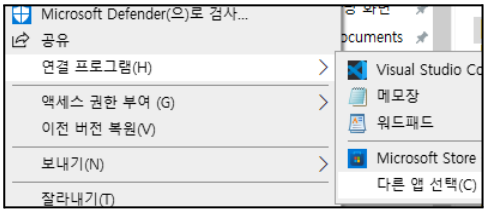


9. eclipse 폴더를 선택해서 들어가면 왼쪽 이미지와 같은 화면이 나온다.

10. 파일 확장자가 안보이면 보이도록 파일 탐색기 상단 '보기 탭' 을 클릭하고 오른쪽 메뉴에서 파일 확장명, 숨긴 항목 등을 선택한다. 왼쪽 이미지 상단에서 찾아 보자. 작업이 완료되면 파일의 확장자와 안보이던 파일들이 보이게 된다.

11. eclipse.ini파일을 폴더안에서 찾아 보자.

INI 파일(Initialization File)은 간단한 텍스트 기반의 설정 파일 형식입니다. eclipse.ini 파일은 이클립스 실행시 필요한 설정 정보를 가지고 있다. eclipse가 어떤 jdk를 사용하는지, 얼마만큼의 메모리를 사용할 것인지, 문자 인코딩은 어떤것을 사용할 것인지 등 eclipse 프로그램 동작에 필요한 설정 정보를 가지고 있다. 설정 정보는 어떤 시스템, 응용 프로그램, 또는 기기의 동작을 제어하고 조정하기 위한 데이터를 의미 한다.



12. eclipse.ini 파일 위에 마우스를 가져다 대고 오른쪽 클릭 >> 연결프로그램 >> 메모장 을 선택하면 해당파일을 편집할 수 있게 메모장 에디터가 화면에 나타난다. 메모장이 없으면 오른쪽 클릭 >> 연결프로그램 >> 다른 앱선택 항목을 선택해서 메모장을 찾아서 파일을 열 수 있다.

12. ini 파일 중간 부분에 아래 이미지와 같은 부분이 있는데 jdk는 openjdk 17버전을 가상운영체제(virtual machine)VM으로 사용하고 있다는 의미이다.

```
-vm
plugins/org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1
```

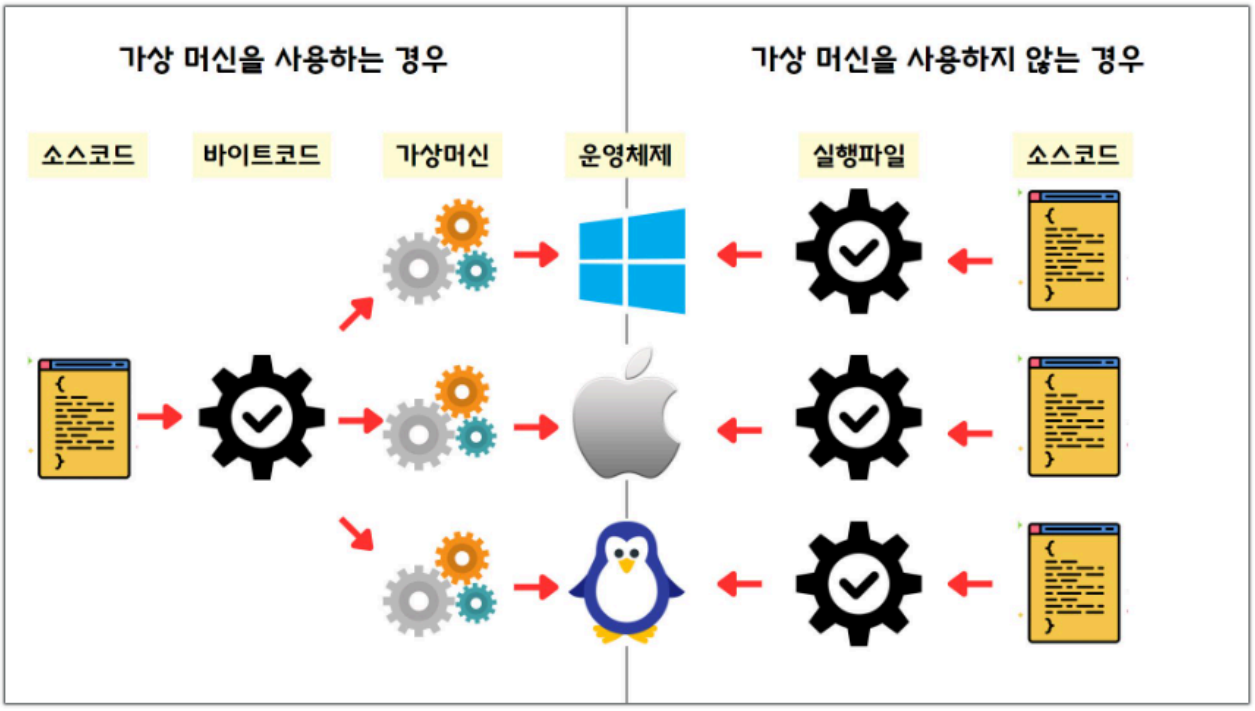
JDK (Java Development Kit):

자바 언어로 애플리케이션 프로그램을 개발하고 실행하기 위한 도구 및 라이브러리를 의미

OpenJDK는 JDK를 오픈 소스로 개발한 자바 개발 키트로 무료 사용을 위해서 많이 사용된다.

java 가상 머신(JVM, Java Virtual Machine)은 자바 프로그램이 작성된 운영체제에서

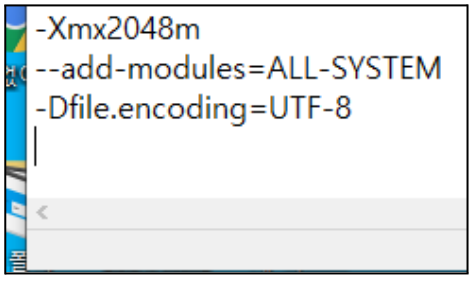
동작할 수 있도록 해주는 **가상화된 소프트웨어 환경**입니다.



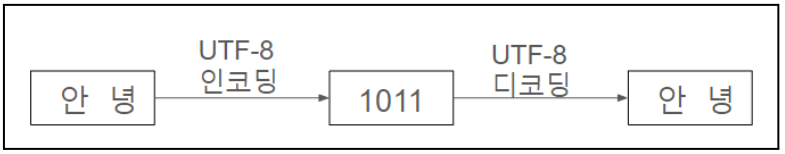
자바의 실행 방법은 코드를 작성하고 기계가 알아보기 쉬운 바이트코드로 컴파일을 통해서 변경한다음 가상머신을 통해서 실행하면 모든 운영체제에서 동일하게 확인할 수 있다.

과거에는 운영체제마다 프로그램을 **다시 만들어야** 했다. 윈도우용, 리눅스용, 맥용으로 각각 다른 코드를 작성해야 했지만 자바는 **한 번만 작성하고** 운영체제에 맞는 JVM만 설치하면 됩니다. 윈도우에서는 윈도우용 JVM, 리눅스에서는 리눅스용 JVM을 설치한 후, **동일한 자바 코드를 실행**할 수 있습니다. 자바 버철 머신을 통해 자바는 운영체제에 상관없이 어디서든 **재작성 없이 실행**되는 장점을 가집니다.

원래 vm은 따로 설치해서 사용해야 하는데 이클립스를 설치하면 자동으로 설치된다. 여기서는 따로 설치 없이 자동 설치된 vm를 사용 할 예정이다.



13. eclipse.ini 파일 맨아래 부분에 왼쪽 이미지처럼 -Dfile.encoding=UTF-8 부분을 추가하고 해당 ini파일을 저장한 다음 오른쪽 상단 x 버튼을 눌러 문서를 닫자. 종종 문서를 열고 eclipse를 실행하면 제대로 적용되지 않을 때가 있으니 열린 ini파일을 닫고 eclipse를 새로 시작 해야 정상 적용된다. 인코딩을 UTF-8로 설정하는 작업을 한 것이다.



인코딩은 정보를 특정 규칙에 따라 다른 형태로 변환하는 과정이며, 디코딩은 그 변환된 정보를 다시 처음의 형태로 되돌리는 과정입니다. 컴퓨터는 0과1로만 데이터를 관리할 수 있기 때문에 데이터를 저장하고 보여주려면 인코딩과 디코딩 과정을 거친다. 이때 인코딩 할때 사용한 기준과 디코딩 할때 사용하는 기준이 동일 하지 않으면 변환중 문제가 발생 하는데 변환 기준을 항상 UTF-8로 하라는 의미이다.

안녕을 UTF-8에서 정한대로 0과 1로 바꾸면 1011이 된다고 할때 1011를 UTF-8이 아닌 다른 기준으로 디코딩을 하면 안녕이라는 문자열을 얻지 못한다.

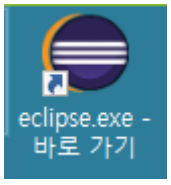
"UTF-8과 UTF-888은 서로 다른 규칙을 사용하는 문자 인코딩 방식이라고 가정할 때, 같은 '1011'이라는 코드가 UTF-8에서는 '안녕', UTF-888에서는 '잘가'로 해석될 수 있습니다. **기준을 명확히 하지 않으면** ‘안녕’이라고 저장했는데, 나중에 읽어올 때 ‘잘가’로 출력될 수 있는 문제가 발생합니다.

예를 들어:

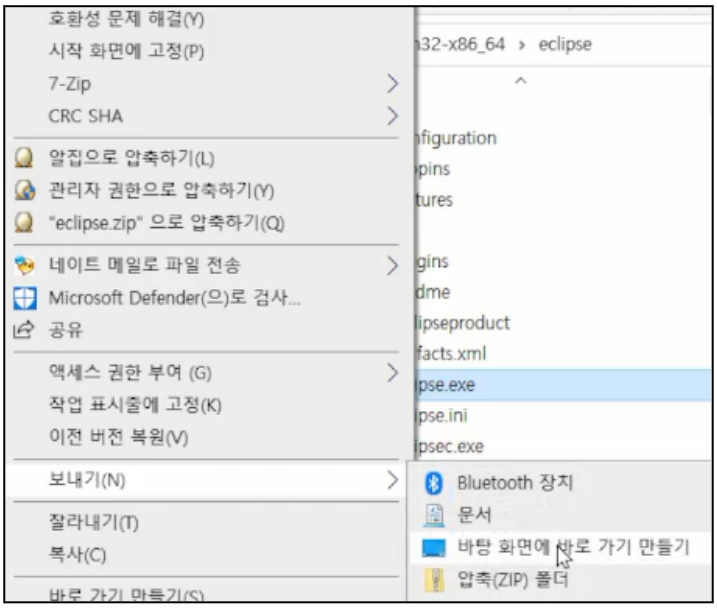
1. **UTF-8로 인코딩**:
    - '안녕' → 1011로 저장.
2. **UTF-888로 디코딩**:
    - 1011 → '잘가'로 출력.

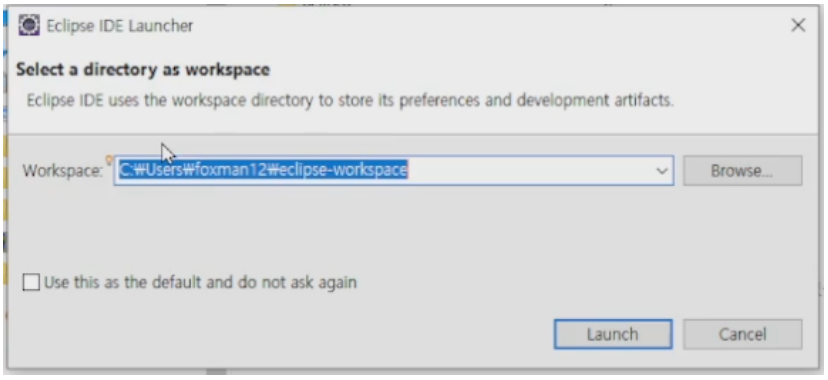
이처럼 **인코딩과 디코딩 방식이 다르면** 저장된 값이 **의도와 다르게 해석**될 수 있으므로, **일관된 인코딩 방식을 사용**하는 것이 중요합니다. 종종 한글이 깨지는 경우가 있는데 인코딩 할때 사용한 기준과 디코딩 할때 사용하는 기준이 달라서 이다.

우리는 한글 깨지는 문제를 없애기 위해서 기준을 UTF-8로 설정해서 사용 할 예정이다.

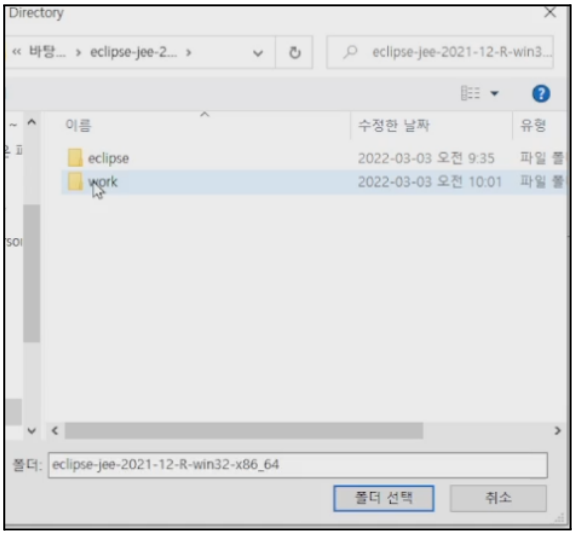


14. 바탕화면에 왼쪽 이미지 처럼 바로가기 파일을 만들기 위해서 eclipse.exe파일에서 마우스 오른쪽 클릭 >> 보내기 >> 바탕화면에 바로가기 만들기를 클릭하면 된다. 다음 이미지를 참고 하자.



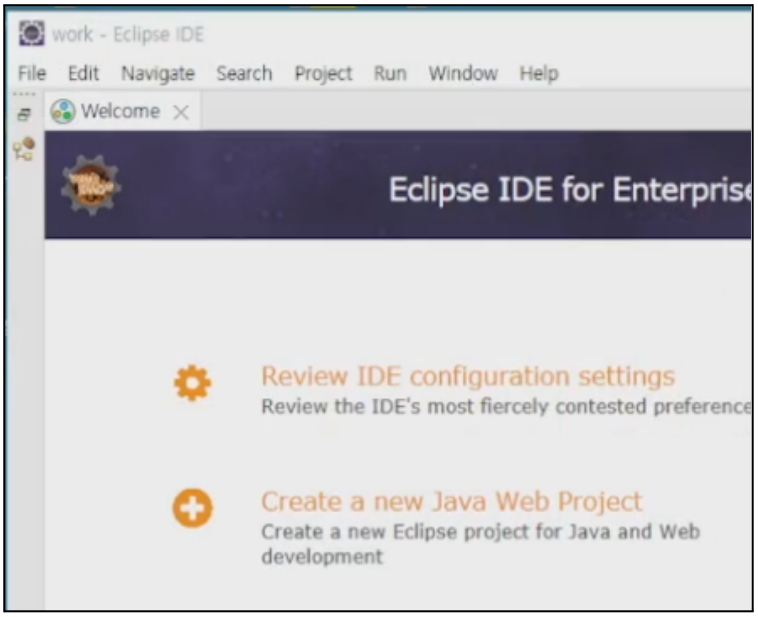


16. eclipse가 실행되면 왼쪽 이미지와 같이 에디터에서 이후 작업할 코드 내용들을 저장할 파일 폴더를 설정 하라는 창이 뜬다. 변경하고 싶다면 browse.. 버튼을 클릭하여 원하는 폴더를 선택한다.



17. 이전에 만든 work 폴더를 찾아서 선택한 다음에 폴더 선택 버튼을 클릭하면 설정이 종료된다. 이후에 저장된 작업파일을 찾고 싶다면 해당 폴더에서 찾을 수 있다.

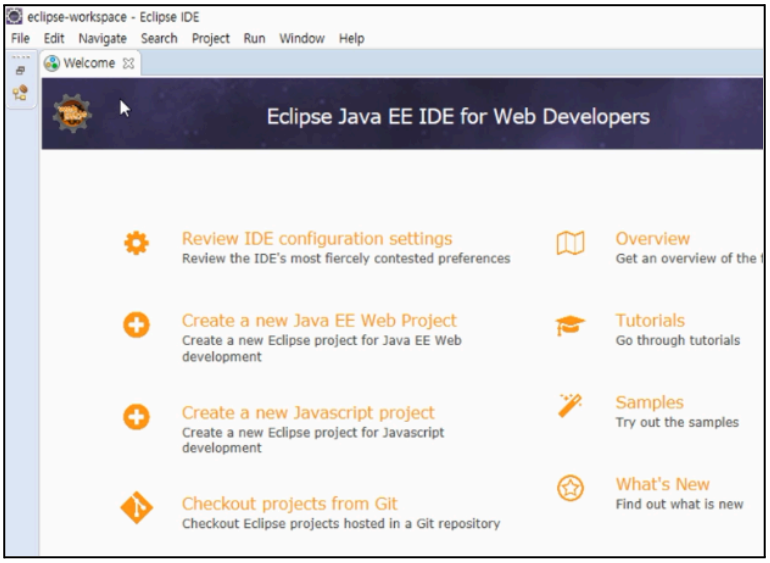
18. 아이콘을 잘못 2번 클릭해서 이클립스가 2개가 실행되는 경우가 있는데 canel버튼을 눌러서 하나를 취소하자.



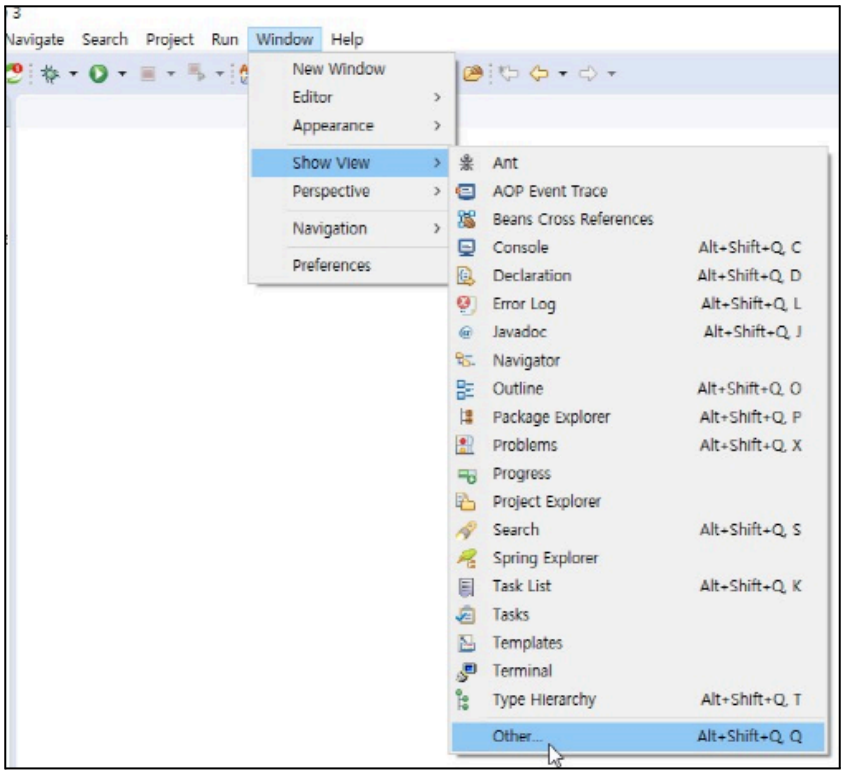
19. 왼쪽 이미지는 정상 실행되었을 때의 화면이다. 종종 버젼 문제로 잘못된 화면이 나오는데 무시하고 넘어가자.

20. 설치가 완료 되었다면 다음 장으로 이동하여 프로그램을 이어나가 보자.

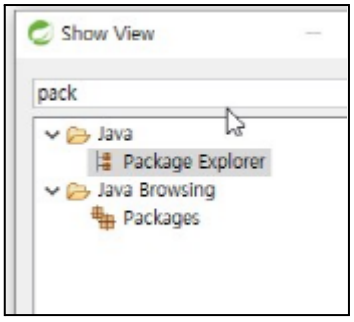
> 02. eclipse 첫 실행 hello world



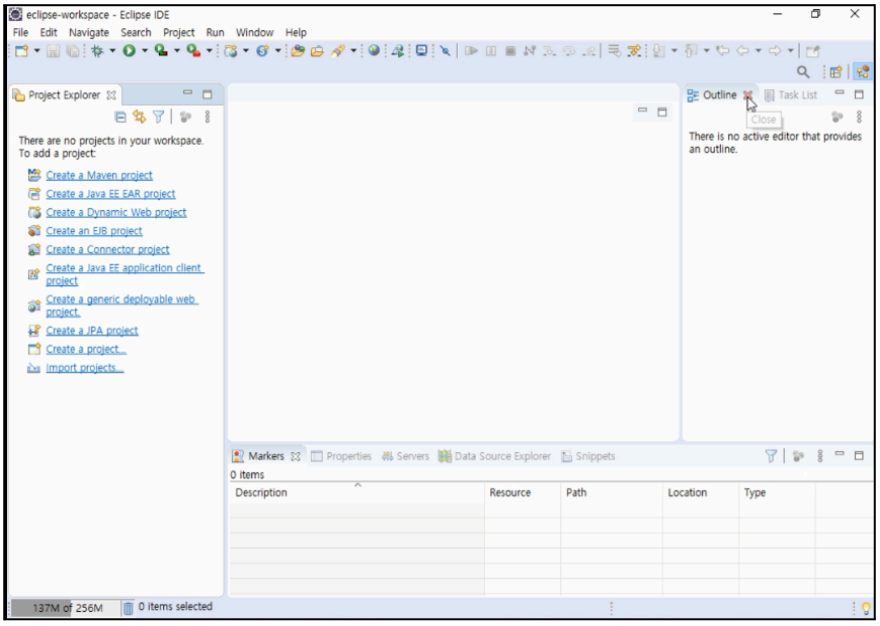
이클립스가 실행되고 나면 왼쪽과 같은 화면이 뜨는데 설치된 버전 문제가 있어서 에러 화면이 뜰 경우도 있으니 프로그램이 실행 되었 다면 무시하고 그냥 사용하면 된다. 프로그램의 welcome 탭의 x 를 눌러 창을 닫을 수 있다. 필요 없는 창은 오른쪽 상단 x 눌러서 없앨 수 있다.



만약 작업하는 창이 닫혀서 다시 열고 싶다면 왼쪽 상단 window >> show view >> other… 를 클릭한 다음 아래 이미지 처럼 창이뜨면 찾고자 하는 창의 이름을 입력 하여 찾은 후 선택한 다음 아래 open버튼을 클릭하면 선택한 창이 화면에 나타난다.

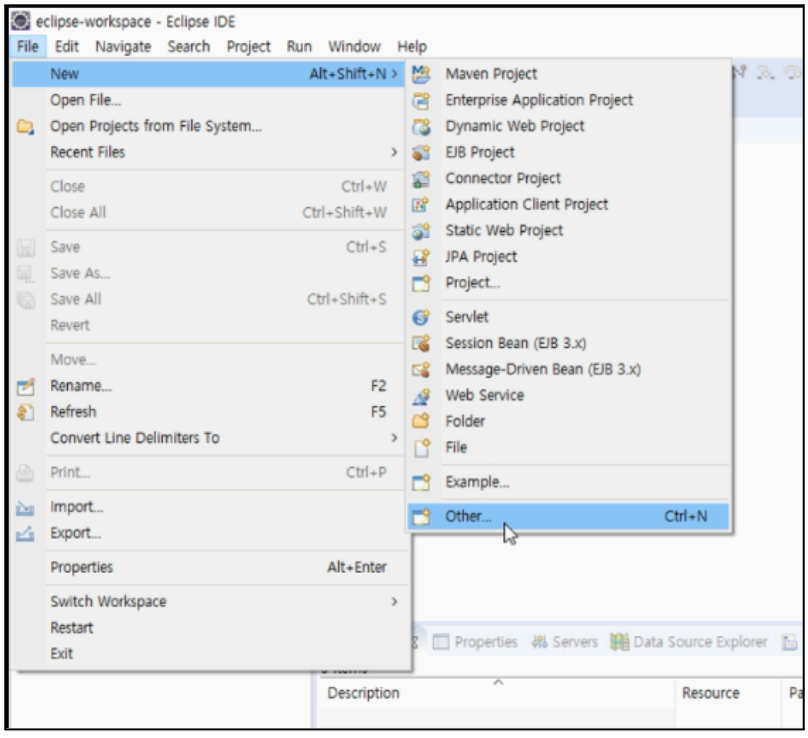


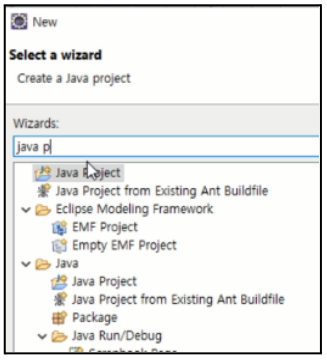
많이 쓰는 창은 package Explorer와 console 창이다. 패키지 익스플로러에서는개발자가 작성한 소스코드 파일을 클릭하면 파일의 내용을 확인할 수 있다. 콘솔 창에서는 소스 코드의실행 결과를 확인할 수 있다



프로그램 작업을 할 수 있는 자바 프로젝트를 만들어 보자.

아래 왼쪽 이미지 처럼 메뉴에서 file > new 에서 Java Project를 클릭하면 프로젝트를 생성할 수 있는데 목차에 자바 프로젝트가 존재하지 않으면 목차 중 other를 선택 하고 아래 오른쪽 이미지 처럼 검색창에 Java Project를 입력해서 찾을 수 있다.





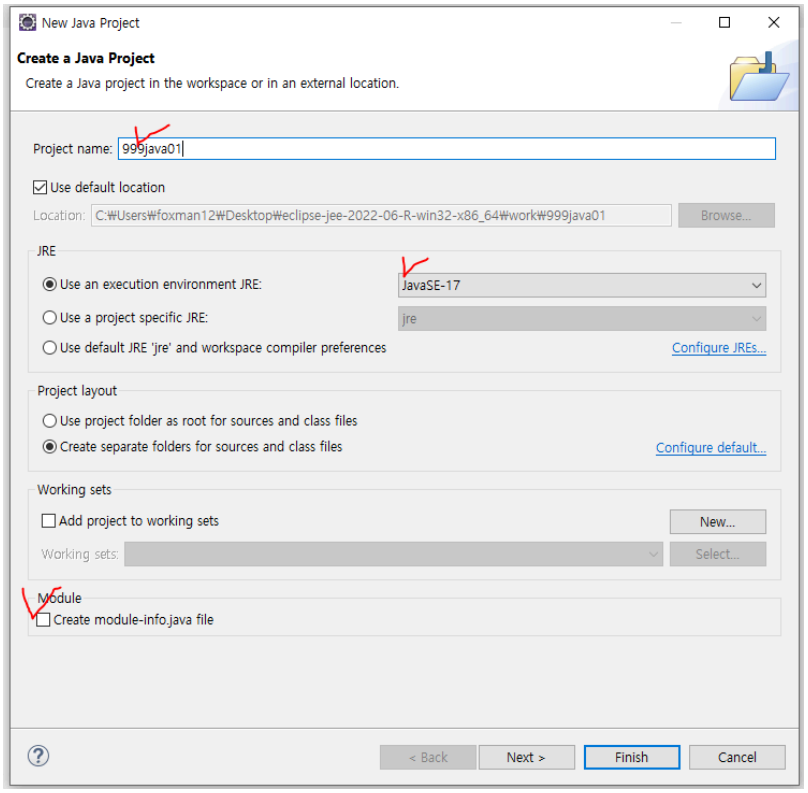
찾은 자바 프로젝트를 마우스로 클릭 한 다음 아래 작업을 이어 나가자.

프로젝트 하나 만들 때마다 워크스페이스 위치에 하나의 폴더가 만들어 진다. 프로젝트 생성 작업 후 이전에 만든 work폴더에 가서 생성 되었는지 확인해 보자.

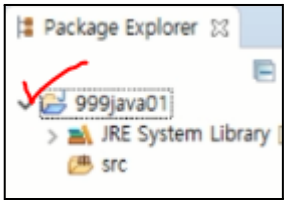
옆에 이미지가 아닌 다음 페이지 이미지의 체크된 부분에 프로젝트 이름을 999java01 이라고 넣고 사용 하고자 하는 jdk버전을 설정하고 맨아래 module부분을 체크 해제한 다음 finish를 누르면 왼쪽 이미지 같은 창이 뜬다. open perspective 버튼을 클릭하면 프로젝트가 생성 된다. 화면을 어떤 모양으로 구성할 것인지를 선택하는 부분인데 별로 중요하지 않으니 선택 하고 그냥 사용하면 된다. jdk는 기본 설정되어 있는 버전을 그냥 쓰면 되고,

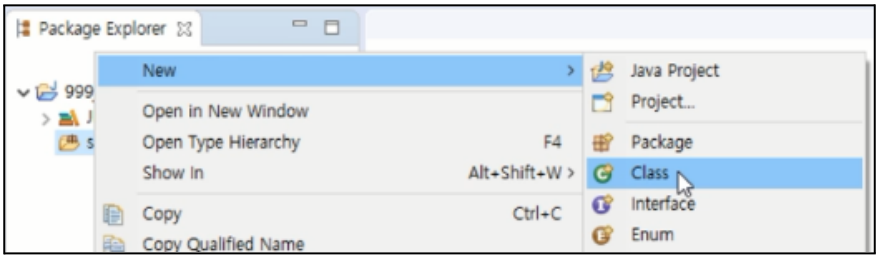
module체크 해제는 새로운 버전에 추가된 내용이다. 체크 해제하고 사용하면 옛날의 방식대로 사용할 수 있다.

아래 이미지에서 빨간색 체크한 부분을 잘 확인해 보면서 따라하자.



확인이 마무리 되고 finish 버튼을 눌러서 프로젝트가 만들어지면 아래 이미지 처럼 Package Explorer에 999java01 프로젝트가 만들어 진다. 폴더 처럼 빨간색으로 체크한 부분을 클릭하면 프로젝트 안에 내용을 확인 할 수 있고 다시 클릭하면 접힌다.





Package Explorer에 src폴더에서 마우스 오른쪽 클릭을 한 다음 new>>class 메뉴를 클릭한다. class가 없다면 아까 처럼 other 메뉴를 클릭후 class 입력하여 찾으면 아래 페이지 이미지와 같은 화면이 나온다.

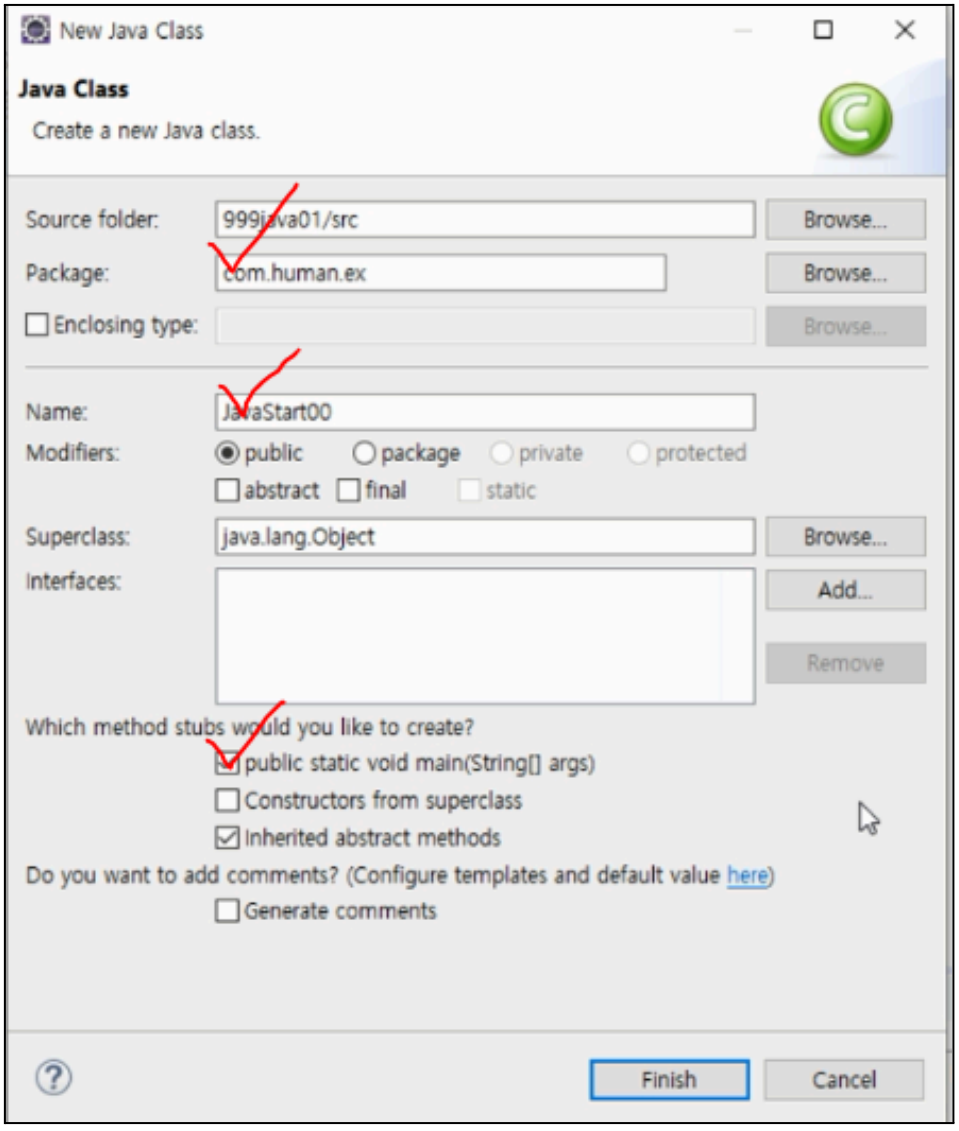
첫 번째 빨간색 화살표 안에는 package를 넣으면 된다. 패키지는 지금 작업할 파일이 저장될 폴더 위치라고 생각하면 된다. com.human.ex라고 입력 한다면 현재 작업할 파일을 com 폴더 아래 human 폴더 아래 ex 폴더에 만들라는 의미이다. 보통 식별자로 사용되어야 하기 때문에 인터넷 주소를 거꾸로 입력한 형태를 사용한다. 소문자로 본인이 생각한 이름을 입력하면 된다.

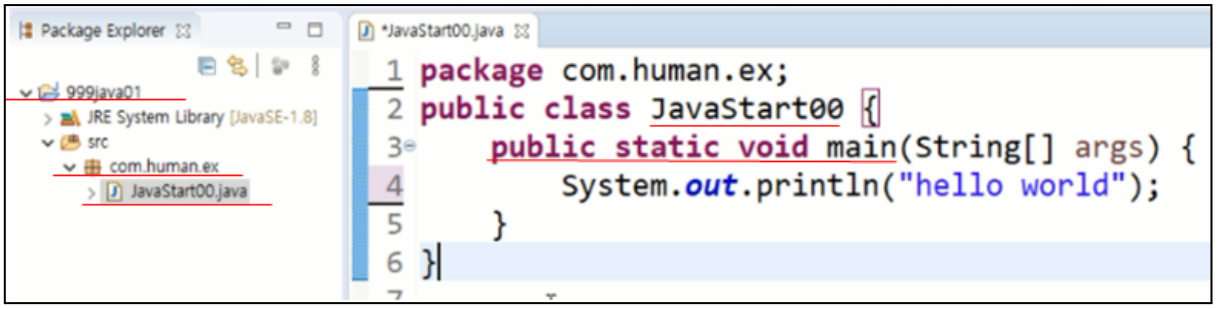
운영체제에서 폴더를 여러개 사용하는 이유는 파일이 많아지면 관리하기 편하게 하기 위해서 이다. 패키지도 마찬가지다 java코드 파일이 많아지면 관리하기 편하게 하기 위해서 패키지를 사용한다.

두번째 빨간색 화살표 안에는 현재 작업할 내용을 저장할 파일 이름을 적어 준다. 입력 작업이 끝나면 입력한 이름으로 이후 작업한 코드가 저장된다. 여기서는 JavaStart00.java

라는 파일 이름으로 저장된다. JavaStart00 과 같이 입력하고 나면 파일이 만들어 질때 이름이 JavaStart00.java로 만들어 지고 클래스 이름도 JavaStart00 으로 만들어 진다.

세 번째 빨간 화살표 앞에 체크박스를 클릭하면 곧 만들어질 JavaStart00.java 파일 안에 특정 코드가(메인(main) 메소드가) 자동으로 추가 된다. 클릭 하지 않으면 추가되지 않는다. main메소드는 프로그램을 시작할때 사용되는 메소드로 해당 파일을 직접 실행하려면 반드시 추가되어야 하는 메소드이다. 모든 입력이 끝나면 맨 아래의 피니쉬(finish) 버튼을 클릭한다.





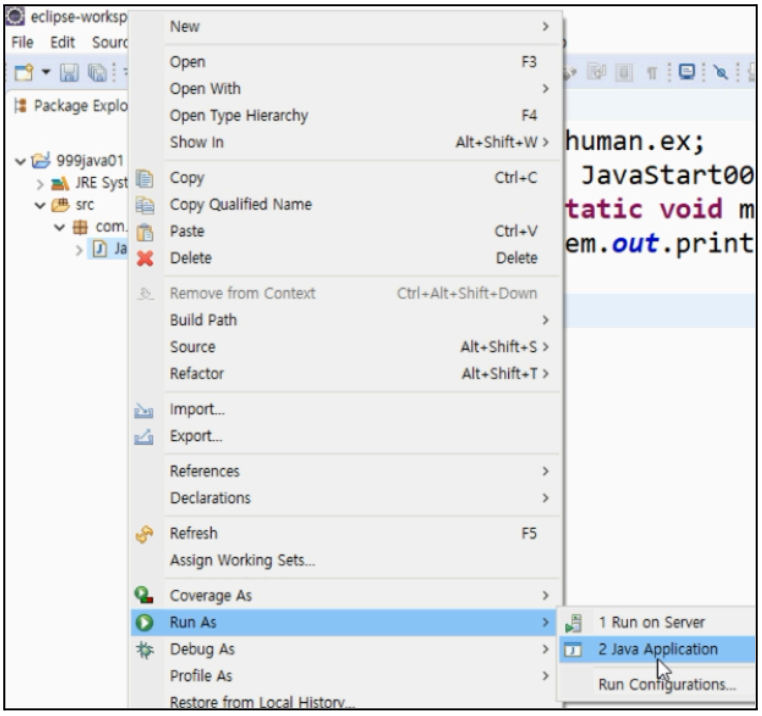
빨간줄 쳐져 있는 부분이 본인이 어디에 입력한 부분인지 확인해 보자.

확인이 끝났으면 // TODO Auto-generated method stub 다음 라인을 삭제하고 상위 이미지 처럼 4번 줄에 System.out.println("hello world”); 를 입력 하자.

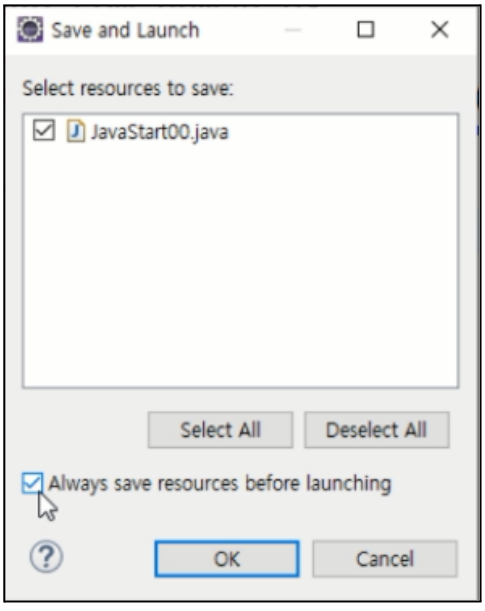
코드를 간단히 설명해 보면 1번 라인 package com.human.ex;은 현재 만든 파일이 존재하는 폴더 경로를 의미 한다.

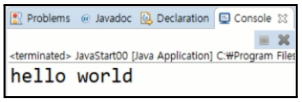
2번 라인은 클래스를 선언 하는 부분인데 모든 자바 코드는 반드시 클래스안에 기술하여야 한다.

3번 라인의 public static void main(String[] args) { 이 부분은 프로그램의 시작 부분을 의미 한다. 중괄호 안의 코드 부분이 순서대로 실행되다가 }를 만나면 프로그램이 종료된다. tab키를 사용하여 띄어쓰기도 맞춰서 작성하자.



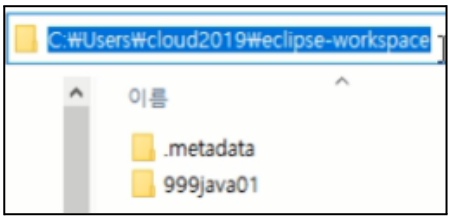
위 이미지 처럼 코드 입력이 끝나면  모양의 버튼을 눌러서 모든 파일을 저장한 다음 왼쪽 이미지 처럼 javaStart00.java 파일에서 마우스 오른쪽 클릭 Run As Java Application를 선택하여 프로그램을 실행해 보자. 실행 시키면 보통 자동 저장 되지만 저장 하지 않으면 오류 메시지에 문제가 생길수 있다. 이를 방지하기 위해서 코드 작업이 끝나면 저장 버튼을 누르는 습관을 가지자.



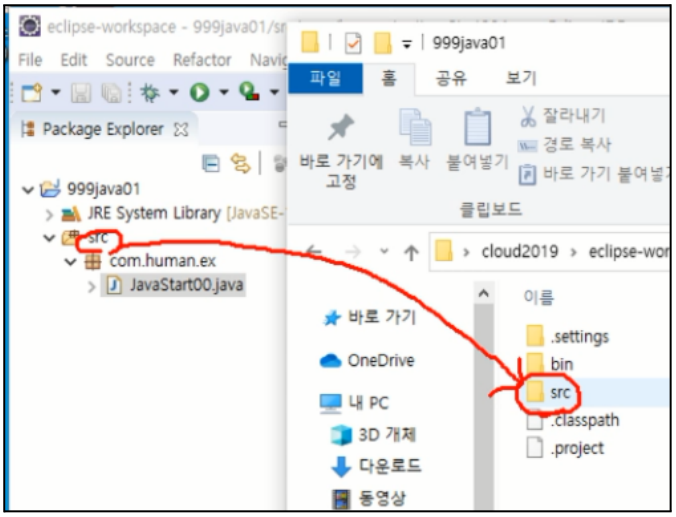


실행되면 console창에 왼쪽 처럼 hello world가 출력될 것이다. 이 창은 상단 부분 탭을 마우스로 드래그 해서 원하는 화면 위치로 이동시킬 수 있다.

만약 작성한 파일을 저장하지 않으면 왼쪽 이미지 처럼 메시지 박스가 생긴다. always save XX 라고 쓰여져 있는 앞 부분의 체크 박스를 클릭 하고 ok 버튼을 누르면 실행되는 것을 확인할 수 있다. 변경한 파일이 저장되어 있지 않은데 저장하고 실행할 것인지 물어보는 메시지 이다. 저장된 파일의 위치는 아래와 같다. 이클립스 실행시 작업 폴더를 work폴더로 변경 했다면 work폴더에서 확인하면 된다.

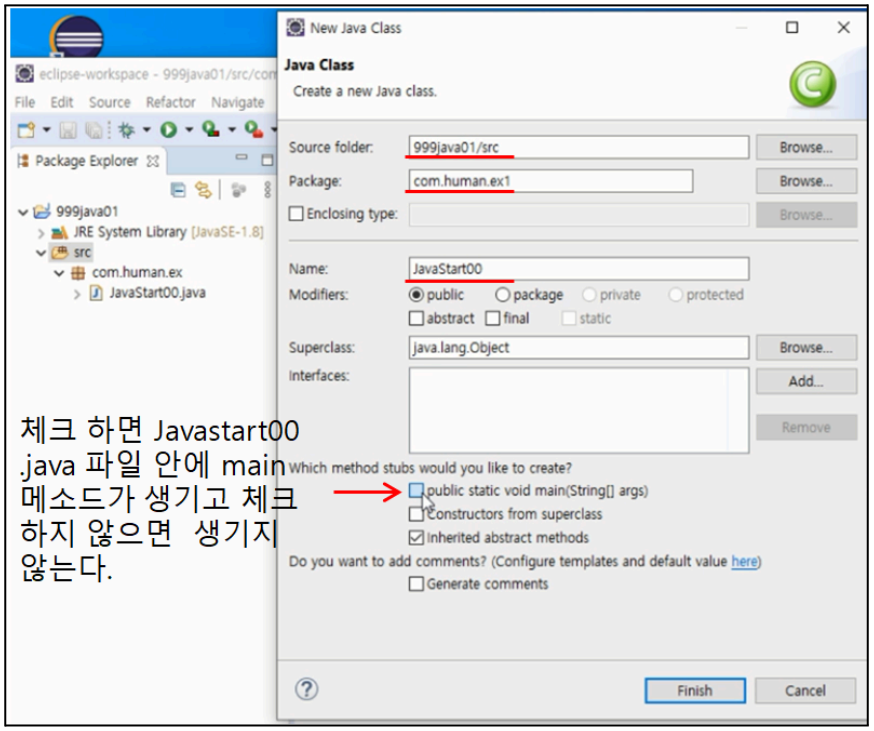


변경하지 않고 기본으로 제공해 주는 workspace 폴더는 다음과 같다. C:\Users\cloud2019\ eclipse-workspace (보통 사용자(user)폴더 밑에 사용자 계정(cloud2019)밑에 eclipse-workspace 폴더를 사용자 계정으로 사용한다. 본인이 위치를 변경 하였다면 변경한 위치에 존재한다.) 우리는 작업폴더를 work폴더로 설정 하였다. 워크스페이스(work) 폴더에서 프로젝트명과 동일한 999java01폴더를 확인할 수 있다.



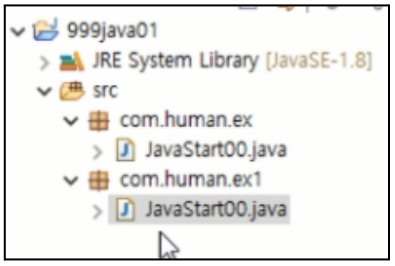
work폴더의 999java01폴더에 들어가 보면 왼쪽 이클립스에서 src 폴더에 해당하는 src 폴더를 확인 할 수 있다.

아래 이미지 처럼 src 폴더를 계속 클릭해서 들어가면 패키지에서 확인할 수 있는 com >> human >> ex 폴더를 모두 확인 할 수 있다. 그리고 ex 폴더 안에 있는 자바 파일을 확인 할 수 있다.



체크 하면 Javastart00 .java 파일 안에 main 메소드가 생기고 체크 하지 않으면 생기지 않는다.

왼쪽 이미지 처럼 새로운 패키지의 새로운 클래스 파일을 만들면 아래 이미지 처럼 새로운 패키지에 파일이 만들어 진다



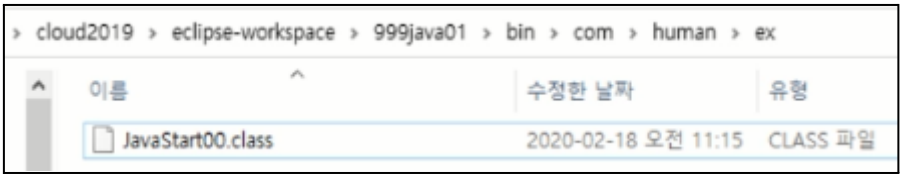
파일안에 메인 메소드를 추가하고 System.out.println(“hello world”);

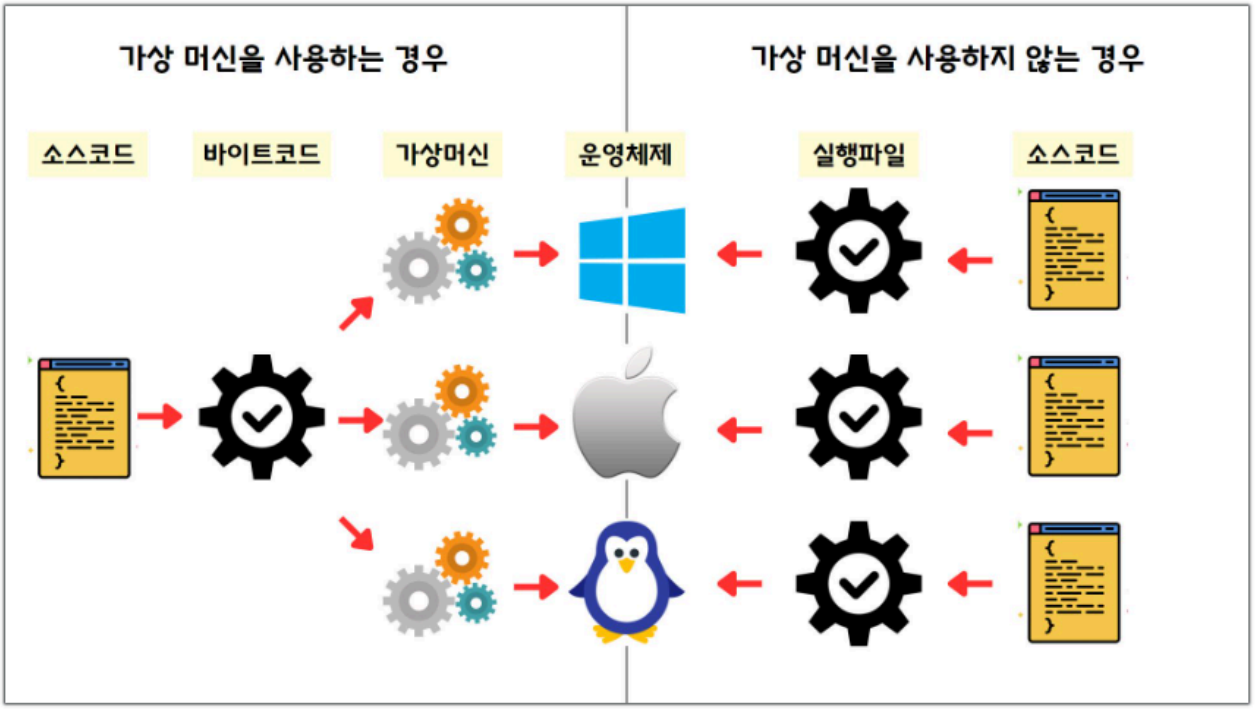
를 추가해보자.

둘 다 메인이 포함된 자바 파일이여서 동시에 실행 시킬 수 없다. 실행을 원하는 파일 위에서 아까 처럼 마우스 오른쪽 클릭 Run As Java Application을 선택해서 원하는 파일을 실행 시킬 수 있다. 실행하고 나서 아까의 src 폴더에 가보면 ex1 폴더와 자바 파일이 생성되어 있는 것을 확인할 수 있다.

패키지가 다르면 같은 이름으로 자바파일이나 클래스를 만들수 있다.

한번 실행하고 나면 다음 이미지 처럼 bin >> com >> human >> ex 폴더에 javaStart00.class 파일이 src폴더 구조와 동일하게 만들어져 있는 bin 폴더를 확인 할 수 있다. 실제로 자바코드가 실행할 때는 컴파일된 빈폴더 안의 .class파일을 이용해서 실행 된다.





실제 실행 단계를 확인해 보면 다음과 같다. 내용에 나오는 javac.exe와 .java.exe와 같은 실행파일은 jdk를 설치하면 설치되는 파일들이다.

본인이 만든 코드(.java 파일)-> 컴파일(javac.exe) 바이너리코드로 변환해 주는 프로그램 -> 컴퓨터가 이해하기 쉬운언어 바이너리 코드(.class)-> java.exe 가상머신(가상 운영체제)로 자바 프로그램을 실행 한다.

이클립스는 이 모든 작업을 자동으로 해주는 통합 개발 소프트웨어이다.

컴파일은 사람이 작성한 코드를 기계가 알기 쉬운 0과 1로 이루어진 바이너리코드로 변환해 주는 프로그램이다.

문제풀이>>

1. 작성한 자바코드의 확장자는 무엇인가?
2. 컴파일이란 무엇인가?
3. 자바파일을 컴파일할때 사용하는 파일은 무엇인가?
4. 컴파일된 파일에 확장자는 무엇인가?
5. 컴파일된 파일을 실행 시킬수 있는 가상머신 화일 이름은 무엇인가?
6. 자바프로그램을 구현하여 실행하는 방법은 무엇인가?
7. 패키지의 의미를 설명하시오.
8. 자바코드는 반드시 어디 안에 기술하여야 하는가?
9. 프로그램의 시작위치와 끝나는 위치를 설명해 보자.

```
package com.human.ex;
public class JavaStart00 {
    public static void main(String[] args) {
        System.out.println("hello world");
    }
}
```

10. 왼쪽 이미지의 소스 코드 파일의 위치와 이름은 src폴더 기준으로 어떻게 되는가?

> 03. 프로그램 관련 용어 살펴보기

프로그램 용어가 특별히 중요해 보이지는 않아도 누군가 본인에게 물어 본다면 답할 수 있어야 한다. 설명 할 수 없는 것은 모르는 것이다. 라는 유명한 말이 있다. 설명을 하다 보면 본인 스스로 정리가 되어서 전체를 이해하는데 도움이 될 것이다. 프로그램관련 용어를 살펴 보자.

프로그램이란 무엇인가?

결과를 얻기위해 정해진 순서대로 진행되는 과정을 프로그램 이라 한다. 예를 들어 우리가 다이어트 프로그램을 따라서 순서대로 진행하면 살이 빠지게 될것 이고, 요리 만드는 프로그램에 따라 순서대로 진행 한다면 요리가 만들어 질 것이다.

프로그램 실행이란?

프로그램 순서대로 하나 하나 진행하여 최종 결과를 얻는 과정을 실행이라 한다.

컴퓨터 프로그램이란?

사람이 컴퓨터에게 어떤 특정 작업을 시키고 싶을 때 컴퓨터가 해야 할 일들을 순차적으로 상세히 기술한 결과물이 있어야 되는데 이를 컴퓨터 프로그램이라 한다.

여기서 특정 작업이란? 컴퓨터에서 할수 있는 동영상 보기, 음악 듣기, 문서 작업 하기, 게임 하기 등을 의미 한다.

프로그램 언어란 무엇인가?

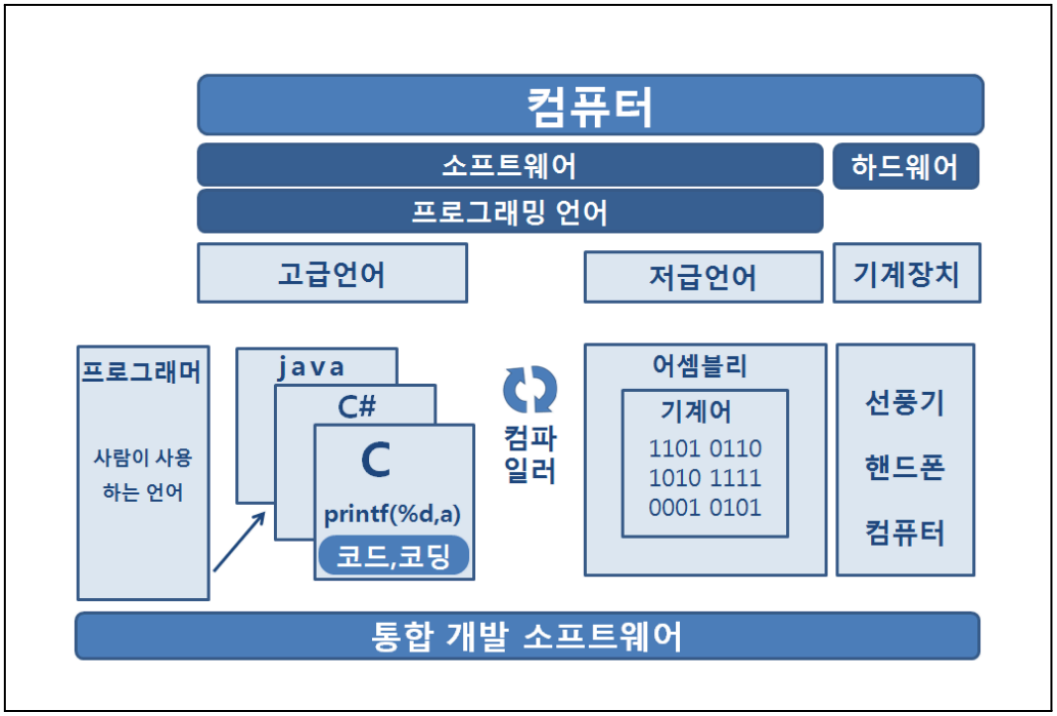
컴퓨터 프로그램을 만들때 사용하는 기호를 컴퓨터 프로그램 언어라고 한다. 컴퓨터 프로그램을 만들면 프로그램 내용 대로 컴퓨터가 한 줄 한 줄 해석해서 컴퓨터에서 원하는 결과를 얻을 수 있다. 프로그램 언어 명령어 하나 하나가 어떻게 하라는 명령어가 된다. 이때 사용하는 대표적인 프로그램 언어로 c언어, java, 파이썬, 자바스크립트 등이 있다.

컴퓨터가 컴퓨터 프로그램을 한줄 한줄 순서대로 실행 하면 컴퓨터 사용자가 게임도 할 수 있게 되고, 동영상 편집도 할수 있게 되고, 문서편집도 할 수 있게 된다.

현실 세계의 언어를 생각해 보면 한국어는 한국 사람들이 서로 의사 소통하기 위해서 만든 언어이고 영어는 영국, 미국 사람이 서로 의사 소통하기 위해서 만든 언어이다. 중국어는 중국 사람이 서로 의사 소통을 하기 위해서 만든 언어이다. 컴퓨터에게 동영상 좀 틀어 달라고 아무리 이야기해도 컴퓨터는 알아듣지 못한다. 컴퓨터가 알아 들을 수 있는 언어로 동영상을 틀어 달라고 해야 한다. 컴퓨터가 알아들을 수 있는 언어가 프로그래밍 언어 이다.

프로그램 언어란? 컴퓨터 같은 기계장치와 의사 소통 하기 위해서 만든 언어이다. 우리가 미국사람과 대화를 나누기 위해 영어를 공부 하듯이 컴퓨터와 대화를 나누고 싶다면 프로그램 언어를 배워야 한다. 결국 프로그램은 프로그램 언어로 만들어야 컴퓨터 같은 기계장치에서 동작할 수 있다. 프로그램 언어는 현실 세계의 언어처럼 하나만 존재하는 것이 아니고 여러개 존재한다. 이 책에서는 많은 프로그램 언어 중 자바를 배우게 된다.

지금까지 프로그램, 컴퓨터 프로그램, 컴퓨터 프로그램 언어의 뜻을 정의해 보았다. 이해 될때까지 여러번 읽어 보자.



상위 이미지는 소프트웨어,컴퓨터 등이 어떻게 구성되어 있고 어떤 과정을 거쳐서 우리가 사용할 수 있게 되는지 표시한 그림이다.

그림에 대한 설명은 다음과 같다. 컴퓨터는 크게 소프트웨어와 하드웨어로 나누어져 있고 하드웨어는 키보드 마우스 모니터 같은 기계장치로 구성되어 있고 소프트웨어는 프로그래밍 언어로 만들어져 있으며 크게 고급 언어와 저급 언어로 구성되어 있다. 저급 언어는 사람이 알아 보기 어려운 0과 1로 이루어진 어셈블리, 기계어이며 고급 언어는 사람이 알아보기 쉬운 언어로 자바 C# 파이썬 자바스크립트 등이 있다. 프로그램을 제작하려면 사람이 알아보기 쉬운 고급 언어로 작성한 후 컴파일을 통해서 하드웨어 기계장치에서 동작하는 저급 언어로 만들어야 한다. 프로그래머가 고급언어로 작성하면 컴파일러가 저급언어, 기계어로 바꾸어 기계 장치에 넣으면, 기계장치가 기계어를 한 줄 한 줄 해석해서 사용자가 원하는 결과를 모니터에 보여준다. 이 복잡한 과정을 쉽게 할 수 있도록 도와주는 소프트웨어가 통합 개발 소프트웨어(eclipse)이다.

소프트웨어란?

컴퓨터 프로그램 언어로 만들어진 컴퓨터에서 사용하는 프로그램으로 모니터 안에서 클릭하면 동작하는 다양한 프로그램들을 의미한다. 즉, 컴퓨터 안에서 컴퓨터를 조작해서 사용자가 원하는 결과를 얻어내는 것을 목표로 하는 컴퓨터 프로그램을 의미한다.
여러분들이 날마다 하는 게임, 브라우저, 메신저, 한글, 워드, 엑셀, ppt, 포토샵 등과 같이 컴퓨터 모니터 안에서 실행되고 있는 응용 프로그램을 말한다. 소프트웨어, 응용 프로그램, 컴퓨터 프로그램 다 같은 의미로 보아도 크게 문제가 없다.

코드 란? 컴퓨터 프로그램을 컴퓨터 프로그램 언어로 작성해 놓은 결과물을 말한다. 코드를

다른 이름으로 소스 또는 소스코드라 이야기 한다. 자바로 예를 든다면 컴퓨터 프로그램을 자바로 구현한 프로그램 결과물을 코드라 한다. 코드 짜는 중이야. 코드좀 만들어줘. 만든 코드를 제출하세요. 등과 같은 용어를 종종 사용한다.

이전 화면에 'hello world'를 출력하기 위해서 작성한 문자들이 코드에 해당한다.

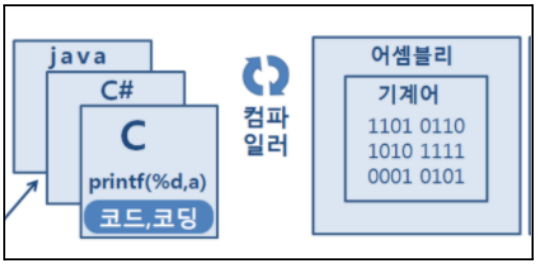
코딩이 란? '코드를 작성 하는 중' 을 의미한다. 코드를 써내려가는 작업을 코딩이라한다. 지금 코딩중이야. 같은 용어를 종종 사용한다.

프로그래머 란? 코드로 프로그램 작성하는 사람을 의미 한다.

하드웨어 란? 우리 눈에 보이는 모니터 키보드 마우스 본체 등 딱딱한 기계 장치들을 의미한다.

기계장치 란? 시계 핸드폰 컴퓨터 전자렌지 등과 같이 전기를 이용해서 사용자의 요청에 따라 어떤 일을 자동으로 처리해 주는 장치들을 의미한다. 이런 기계장치에게 명령을 내리리기 위해서 프로그램을 사용한다.

고급언어 저급언어 란? 프로그램언어는 고급언어와 저급언어로 구성되어 있고 고급 언어는 사람이 알아보기 쉬운 언어이고 저급 언어는 기계장치 컴퓨터가 알아 보기 쉬운 언어이다. 대표적인 고급언어로는 자바 씨샵 씨 등이 있고 저급 언어로는 0과 1로 이루어진 기계어와 어셈블리어가 있다.



컴파일러 란? 고급 언어를 0과 1로 이루어진 저급 언어로 변환해 주는 프로그램을 의미한다 고급 언어는 결국에 0과 1로 이루어진 기계어로 변경되어야 기계장치에서 사용할 수 있다. 보통 프로그래머가 고급언어로 프로그램을 작성하면 컴파일러가 0과 1로 이루어진 저급 언어인 기계어로 변환해 준다. 저급 언어를 사람이 직접 구현할 수도 있지만 저급 언어는 0과 1로 이루어져 있어 사람이 프로그램 하거나 이해하는 데는 어려움이 있어 저급 언어보다는 고급언어로 작성한다. 따라서, 프로그래머가 알아보기 쉬운 고급언어(자바)로 작성하면 컴파일러가 0과 1로 이루어진 기계어로 변환 한 다음 기계어를 기계장치에 넣어 프로그램을 실행 시키다.

컴퓨터에게 화면에 hello world좀 출력해줘 라고 하면 컴퓨터가 알아서 화면에 hello world를 출력해 주면 좋겠지만 컴퓨터는 사람의 말을 이해할 수 없고 사람도 컴퓨터의 말을 이해할 수 없다. 00101100101010111 이게 컴퓨터 언어로 화면에 hello world를 출력해 달라는 이야기라면 0010101010111 이것의 의미를 사람이 인식하기는 어렵다. 위쪽 코딩 이미지 처럼 사람이 사용하는 언어는 아니지만 사람이 알아보기 쉬운 프로그램 언어로 작성한 다음 00101100101010111와 같은 기계어로 변환하는 컴파일 과정을 거친다면 누구나 쉽게 컴퓨터 프로그램을 작성 할 수 있을 것이다.

결과적으로 우리가 컴퓨터에서 돌아가는 소프트웨어를 만들려면 에디터로 고급언어 자바를 이용하여 코드를 작성한 .java파일을 컴파일 (javac.exe)하여 0과 1로 이루어진 기계어

바이너리 .class 파일로 만든 다음 .class파일을 가상 머신(java.exe)을 사용하여 실행 시키면 실행 결과를 얻을 수 있다. 고급언어를 저급언어로 변경하는 과정을 컴파일이라 하고 이때 사용하는 프로그램을 컴파일러라 한다.

통합 관리 프로그램이란? 이클립스 같은 프로그램 제작을 쉽게 할 수 있도록 도와주는 응용 프로그램 소프트웨어를 의미한다. 통합 관리 프로그램은 고급 언어로 프로그램을 작성하면 자동으로 기계어로 컴파일하여 바로 컴퓨터에서 실행 결과를 볼 수 있게 해주는 소프트웨어 이다.

실행이란? 작성한 코드를 컴퓨터가 한줄 한줄 읽고 적용 시킨다는 의미이다.

프로그램 언어를 작성해서 컴파일하는 도중에 잘못된 문자가 있으면 문제가 발생 하는데 이런 잘못된 문자를 버그라 하고 잘못된 문자인 버그를 찾는(잡는) 과정을 디버그,디버깅 이라고 한다.

버그란? 프로그램 언어로 프로그램 작성시 잘못된 프로그램 코드를 의미한다.

디버그, 디버깅이란? 잘못된 코드를 올바르게 수정하는 과정을 의미한다.

컴퓨터의 용도는 결국에는 정보 처리이다. 웹사이트를 생각해 보면 정보를 모아 두고 처리해서 사용자가 원하는 형태로 결과를 보여주는 작업을 하는 것이다. 요즘 이런 회사들이 떼돈을 벌고 있다. 구글, 네이버, 다음 등 자세히 보면 정보처리 작업을 하고 있는 것이다. 3+6 에서 3,+,6 이런 것들은 정보에 해당 하고 처리란 실제 계산해서 9라는 결과를 얻는 것을 의미한다. 원하는 데이터를 웹사이트에 요청해서 결과를 받는 것도 잘생각해 보면 정보처리이다. 그래서, 컴퓨터 관련 과목들을 정보 처리 과정이라 이야기 하고 해당 자격증 이름을 정보 처리 기사 자격증이라고 한다.

정보처리 기사가 없는 사람이라면 공부해서 꼭 취득 하기 바란다. 2019년 통계에 따르면 모든 직종이 안 좋은 취업률을 보이고 있으나 프로그램 쪽은 올라가고 있다. IMF때에도 프로그램 직종은 문제가 없었다. 미래에는 모든 직종은 사라지고 프로그래머만 남게 될 것이고 그 날이 멀지 않았다. 은행에서 이제는 프로그래머로만 구성된 지점을 선보였다. 인건비가 올라가면서 지점에 모든 사람이 프로그래머로 교체되고 있다. 프로그램은 모든 분야에서 사용되는 기초과목이 되었다. 프로그램을 벗어나서 아무것도 생각할 수 없다. 프로그램을 포기한다는 것은 앉아서 할 수 있는 모든 일을 포기 한다는 것과 같은 이야기이다. 포기하지 말고 끝까지 '살아 남으라'는 이야기를 해주고 싶다.

본인이 학력이 부족해서 정보처리기사 자격증을 따지 못한다면, 독학사, 학점은행제, 사이버대학 k-mooc등을 이용해서 무료나 싼가격으로 빠른 시간에 학사를 취득할 수 있다. 남에 일이라 생각하지 말고 본인 일이다. 시간만 투자하면 쉽게 학사를 취득할 수 있다.

자격증은 정보처리기사 하나만 있으면 될거 같다. 다른 잡다한 자격증은 없어도 된다.

행운을 빈다. 중간에 모르는 것이 있으면 웹에서 검색해 보자. stack overflow 나 많은 한국사이트가 나올 것 이다. 구글, 네이버, 다음 등과 같은 검색 사이트를 검색 하는 습관을 가지자. 요즘은 AI에게 먼저 물어보는 것도 좋은 생각이다. 잭 안드리카라는 소년은 15살에 인터넷 검색으로 췌장암 조기진단방법을 찾아냈다. 인터넷을 찾아 집에서 핵융합을 한

사람도 있다. 프로그램의 모든 답은 구글 검색과 AI에 있고고 영어 공부도 열심히 하자. 이 과정이 마무리되면 자료구조를 구현하는 방법을 배워 보도록 하자. 자바와 자료구조만 잘해도 전공자 보다 낫다는 이야기를 들을 것이다. 모르는 코드를 chatgpt에게 물어보면 한줄 한줄 잘 설명해 줄것이다.

실제로 우리가 사용하고 있는 컴퓨터 프로그램을 만들기에는 아직 멀고 험한 길이 기다리고 있지만 참고 견디면 생각보다 쉽게 만들 수 있을 거라 생각된다. 본인의 인생에서 방향 전환은 있을 수 있으나 포기는 있을 수 없다. 방향 전환을 잘 하려면 가지 않은 길도 잘 알고 있어야 가지 않는 이유를 명확히 알 수 있다. 본인 이 프로그램 일을 하지 않는다 해도 프로그램 지식은 본인이 살아가는데에 많은 영향을 미칠 것이다. 절대로 프로그램을 포기하지 않기를 바란다. 중간에 그만두는 수많은 학생들을 보면 적성에 안맞다는 이유로 인생을 편히 살려고 하는 경향이 있다. 대부분 프로그램 말고 뭘 할거냐라는 질문에 답을 하지 못한다. 답을 하더라도 식견이 너무 좁다. 고등 학교때 국영수 과목이 인생을 사는데 크게 필요가 없다고 생각한 적이 있다. 지금 생각해 보니 국영수 과목은 도서관에 있는 모든 책을 읽을 때 가장 많이 막히는 부분에 대한 선행 학습 이었던 것 같다. 프로그램 역시 이미 사람의 기본 요소 중 하나가 되었다. 본인들이 프로그램을 포기 한다고 해도 언젠가 사회에서 지식인으로 살아 남고자 한다면 국영수 뿐만 아니라 프로그램 책을 다시 펴고 공부할 날이 반드시 올 것이니 포기하지 말고 끝까지 프로그램을 공부하자. 이 책은 프로그램 경험이 없는 사람들에게 경험을 제공해 주는 것을 목표로 한다. 왕도는 없다. 많이 읽고 많이 치자. 이해가 안될때는 ai에게 이해가 안돼니 비슷한 다른 예제를 만들어 달라고 해서 확인해보고 이해가 여전히 안돼면 더 쉽게 가르쳐 달라고 이야기하자.

연습문제

1. 프로그램이란 무엇인가? 컴퓨터 프로그램이란?
2. 프로그램 언어란 무엇인가? 프로그램 언어란?
3. 소프트웨어란? 하드웨어 란? 코드 란? 코딩이 란?
4. 실행이란?
5. 프로그래머 란? 기계장치 란?
6. 컴파일러 란? 고급언어 저급언어 란?
7. 버그란? 디버그, 디버깅이란?
8. 통합관리 프로그램이란?

> 04. 프로그램의 기초 문법 - 주석

이후 주어지는 코드를 HelloWorld.java파일을 만들고 내용을 탭을 이용해서 띄어쓰기까지 동일하게 작성해 실행해 보자.

HelloWorld.java

```java
package com.human.ex;

/*
    여러줄 주석이다.
    이 프로그램은 프로그래머가 되고자 하는 사람이 맨처음 작성하는
    프로그램이다.
    2020.03.05
 */
public class HelloWorld {
    public static void main(String[] args) {
        // 한줄 주석이다
        System.out.println("HelloWorld");

    }

}
```

주석은 프로그램과 관련 없는 내용을 코드에 기록 할때 사용 한다.

상위 코드는 이전에 콘솔 창에 HelloWorld를 출력 하는 프로그램이다.

만약 이 프로그램 코드에 프로그램과 관련없는 내용들이 들어가면 문제가 발생 한다. 툴은 관련 없는 내용에 빨간 줄을 보여 줄 것이고 컴파일 하면 에러가 발생할 것이다. 하지만, 프로그램을 하다 보면 프로그램 코드 말고 기술 해야 할 것들이 생긴다. 오늘 어느 부분까지 코드를 작성 하였는지 적거나 특정 코드를 작성한 이유가 뭔지등을 자바 코드에 기록 하고자 한다면 기록한 내용으로 인해 에러가 생길 긴다. 이런 문제를 해결하기 위해서 주석 이라는 것을 사용하여 프로그램과 관련없는 내용을 기술할 수 있도록 할 수 있다.

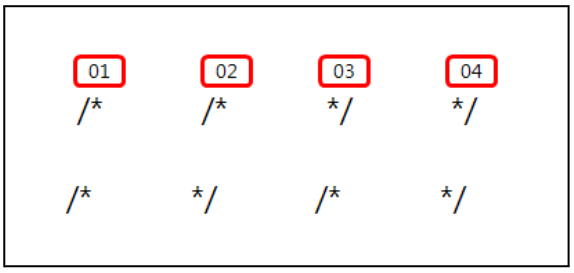
주석은 한줄 주석과 여러줄 주석이 있다.

한줄 주석은 // 를 사용 한다. // 뒤에 주석을 기술하면 된다. //를 사용하게 되면 //가 사용된 줄은 프로그램 컴파일 할 때 제외 된다.

여러줄 주석은 /* 외 */ 안에 여러줄 주석 내용을 기술 한다. 기호(/* */) 사이에 있는 여러 줄의 내용이 모두 주석으로 처리 된다. 한 프로그램 안에서 한줄 주석과 여러줄 주석을 여러번 사용해도 되지만, 여러줄 주석을 겹쳐 쓰게되면 문제가 발생한다. 반드시 주석을 열었으면 닫은 다음에 다시 열어서 사용해야 한다.

다음 그림을 보면 아래 줄은 맞게 주석을 사용한 것이고 위 줄은 주석을 잘못 사용한

예이다. 주석을 잘못 사용한 이유를 설명해 보자.



윗 부분에서 01번의 짝이 03번이 되어서 02번 주석 기호는 주석 처리되어 주석기호의 기능을 못하여 04번과 짝을 이루지 못한다.

02부분이 주석처리 되어서 04의 짝이 없어진 것이다. 초임들이 많이 하는 실수다. 실수하지 않도록 꼭기억해 두자.

상위 코드를 해석하면 다음과 같다.

상위 코드에서 1번라인 패키지 package com.human.ex;는 해당 파일이 존재하는 폴더 위치를 나타낸다. 프로젝트 폴더에 들어가 보면 com폴더 밑에 human폴더 밑에 ex 폴더 밑에 HelloWorld.java파일이 있는 것을 확인 할수 있다. 소스코드 .java파일이 많아 지면 파일들을 폴더로 관리해야 사용하기 편리해 진다. package를 이용하여 java파일을 폴더 별로 관리 할 수 있다.

eclipse를 사용하면 쉽게 프로젝트와 관련된 파일과 폴더를 관리하고, 만든 프로그램을 실행해 보고, 결과를 확인 할 수 있다. 코드 입력시 잘못된 부분(에러)이 발생하면 빨간 줄이나 메시지가 생기고 코드를 올바르게 고치는(디버깅) 과정을 쉽게 할 수 있다.

코드에 {}모양의 중괄호 블럭이 있다. 관련있는 코드들을 묶어 놓은 것이다. 9번 라인 public class HelloWorld{} 클래스를 선언 하는 문구인데 클래스 관련 부분의 코드 블럭을 {}로 묶은 것이다. 클래스는 관련있는 코드들을 묶어 놓은 것이다. 나중에 좀더 자세히 설명할 예정이다. 중요한건 중괄호는 관련있는 코드를 묶어 놓은 것이고 클래스 블럭, 메소드 블럭, 조건문 블럭, 반복문 블럭 등 다양한 관련있는 코드를 묶는다. 10번 라인은 public static void main(String[] args) {} 메소드를 선언한 부분으로 메소드 관련 코드를 묶은 것이다. 메소드와 관련된 코드를 {}중괄호로 묶어 놓은 블럭이다. 메소드는 함수와 같은 의미 이고 메소드나 함수는 관련있는 코드들을 미리 정의해 놓고 필요할때 호출해서 사용하는 코드 블록이다. 나중에 자세히 설명할 예정이다. 중요한 것은 이 부분 메인 메소드가 프로그램의 시작점 이라는 것이다.

프로그램이 시작되면 메인 메소드에서 시작해서 한줄 한줄 실행되다가 메인 메소드가 끝나면 프로그램은 종료된다. 즉, 프로그램은 메인 함수의 '{' 중괄호에서 시작해서 메인 메소드가 끝나면 즉, 메인 함수 중괄호가 닫히면 '}' 종료 한다.

상위 코드에 나와 있는 코드들 class public void와 같이 프로그램에서 문법적으로 사용하도록 미리 약속된 명령어들을 키워드 라하고, 현재 키워드로 사용되고 있지 않지만 앞으로 키워드로 사용될 예정인 명령어 들을 예약어라 하고, main out System 와 같이 식별의 용도로 사용되는 것들을 식별자라 하고 식별자는 프로그램안에서 한개만 선언하여 사용할 수 있는데 이미 사용하고 있으므로 다시 사용할 수 없다.

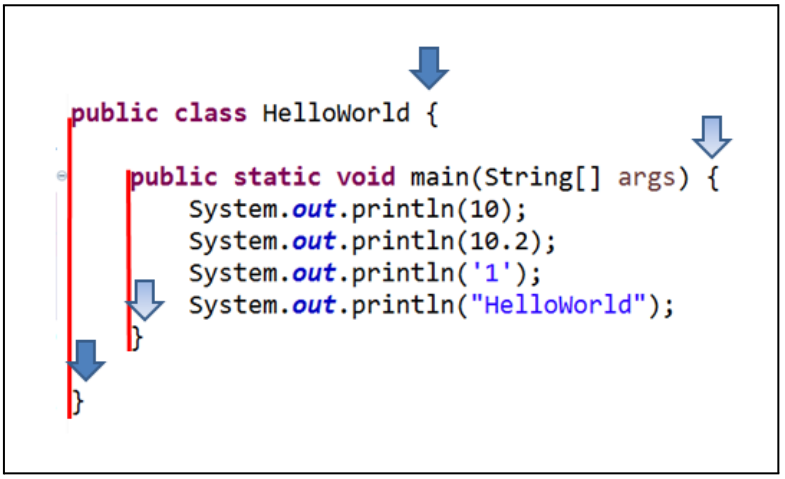
식별이란? 여러 개중 하나를 구분하는 것을 의미한다. main은 여러 메소드중에서 main이라는 메소드를 식별하기 위해서 사용된것이고 out은 여러 변수중에 out이라는 변수를

식별하기 위해서 사용한 것이고 System은 여러개의 클래스 중에 System이라는 클래스를 식별하기 위해서 사용한 것이다.

main 메소드 안의 System.out.println()은 소괄호안에 내용을 모니터 화면에 출력하라는 의미의 메소드이다. 일단 이미 약속된 명령어라 생각하면 된다.

파일 이름과 해당 파일에 있는 클래스의 이름은 반드시 같아야 하고 자바는 대소문자를 구분한다. 초임들이 많이 하는 실수 이니 꼭 기억해 두자.

아래와 같이 입력하고 실행 시켜서 콘솔 창에 무엇이 출력 되는지 확인해 보자. 확인이 끝났다면 println 대신에 print로 바꿔서 실행해 보고 이전과 어떻게 다른지 확인해 보자. 소괄호 안에 다양한 정수, 실수, 문자, 문자열과 같은 다양한 데이터를 화면에 출력 할 수 있다. 다음 코드를 작성하고 실행해 보자.



프로그램 기초 문법은 다음과 같다.

1. 프로그램은 main 메소드에서 시작해서 main 메소드에서 끝난다.
2. 프로그램은 위에서 아래로 실행된다.
3. 프로그램 명령문의 끝은 ;(세미콜론)을 붙여야 한다.
4. 관련있는 코드는 중괄호로 묶는다.
5. 종괄호 안에 있는 코드들은 탭으로 들여 쓰기한다.
6. 중괄호 {}는 시작 위치와 같은 위치에서 닫는다.

클래스와 메소드의 선언문 중괄호 끝에서는 마지막에 ;를 붙이지 않아도 되지만 대부분의 프로그램의 명령문 들은 ; 으로 끝난다.

중괄호 {}는 시작 위치와 같은 위치에서 닫는다. 상위 코드 화살표 부분을 확인해 보아라 중괄호가 시작한 줄의 맨 앞 부분과 같은 라인에서 닫는다.

관련있는 코드는 중괄호로 묶고 중괄호 안에 있는 코드들은 탭으로 들여 쓰기한다.

빨간색 라인을 보면 안쪽에 코드를 전부 탭으로 한 칸 띄웠다. 탭은 키보드 맨 왼쪽 중간 부분에 tab이라 쓰여져 있는 키이다. 여러번 띄어쓰기 하는 것을 탭이라 한다. 상위 코드의 빨간 줄을 기준으로 중괄호 안에 코드들은 탭으로 모두 띄어쓰기 되어 있다. 상위 기초 문법을 이해하고 외워 보자.

다음 코드도 작성해 보자. 코드 중간쯤 보면 System.out.println(14)를 두 줄로 기술하였다. 보통 1줄로 기술하여야 되지만 특정부분에서 상위와 같이 두줄로 기술하여도 문제 없이 동작하지만 특별한 경우가 아니면 한줄로 기술하는 것이 좋다. 하나의 실행 코드를 여러 줄로 기술하고 싶으면 '.' , '=', ',', 괄호 다음에 줄을 바꾸어 쓸 수 있고 실행 명령이 끝나는 맨 마지막에 ;를 붙이면 된다. 보통 한줄로 써야할 코드가 너무 길어 여러줄로 기술할때 사용한다.

```
package com.human.ex;    //폴더 경로
public class HelloWorld2 {//클래스
    public static void main(String[] args) {//main
        System.out.println(true);//화면에 출력
        System.out
        .println(14);
        System.out.println(14.3);
        System.out.println('a');
        System.out.println("helloWorld");
    }
}
```

```
1  package com.human.ex;
2  public class JavaTest02 {
3      public static void main(String[] args) {
4          // . 괄호 = , 에서 줄을 변경하여 사용할 수 있다.
5          System.
6              out.
7              println
8              (
9                      true
10                     )
11             ;
12         Sys
13         tem.out.println(true);
14     }
15 }
```

왼쪽 코드에서 잘못된 코드를 찾아보자.

12,13라인에 줄바꿈 한 것은 잘못된 코드이고 나머지는 정상적인다.

큰이유가 없다면 한줄로 기술하자.

페이지에 있는 코드들을 모두 쳐서 확인해 보자.

다음 코드를 확인해 보고 읽어 보자. 최상의 package com.human.ex; 작업한 파일이 존재하는 폴더라고 생각하면 된다.

main메소드가 중괄호 안에 포함된 public class HelloWorld3{ 에서 HelloWorld3은 클래스 이름을 의미 하는데 메인 메소드가 포함된 클래스의 파일 이름은 동일 해야 정상적인 문법 이어서 이 프로그램의 파일이름은 HelloWorld3라고 생각해도 무방하고 이파일의 위치는 해당 프로젝트 src폴더 밑에 com 아래 human아래 ex 폴더 일 것이다.

`public static void main(String[] args) {`은 메인 메소드라고 부르며 프로그램이 시작하는 부분이라고 생각해도 된다.

자바의 `System.out.println`과 `System.out.print` 메서드는 콘솔에 출력을 할 때 사용됩니다. `println`은 출력 후 줄을 바꿔주고, `print`는 줄을 바꾸지 않고 같은 줄에 이어서 출력합니다. 이 차이를 이해하기 위해 간단한 따라치기 예제를 만들어 보겠습니다.

아래의 코드를 직접 따라 쳐보면서, 실행 결과를 확인해보세요.

```
public class PrintExample {
    public static void main(String[] args) {
        // System.out.print 예제
        System.out.print("안녕하세요, ");
        System.out.print("여기는 ");
        System.out.print("자바입니다.");

        // 줄바꿈을 위해 빈 println 추가
        System.out.println();

        // System.out.println 예제
        System.out.println("안녕하세요, ");
        System.out.println("여기는 ");
        System.out.println("자바입니다.");
    }
}
```

예상 출력:

이 코드를 실행하면, 아래와 같은 결과를 볼 수 있습니다.

```
안녕하세요, 여기는 자바입니다.

안녕하세요,

여기는

자바입니다.
```

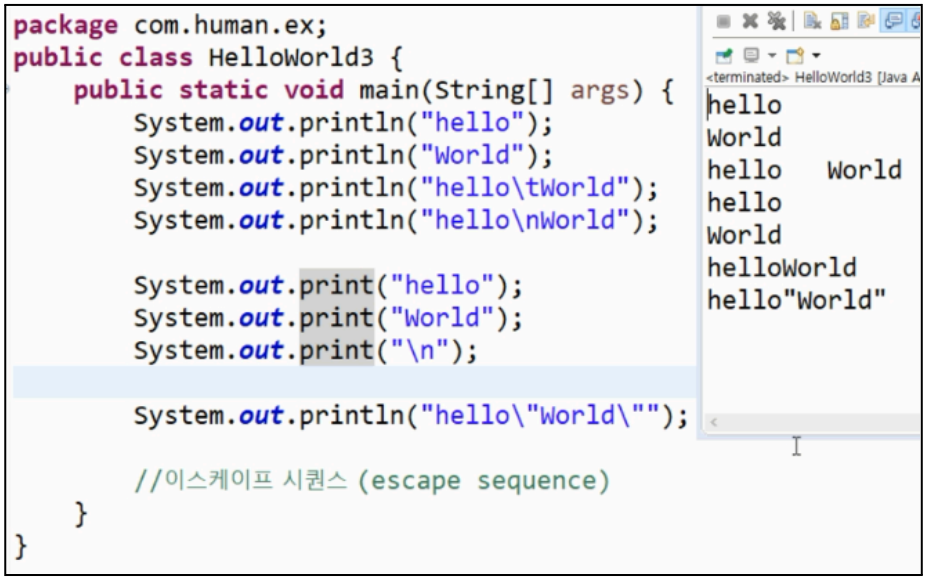
설명:

1. **`System.out.print`**:
    - `print` 메서드는 줄바꿈 없이 문자열을 출력합니다. 따라서 세 개의 `print`가 모두 같은 줄에 출력됩니다.
2. **`System.out.println`**:
    - `println` 메서드는 문자열을 출력한 후 줄바꿈을 합니다. 따라서 각 `println`

호출마다 새로운 줄에 출력됩니다.

이 간단한 예제를 통해 System.out.print와 System.out.println의 차이를 이해할 수 있습니다.

코드에 System.out.println() 메소드는 소괄호안에 오는 다양한 매개변수 자료형을 화면에 출력 한다. 이것과 비슷한 System.out.print() 메소드도 있다. 차이점을 보면 끝에 ln이 있고 없고의 차이이다. line 의 약자로 println의 경우 소괄호안의 매개변수를 출력하고 행이 갱신되어 다음 출력 때에 다음 줄에 출력이 되지만 print의 경우는 다음 출력 때 행이 갱신되지 않고 옆으로 이어서 출력 된다.



상의 코드를 작성하고 실행해 보자.

\t은 출력 할 때 탭 만큼 띄어쓰라는 의미이다. 스페이스를 여러개 넣은 것이 탭이라고 생각하면 된다. \n은 엔터 처럼 줄을 바꾸라는 의미이다. \"은 " 따옴표 를 화면에 출력하라는 의미이다. (hello"world")를 화면에 출력하고 싶으면 어떻게 해야할까? 문자열은 쌍따옴표 로 묶여 있어야 하는데 ("hello"world"")이렇게 문자열을 입력하면 "hello"에서 문자열이 끝나서 에러가 난다. 이렇게 엔터, 탭, 쌍따옴표와 같이 출력시 문제가 되는 문자열을 특수한 문자를 추가하여 출력 할 수 있도록 할때 사용하는 방법을 이스케이프 시퀀스 라 이야기 한다. 이스케이프 시퀀스는 코딩중 표기하기 어려운 문자를 표현할때 사용한다. 엔터,탭이 대표적이다.

이클립스 단축키를 확인해서 사용해 보자.

사용키	설명
Ctr + 왼쪽 오른쪽 방향키	단어 기준으로 이동
shift+방향키	영역 선택시 사용
Ctr+shift+왼쪽 오른쪽 방향키	단어 기준 영역 선택시 사용
home, end	현줄에 맨압출 이동, 마지막 줄로 이동
ctr+shift+ + , -	글자 폰트 크기 조절
ctr+z,y,c,v,x	취소,복구,복사,붙여넣기,잘라내기
ctr+f	찾아서 변경하기
ctr+a,ctr+shift+f	모두선택, 코드정렬

여유가 되면 검색을 통해서 다양한 단축키를 익혀 보자.

연습문제>>

1. 주석이란? 주석의 종류와 사용방법은?
2. 현재 단원 처음에 있는 코드1, 9, 10번 라인을 설명하시오.
3. 키워드, 예약어, 식별자의 의미를 설명하시오.
4. 프로그램의 시작점은 어디이며 기술해 보자.
5. {}괄호가 사용되는 용도와 사용되는 곳을 기술하시오.
6. 프로그램에서 사용되는 자료형 4가지를 기술하고 설명하시오.
7. 프로그램 기초 문법 6개를 기술하시오.
8. println과 print의 차이를 설명하시오.
9. 이스케이프시퀀스란 무슨 의미이고 어떻게 사용하는지 기술하시오.
10. 다음은 파일 이름은 helloWorld.java 이다. 잘못된 부분을 찾아 보자.

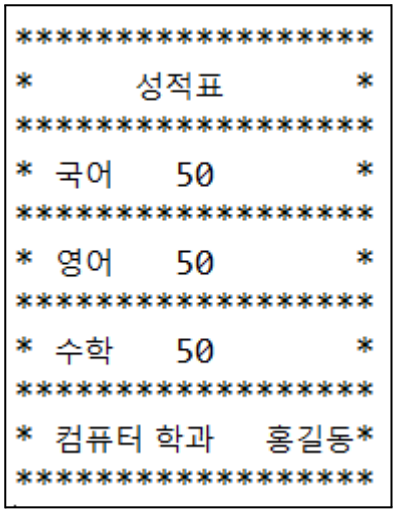
```
public class MyHelloWorld {
  pubIic static void main(string[] args) {
      system.Out.print1n(
                   'Hello
                  World~'
                  )
  }
}
```

11. 상위 코드를 정상적으로 변경하였다면 키워드,예약어,식별자를 구분해보자.

12. 화면에 ‘안녕자바’를 출력해보자.

13. 본인의 정보를 출력하는 프로그램을 만들어 보자. 이름 이메일 전화번호등 명암으로 만들어 출력해 보자.

문제5. 다음 각종 삼각형 사각형 모양을 화면에 출력해보자.



문제6. 다음 성적 표를 콘솔에 출력하시오.

문제7. 다음과 같은 달력을 \t를 이용해서 오늘 날짜 월로 만들어 보자.

일	월	화	수	목	금	토
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	26	27	28
29	30	31				

> 05. 화면 출력 - 기본 문법 1

입력 부분은 사용자가 프로그램의 메인 메소드에 입력하는 부분이고 출력 부분은 입력 내용이 실행 되었을때 화면에 출력되는 결과문이다. ? 부분에 어떤 내용이 기술 될지 생각해 보고 입력하여 확인해 보자.

1. 입력
```
System.out.println(“1”); // 출력하고 줄바꿈
System.out.println(“2”);
System.out.println(“3”);
```

출력
```
1
2
3
```

2. 입력
```
System.out.print(“1”); // 출력하고 줄바꿈 하지 않음
System.out.print(“2”);
System.out.print(“3”);
```

출력
```
123
```

3. 입력
```
System.out.print(“hello”);
System.out.println(“ World”);
System.out.println(“ World”);
System.out.print(“hello”);
```

출력
```
?
```

4. 입력
```
System.out.print(“12”);
System.out.println(“34”);
System.out.println(“56”);
```

출력
```
?
```

5. 입력
```
?
```
출력
```
hello
java
```

6. 다음은 System.out.print(“1”);과 System.out.print(“ ”);
System.out.println(“”); System.out.print(“2”); System.out.print(“3”); 만
사용해서 다음 출력을 완성해 보자.
입력
?
출력
```
  1
 12
123
```

7. 다음은 System.out.print(“*”);과 System.out.println(“”); 만 사용해서 다음 출력을
완성해 보자.
입력
?
출력
```
****
****
****
****
```

8. 다음은 System.out.print(“*”);과 System.out.println(“”); 만 사용해서 다음 출력을
완성해 보자.
입력
?
출력
```
*
**
***
****
```
9. 다음은 System.out.print(“*”);System.out.print(“ ”);System.out.println(“”); 만
사용해서 다음 출력을 완성해 보자.
입력
?
출력
```
   *
  **
 ***
****
```
10. 다음은 System.out.print(“*”);System.out.print(“ ”);System.out.println(“”);
만 사용해서 다음 출력을 완성해 보자.
입력
?
출력
```
  *
 ***
*****
 ***
  *
```

02. 자료형과 관련된 것들

> 01. 컴퓨터의 자료형 및 저장방법

프로그램의 목적이란?

현실 세계의 데이터를 컴퓨터에 넣어 사용자가 필요로 할 때 데이터를 가공하여 원하는 결과를 보여주는 것이 컴퓨터 프로그램의 목적이다.

간단한 예를 들어 생각해 본다면 5와 6이라는 데이터를 사용자에게 입력 받아 두 수를 더한(데이터를 가공한) 11이라는 결과를 사용자에게 보여 준다. 이를 정보처리라 한다.

자료형이란? 현실 세계의 다양한 데이터를 효율적으로 저장하고 처리하기 위해 프로그램에서 미리 정의된 데이터의 형식을 의미합니다.

프로그램에서 사용되는 자료형은 다음과 같다. 예,아니오(true,false) 형태의 값을 저장할 수있는 불리언 (boolean), 소수점이 없는 숫자 정수(int), byte, short, long 되도록 int를 사용하자. 소수점이 있는 숫자 실수 (double), float 되도록 float를 사용하자.
문자(char), 문자 기호적으로 나타낼수 있는 최소한의 단위를 문자라고 한다.
문자열(String) 문자가 모여서 문자열을 이룬다.

자바에는 다양한 자료형이 있는데 본인이 만들어 사용할 때는 다음 4개의 자료형만 쓰자.

불리언, 정수, 실수, 문자열 -> boolean, int, double, String

다양한 자료형을 사용하는 이유는 자료형을 사용하는 이유는 데이터를 알맞게 저장하고 처리하여 프로그램을 효율적으로 작동하기 위해서이다.

요리(프로그램)를 할때 부엌(메모리)에 식재료(사용할 데이터가 담긴 용기(자료형)를 올려놓고 원하는 요리(프로그램)를 만들듯이 컴퓨터 프로그램을 실행할 때에는 메모리에 원하는 자료형의 데이터를 올려놓고 프로그램을 실행한다.

컴퓨터에서 데이터를 사용하려면 메모리에 데이터를 기록 해야 하는데 데이터 종류 별로 메모리에 기록하는 크기와 내부 표현 방법이 달라 개발자가 자료형을 선택해서 데이터를 효과적으로 메모리에 저장할 수 있도록 다양한 형태의 자료형을 사용한다.

자료형이 다양한 이유의 예

요리 할 때 설탕 소금 마늘 양파 상추 등 다양한 재료가 있다고 생각해 보자. 설탕, 소금은 양념통에 담는 것이 편하고, 양파, 마늘은 접시에, 기름 같은 음식재료는 병에 담는 것이 좋고, 상추는 채에 담는것이 편하다. 그리고 양이 많아지면 큰접시에 적으면 작은접시에 담아야 한다. 미리 접시를 준비해 두고 요리사가 어떤 접시를 사용할 것인지 결정하면 접시에 쉽게 재료를 담을 수 있다. 미리 준비 해둔 접시가 자료형에 해당 한다.

프로그램의 자료형이 있는 이유를 요리에 비유 하여 생각해 보면 다음과 같다.

데이터는 식재료, 자료형은 용기, 부엌은 메모리에 해당한다. 요리를 하려면 식재료를 용기에 담겨 부엌에 있어야 부엌에서 요리가 가능하다.

프로그램에서는 메모리에 있는 다양한 자료형의 데이터를 가지고와 원하는 프로그램 결과를 얻는다.

요리를 하려면 부엌에 있는 용기의 식재료를 가지고와 원하는 요리를 한다.

프로그램, 요리 : 목적 (실행 후 최종결과)

메모리, 부엌 : 실제 데이터를 저장하는 작업공간

자료형, 용기 : 작업 공간에서 데이터 저장용도로 사용

데이터, 식재료 : 실제 공간에 저장된 데이터

원하는 프로그램, 원하는 요리 : 최종 결과물

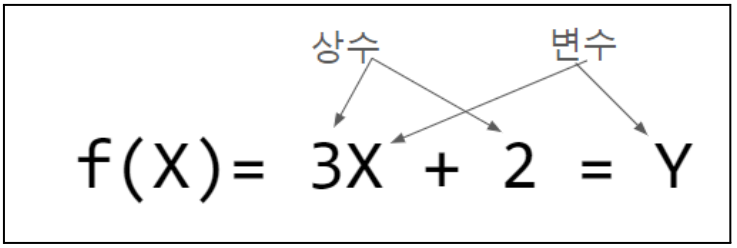
용기를 사용하면 부엌의 작업 공간 확보와 요리 시간 단축을 할 수 있듯이 자료형을

사용하여 메모리 저장 공간을 확보 하고 프로그램 실행 속도를 높일 수 있다.

연습문제

1. 프로그램의 목적은 무엇인가?
2. 자료형이란?
3. 자료형의 종류와 어떤 자료를 담는지 기술하시오.
4. 컴퓨터 프로그램에서 데이터를 사용하려면 어디에 저장해야 하는가?

> 02. 프로그램 개념



함수란? 복잡한 수식을 기호로 표현한 것이다. 값을 입력받아 원하는 결과를 쉽게 얻어내기 위해 기호화한 것이다.

함수 f(X)=3X+2=Y라고 할때 X가 1이면 Y=5, X가 2이면 Y는 8이 된다. X값에 따라 Y값이 변화 한다. 여기서 변하는 X, Y는 변수라 하고, 변하지 않는 3, 2 같은 변하지 않는 수를 상수라 한다.

상수는 변하지 않는 데이터를 의미 한다. 프로그램안에서 한번 기술 하면 프로그램이 끝날때 까지 해당 값을 변경할 수 없다.

변수는 상황에 따라 변하는 데이터를 저장하는 용도로 사용된다.

상수란? 변하지 않는 수를 의미한다.
변수란? 변하는 수를 의미한다.
함수란? 반복되는 복잡한 코드를 미리 선언해 두고 필요할때 마다 호출해서 사용하는 코드 블럭을 의미한다.
객체란? 관련있는 데이터와 메소드를 묶어서 표현한것을 의미한다.
메소드란? 함수가 객체에서 사용되면 메소드 부른다.

상수 (Constant)

- **설명**: 상수란 한 번 정해지면 변하지 않는 수를 의미한다. 예를 들어, "100원"이라는 돈이 항상 100원인 것과 같다.
- **예시**: "파이(π) = 3.14"처럼 언제나 같은 값을 갖는 것

변수 (Variable)

- **설명**: 변수란 변하는 수를 의미한다. 값이 바뀔 수 있는 "상자"라고 생각할 수 있다. 상자 안에 숫자나 글자를 담을 수 있고, 그 내용을 언제든지 바꿀 수 있다.
- **예시**: "상자 A에 5를 넣었어요. 나중에 10으로 바꿀 수 있다."

함수 (Function)

- **설명**: 함수란 반복되는 복잡한 코드를 미리 선언해 두고 필요할 때마다 호출해서 사용하는 코드 블록을 의미한다. 특정한 일을 하는 "작은 기계"라고 생각하면 된다.
- **예시**: "2와 3을 더해줘!"라고 말하면, 이 함수가 "5"라는 결과를 만들어 준다.
- System.out.println()이 함수에 해당한다. 화면에 출력하는 자세한 방법을 몰라도 해당 함수를 이용해서 화면에 원하는 데이터를 출력할 수 있다.

객체 (Object)

- **설명**: 객체란 관련 있는 데이터와 메소드를 묶어서 표현한 것을 의미한다. 여러 가지 정보와 행동을 가지고 있는 "사람"이나 "물건"이라고 생각할 수 있다. 정보는 데이터 행동은 메소드에 해당한다.
- **예시**: "강아지 객체"에는 이름이 "바둑이", 나이가 "2살", 행동으로 "짖기"가 있을 수 있어요

> 02. 상수

상수는 자료형이 무엇인지 식별 가능 하도록 약속된 문법에 맞춰서 기술해야 한다.

다음은 약속되지 않은 상수를 표기하여 에러가난 코드이다. 올바른 상수 표기 법을 알아보자.

```
3 public class JavaStart00 {
4
5     public static void main(String[] args) {
6         System.out.println(안녕);
7         System.out.println(10000000000);
8
9     }
10 }
```

System.out.println(); 에서 그동안 소괄호 안에 아무거나 입력하면 출력이 되는 것 처럼 보였다. 하지만, 왼쪽 처럼 입력하면 문법에 맞지 않아 동작하지 않는것을 확인할 수 있다.

상수 표현 방법이 잘못 되었다. 상수 사용시 올바른 상수 데이터 표현 방법에 맞춰서 기술해야 한다.

System.out.println();에서 소괄호 안에는 한개의 약속된 자료형이 와야 한다.

상위 6번 라인의 문자열 상수는 쌍따옴표로 변경해 주어야 한다. 7번 라인의 int자료형 상수는 정수의 정장 범위를 초과 하였다. int 범위를 맞추던가 더큰 수를 저장할 수 있는 long형 상수로 변환 하여야 한다. 이번 챕터를 배우고 나면 올바르게 변경할 수 있을 것이다.

다음은 올바른 상수 표기법이다. 각 자료형의 상수 표기법을 확인해 보자. 이미지에 있는 코드를 main메소드 부분에 기술한 다음 실행하면 된다.

불리언 상수

```
//아무거나 쓰면 안돼고 자바 문법만
//아무거나 쓴것이아니고 boolean상수를 쓴거다.
System.out.println(true);
System.out.println(false);
//boolean상수는 true, false 2개밖에 없기 때문에
//모든 boolean상수를 출력한결과다.
```

boolean상수를 확인해 보자. boolean자료형은 true, false값 밖에 없다. 문자열로 true, false라고 기술하면 boolean 상수 표기 법이 된다.

int형 상수

int형 상수는 int자료형 크기에 맞는 소수점이 없는 정수를 앞뒤에 아무것도 붙이지 않고 기술하면 된다.

```
System.out.println(1992);

System.out.println(30);
```

byte, short, int형 상수

byte, short, int형 자료형은 해당 데이터의 범위에 맞는 소수점이 없는 정수 형태 숫자를 그대로 기술하면 해당 자료형의 상수가 된다.

byte형이라면 -128~127, short형이라면 -32768 ~ 32767, int형 자료형이라면 -2147483648 ~ 2147483647 사이의 숫자를 앞뒤에 아무것도 붙이지 않고 그대로 기술하면 된다. 만약 값의 범위를 벗어난 수가 들어오면 에러가 발생한다.

```
System.out.println("byte:"+127);

System.out.println("short:"+32767);

System.out.println("short:"+2147483647);
```

long형 상수

long형 자료형을 만드는 방법을 확인해 보자. 소수점이 없는 숫자를 그냥 기술하면 int형 상수가 되지만 소수점이 없는 숫자 끝에 대문자 L, 소문자 l를 붙이면 long형 상수가 된다. 10과 10L은 같아 보이지만 메모리에 기록할때 크기가 다르다. 10은 int형 으로 10L은 long형 저장 방법으로 저장된다.

```
// long형 자료형
// 소문자 l또는 대문자 L를 붙이면 된다.
System.out.println(10000000000L);
System.out.println(10000000000l);
```

float형 상수

```
// float형 상수
// 소수점이 있는 수끝에 F,f를 붙여서 표시한다.
System.out.println(3.14F);
System.out.println(3.14f);
System.out.println(3f);
System.out.println(3.f);
System.out.println(.14f);
```

float형 자료형 상수를 만드는 방법을 확인해 보자. 숫자의 끝에 f,F를 붙이면 된다.

왼쪽 이미지에 있는 모든 방법이 float형 상수를 표현하는 방법이다. float형을 되도록 사용하지 말고, double를 사용하자.

double형 상수

double형 자료형 상수를 만드는 방법은 소수점이 있는 수를 아무것도 붙이지 않고 사용하면 double상수가 된다.

3.14f는 float형이고 3.14는 double형 이다. float형 사용할 곳에 double형을 사용하면 문제가 발생한다.

```
System.out.println(20.145);

System.out.println(0.145);

System.out.println(.301);

System.out.println(15.);
```

숫자가 너무 크거나 소수점이 길면 범위를 초과해서 오류가 나니 주의하자.

문자형 상수

char상수 형은 원하는 문자 하나를 ‘’로 묶어서 사용한다. ‘ab’와 ‘’같이 따옴표 안에 문자 2개나 아무것도 없으면 문자 상수가 아니여서 문제가 된다.

문자열에서 “”이 가능해서 종종 ‘’도 되는걸로 착각할 수 있지만 해보면 실행하지 않는다. “”은 null이고 ‘’은 컴파일 에러이다.

다음 예제를 확인해 보자.

```
// char 문자 상수 유닛코드 1개
// 문자상수는 보통 앞뒤에 ''를 사용하여 표시한다.
// System.out.println(a);error a에 대해서 약속된 바가 없다.
System.out.println('a');
// 각 문자를 화면에 출력하려면 어떻게 해야 할까요.
System.out.println('각');
```

문자열 상수

문자열 상수는 원하는 문자열을 쌍 따옴표(“”)로 묶으면 된다.

```
System.out.println(“문자열상수”)
```

이름을 가지고 있는 상수(심볼릭 상수)

```
//문자열 상수
//3.141592653589793
System.out.println(Math.PI);
```

심볼릭 상수는 변하지 않는 값을 기호로 표시한 상수이다. 상수여서 한번 선언하면 변경이 불가능하고 값을 복잡한 데이터를 문자로 표현하고자 할때 사용 한다. 수학 시간에 PI를 생각해 보자. 3.141592….가 PI값인데 프로그램에서 PI값을 사용하려면 3.141592653589793를 파이 값으로 사용 하여야 한다면

3.141592653589793 * 3.141592653589793 + 3.141592653589793 처럼 복잡한 수식을 사용해야 하는데 심볼릭 상수를 사용하면 Math.PI * Math.PI + Math.PI와 같이 간단히 기술할 수 있다.

3.141592653589793를 수학의 파이 처럼 Math.PI라는 문자로 변경해서 사용 할수 있는데 이런 것들을 문자상수(심볼릭 상수)라고 한다.

심볼릭 상수란 프로그램에서 일정한 값을 변수이름 문자열로 정의해 사용하는 상수를 말합니다.

심볼릭 상수는 복잡한 데이터를 변수로 선언하여 복잡한 데이터 대신에 변수명으로 데이터를 표현하고자 할 때 사용한다.

자바에서 심볼릭 상수는 변수 같지만 문자열 상수로 한 번 할당되면 그 값을 변경할 수 없다.

심볼릭 상수를 선언하는 예제 코드는 다음과 같다.

```
public static final double PI = 3.14159;
```

심볼릭 상수 사용 용도는 주로 읽기 전용 값이나 프로그램 전체에서 공유해야 하는 복잡한 데이터를 간단한 이름으로 표현하는 데 사용됩니다.

아래는 자바에서 상수를 정의하고 사용하는 방법에 대한 설명입니다.

상수 정의하기:

상수는 주로 클래스 내에서 선언되며, 다음과 같은 형식을 가지게 됩니다:

```
public class MyClass {
    // final 키워드를 사용하여 상수 정의
    public static final 자료형선택 CONSTANT_NAME = 문자상수에넣을값;
}
```

public: 접근 제한자는 다른 클래스에서 접근 방식을 결정 한다.

static: 클래스 변수 선언을 의미하고 클래스 변수로 선언되면 프로그램 모든 지역에서 접근 가능하다.

final: 한 번 할당되면 값을 변경할 수 없음을 나타냅니다.

자료형선택: 상수의 데이터 타입을 나타냅니다. 예를 들어, int, double, String 등이 될 수 있습니다.

CONSTANT_NAME: 상수의 변수명에 해당하고 관례적으로 대문자와 언더스코어를 사용한 스네이크 케이스로 작성됩니다.

문자상수에넣을값: 상수에 할당되는 값입니다.

상수 사용하기:

상수는 클래스명을 통해 직접적으로 접근할 수 있습니다. Main.PI

```
public class Main {
    public static final double PI = 3.14159;
    public static void main(String[] args) {

        System.out.println("원의 넓이: " + Main.PI * 5 * 5;);
    }
}
```

다음 심볼릭상수 예제를 확인해보자.

상수 기술 위치는 main 메소드 위쪽의 클래스 안에 기술한 다음 화면에 출력하는 예제이다.

```
package com.human.ex;
public class Javastart00 {
      public static final String  IP_ADDRESS = "127.120.051.223";
      public static final String  SITE_NAME = "네이버쇼핑몰생활용품";
      public static final double  PI= 3.14;
```

```
	public static void main(String[] args) {
		System.out.println(Javastart00.IP_ADDRESS);
  //Javastart00.IP_ADDRESS= “192.168.0.1”; 변수가 아니고 상수여서 변겨이 불가능
		System.out.println(Javastart00.SITE_NAME);
		System.out.println(Javastart00.PI);
		//같은 클래스에서는 클래스이름을 생략해서 다음과 같이 사용할 수 있다.
		//생략하는 방법은 사용하지 않는 것이 좋다.
		System.out.println(IP_ADDRESS);
	}
}
```

상수들을 연산하면 순서대로 연산되는 것이 아니라 연산자 우선순위에 따라 계산이 된다.

연산자 우선순위(Operator Precedence)는 수식에서 여러 개의 연산자가 사용될 때, 어떤 연산자가 먼저 수행되는지를 결정하는 규칙입니다. 연산자 우선순위에 따라 계산의 순서가 달라질 수 있습니다.

5+6은 11이 된다.

5+5*6은 35가 된다. 더하기 보다 곱하기가 연산자 우선순위가 높다.

“문자열” + “모든 자료형” 은 문자열이 된다.

+는 연산자 우선순위에 따라 앞에서 부터 2개 씩 연산 된다.

50+“10”은 숫자+문자열이므로 문자열 “5010” 이 된다.

10+10+“10”은 앞에 2개 자료형이 연산되면 20+ “10”이 되고 이것이 다시 연산되면 “2010”이된다. “10”+10+10은 앞에 2개가 연산되면 “1010”+10 다시 2개가 연산되면 “101010”이 된다.

연습문제

1. 함수란?
2. 상수란?
3. 변수란?
4. 메소드란?
5. 객체란?
6. 심볼릭 상수란?
7. 각 자료형의 상수표현 방법을 설명하시오.
8. 우선순위를 이야기해 보자.
9. “10”+10+10, 10+10+“10”, 10+10, “10+10”, “10”+ “10” 다음 각각의 실행결과를

생각해보자.

10. 다음 요구사항에 맞는 Java 클래스를 작성하세요.

MAX_STUDENTS라는 심볼릭 상수를 정의하고, 값을 30으로 설정합니다.

SCHOOL_NAME이라는 심볼릭 상수를 정의하고, 값을 "우리학교"로 설정합니다.

main 메소드에서 두 상수를 출력하세요.

```
public class School {
    // 여기에 심볼릭 상수를 정의하세요.
    public static void main(String[] args) {
        // 여기에 상수를 출력하는 코드를 작성하세요.
    }
}
```

11. 다음 요구사항에 맞는 Java 클래스를 작성하세요.

FAVORITE_COLOR라는 심볼릭 상수를 정의하고, 값을 "파란색"으로 설정합니다.

FAVORITE_FOOD라는 심볼릭 상수를 정의하고, 값을 "피자"로 설정합니다.

main 메소드에서 두 상수를 출력하세요.

```
public class Favorites {
    // 여기에 심볼릭 상수를 정의하세요.
    public static void main(String[] args) {
        // 여기에 상수를 출력하는 코드를 작성하세요.
    }
}
```

> 03. 자료형과 연산출력 기본문법 2

1. 입력 System.out.println(100);
출력 ?

2. 입력 System.out.println(20+30);
출력 ?

3. 입력 System.out.println(“20+30”);
출력 ?

4. 입력 ?
출력 15.4

4. 입력 System.out.println(“15.4”);
출력 ? 결과는 같지만 숫자가 아닌 문자로 출력한 것이다.

5. 입력 System.out.println(“10”+10+20);
출력 ? 문자+숫자는 문자가 되어 결과가 다음과 같다.

6. 입력 System.out.println(10+10+ “20”);
출력 ? +연산은 앞에 두개씩 연산이 되어 10+10은 20이 된다., 다음에 숫자+문자 연산된 것이다.

7.입력 System.out.println(2+6*(4*2));
출력 ? 연산자 우선순위를 생각해 보고 계산 결과를 얻어 보자.

8.입력 System.out.println(“2*(4+3+2)+(4+1*2)”);
출력 ?

11.입력 System.out.println(“(5+3)”);
출력 ?

12.입력 System.out.println(“7+3=”+(7+3));
출력 ? 일부 문자열과 정수 7과 3만 이용해서 다음과 같이 출력해보자.

13.입력 System.out.println(“7+3=”+7+3);
출력 ?

다음 실행결과를 눈으로 예측해 보자.

```
System.out.println(50 + 25);

System.out.println("50 + 25");

System.out.println("50” + “25");

System.out.println(12.3);

System.out.println("12.3");

System.out.println("5" + 15 + 10);

System.out.println(5 + 15 + "10");

System.out.println(3 + 5 * (2 + 4));

System.out.println("3 * (2 + 4) = " + (3 * (2 + 4)));

System.out.println("5 + 3 = " + 5 + 3);
```

> 04. 변수

변수는 변하는 수를 의미 한다. 프로그램에서 변하는 데이터를 식별하기 위해서 사용한다.

프로그램을 진행하다 한번 정한 값을 변경 해야 하는 경우 변수를 사용 한다.

변수는 하나의 상수값을 넣을수 있고, 변수는 들어 있는 상수값과 동일하게 동작한다.

데이터가 프로그램에서 실행되려면 메모리에 저장되어야 합니다.

메모리에 저장하려면 두 가지가 필요합니다.

저장 공간 확보: 자료형을 이용하여 메모리에서 필요한 공간의 크기를 결정합니다.

저장 공간 식별: 변수명을 통해 여러 변수 중 특정 변수를 식별할 수 있습니다.

변수선언 방법

변수를 선언 하는 방법은 다음과 같이 2가지가 있다.

1번은 선언 이후에 데이터를 저장하는 방법이고,

2번은 선언과 함께 변수에 원하는 데이터를 저장하는 방법 이다.

1.자료형 변수명;	int a;
변수명 = 자료형에 해당하는 상수;	a = 10;
2. 자료형 변수명 = 자료형에 해당하는 상수;	int a = 10;

= 연산자는 할당 연산자로 오른쪽 데이터를 왼쪽 변수에 할당 할수 있다.

a=10;은 a에 10을 넣으라는 이야기이다. 실행되면 a가 10의 값을 가지게 되어 프로그램에서 a가 상수 10처럼 사용 된다.

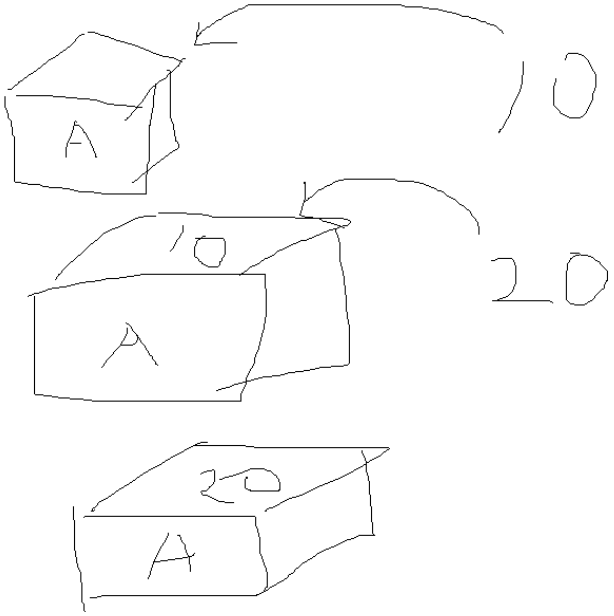


그림 설명:

위의 그림에서 A라는 변수가 있습니다.

위쪽 큐브: 초기값 10이 할당된 상태입니다.

중간 큐브: 값이 20으로 변경된 상태입니다.

아래쪽 큐브: 여전히 A라는 변수를 가리키고 있으며, 현재 값은 20입니다.

변수의 변화 과정:

처음에는 A에 10이 저장됩니다.

나중에 A의 값이 20으로 변경됩니다.

변수 A는 항상 가장 최근의 값을 가리킵니다.

접시 (변수)

설탕 소금 마늘 양파 등 다양한 재료가 있다고 생각해 보자. 부엌에 재료를 여기 저기 놓아 두면, 재료들이 섞이고 설탕, 소금 같은 비슷한 재료는 알아 볼 수 없게 된다. 이런 문제를 해결 하기 위해서 용기에 재료를 담고 라벨을 붙여 두면 문제가 해결 된다. 그릇과 라벨의 용도는 저장 공간 확보와 재료 구분의 용도이다. 용기는 자료형, 라벨은 변수명에 해당 한다.

변수 선언 방법은 자료형을 기술하고 다음에 변수명을 기술하면 된다. 자료형은 이전에

배운 int,double에 해당하고, 변수명은 식별 가능하게 본인 마음대로 만들면 된다.

변수명을 붙이는 데에는 약간의 규칙이 있으며 규칙에 어긋나지 않는 한 프로그래머가 원하는 형태로 기술할 수 있다. 변수 이름 짓는 방법을 간단히 살펴 보면 '소문자' 또는 '소문자+숫자형태'로 변수 이름을 사용 하자. 변수는 식별자로 사용되므로 같은 이름으로 2개 선언하면 안된다.

다음은 모든 자료형의 변수를 선언하고 상수를 할당하여 화면에 출력하는 방법이다.

코드를 작성해 보고 실행해 보자.

1.메인 메소드를 만들고 main메소드 안에 탭으로 뛰어서 기술하여 실행해 보자.

2.연산시 변수는 변수에 들어 있는 실제 값으로 인식되어 실행 된다.

3.변수를 더하면 변수가 가지고 있는 값을 가지고 연산이 된다.

a+1에서 a가 가지고 있는 값이 1이라면 실행 결과는 2가 된다.

변수의 사용

다음은 다양한 자료형을 사용하여 화면에 출력해 보는 예제이다. 연산자와 변수를 사용하여 프로그램 코드를 기술하고 실행시켜 보자. 하나의 메인에 기술을 하면 변수명이 같아 문제가 발생할 수 있으니 하나의 메인에 하나의 자료형 예제만 기술하던가 변수명을 달리해서 기술하자.

```
boolean b1 = true;

boolean b2 = false; //boolean은 연산이 되지 않음

System.out.println(b1);

System.out.println(b2);
```

```
char c1 = 'A';

char c2 = 'B';

System.out.println(c1);

System.out.println(c1 + "hello");

System.out.println(c1 + c2);
```

```
byte a1=10;
byte a2;
a2=11;
System.out.println(a1);
System.out.println(a2);
System.out.println(a1+a2);
System.out.println(a1+1);
```

```
short a1=10;
short a2;
a2=11;
System.out.println(a1);
System.out.println(a2);
System.out.println(a1+a2);
System.out.println(a1+1);
```

```
int a1=10;   int a2;
a2=11;
System.out.println(a1);
System.out.println(a2);
System.out.println(a1+a2);
System.out.println(a1+1);
```

```
long a1=10L;    long a2;    a2=11l;
System.out.println(a1);      System.out.println(a2);
System.out.println(a1+a2);   System.out.println(a1+1);
```

```
float a1=10.23f;   float a2;   a2=11.f;
 System.out.println(a1);      System.out.println(a2);
 System.out.println(a1+a2);   System.out.println(a1+1.4f);
```

```
 double a1=10.23;   double a2;   a2=11.94;
 System.out.println(a1);      System.out.println(a2);
 System.out.println(a1+a2);   System.out.println(a1+10.4);
```

문자열 같은 경우 String이라는 클래스를 사용한다.

```
 String st1= “Hello”;     String st2= “World”;
 System.out.println(st1);
 System.out.println(st1+ “world);
 System.out.println(st1+ st2);
 st1= “new Word”;        String str3=st1+st2;
 System.out.println(str3);
```

변수 사용법을 확인해 보았다. 주의 해야할 점은 float f=10.4는 에러가 난다. 이유는 f는 float형이고 10.4는 double형이기 때문이다. 변수에는 같은 데이터만 넣을 수 있다. 자료형 변수와 상수의 자료형을 반드시 일치해야 한다.

다음을 읽어보고 변수를 사용하는 이유를 정리해 보자.

변수는 프로그램 내에서 지속적으로 변하는 값을 저장하고 식별할 때 사용한다. 다음과 같은 코드가 있다고 생각해보자.

```
int sum=5+6;      System.out.println(sum);
sum=6+6;          System.out.println(sum);
```

첫번째 sum 출력은 11이 출력 될 것이고 두번째 sum 출력은 12가 출력 될 것이다. sum 값이 지속적으로 변경되고 저장된다.

이전 코드의 6를 7로 변경한 다음 sum에 넣어 출력해 보자. 이전 코드의 6이란 앞부분 6를 의미하는 것인지 뒤부분 6을 의미하는 것인지 애매하다. 상황이 sum=7+6이 될수도 있고 sum=6+7이 될수도 있다.

명확함을 중시하는 컴퓨터 프로그램에서 치명적인 의사소통 문제가 될 수 있어 변하는 수를 명확하게 식별하기 위해 변수를 사용 한다. 데이터의 또 다른 이름이라 생각하면 되고 값이 계속 변하니 변수라고 생각하면 된다. 다음 코드 처럼 변수를 이용하여 변경 하였다.

int a=6; int b=6; b를 7로 변경하면 b=7; 이고 sum에 a,b변수를 저장하려면 sum=a+b;하면 되고 최종 sum에 저장된 결과 sum은 13이 된다.

변수 선언시 값을 할당 하여 사용하자.

int a; int b=a+10; //a값이 할당되지 않아서 b값이 애매하다. 변수가 초기화 되어 있지 않으면 문제가 발생할 수 있다.

연습문제

1. 저장공간 확보와 저장공간 식별을 위해 사용하는것은 각각 무엇인가?
2. 변수를 선언하는 방법2가지를 기술하시오.
3. 같은 이름의 변수를 2개 생성해서 사용하면 안되는 이유는?
4. = 연산자에 대해서 설명하시오.
5. 변수 a에 15를 넣고 기존 a값에 15를 더한 a 값을 구하는 코드를 기술하시오.

> 05. 변수 - 기본문법 3

1.입력
```
int a=10;
System.out.println(a);
```

출력
10

2.입력
```
int b=30;
System.out.println(b);
```
출력
?

3.입력
```
int b=30;
System.out.println(b+10);
```
출력
?

4.입력
```
int c=30;
System.out.println(c+c);
```
출력
?

5.입력 변수명hello에 50를 넣고 변수를 이용해서 80이 찍히도록 해보자
?
출력
80

6.입력
```
int   d=30;
System.out.println(d);
d=60;
System.out.println(d);
//변경된 변수의 이전 값은 접근할 수 있는 방법이 없다.
//프로그램 안에서 지속적인 접근이 필요 하다면 변수를 2개 선언해서 각각의 변수에 따로 저장해야 한다.
```
출력
30
60

7.입력
```
int   d=30;
```

```
int   c=60;
System.out.println(c);
System.out.println(d);
```
출력
60
30

6.입력
```
int i1=22;
int i2=11;
int i3=33;
System.out.println(i1+i2+i3)
```
출력
66

7.입력
```
?
System.out.println(world);
?
System.out.println(world+a);
```
출력
5
11

8.입력
```
int world=5;
System.out.println(world+5);
```
출력
10

9.입력
```
int world=5;
System.out.println(“world”+5);
```
출력
world5

10.입력
```
int world=5;
System.out.println(“world”+world);
```
출력
?

11.입력
```
int a=5;
```

```
int b=6;
System.out.println(“a+b=”+a+b);
```

출력
?

12.입력
```
int a=5;
int b=6;
System.out.println(“a+b=”+(a+b));
```

출력
?
13.입력
```
int a=5;
System.out.println(a);
a=3;
System.out.println(a);
```

출력
5
3

14.입력
```
int a=5;//처음 a를 선언할때는 자료형을 붙인다.
System.out.println(a);
a=6;//기존  a를 사용할때는 자료형을 붙이지 않는다.
System.out.println(a);
int b=5;
System.out.println(a+b);
```

출력
?

15.입력
```
int a=5;
a=6;
a=7;
a=8;
System.out.println(a);
```

출력
?

소수점이 있는 실수는 double사용 문자열은 String사용

16. 입력

```
int a=5;
double b=5.1;
String st= “hello”;

System.out.println(a+b+st);
System.out.println(st+b+a);
System.out.println(“a+b=”+(a+b)+st);
```

출력

?

17.입력

```
int a=5;
a=a+2;
System.out.println(a);
```

출력

7

18.입력

```
int sum=0;
sum=sum+1;
System.out.println(sum);
sum=sum+2;
System.out.println(sum);
sum=sum+3;
System.out.println(sum);
int a=5;
sum=a+5;
System.out.println(sum);
a=7;
sum=sum+a;
System.out.println(sum);
```

출력

1
3
6
10
?

> 06. 변수이름 짓기

일반적인 변수명 짓는 가장 일반적인 방법은 소문자로 시작하고, 새로운 단어 의미가 변수명에 추가 될때 마다 단어의 첫 문자만 대문자로 기술 한다.

변수의 끝에 숫자를 붙여 표현해도 된다. ex) myValue1, myValue2, isMyBag

다음 변수 명명법을 확인해 보자.

1.대소문자를 구분한다.

```
int a; int A; 다른 변수 2개를 선언한 것이다.
```

2.int double과 같은 예약어, 키워드를 사용하면 안된다.

```
int int=10; //안된다.
```

키워드는 자바 문법으로 사용하는 약속된 문자열을 의미한다. 예약어는 키워드로 예약되어 있는 문자열이다.

3.숫자로 시작해서는 안된다.

```
int 1a=10;//안된다. int a1a=20; //된다. int a1=50;된다.
```

4.특수문자는 _ $ 두 가지만 허용된다. int _a=5; //된다. 되도록 사용하지 말자.

5.이미 선언된 변수명이나 메소드명 등 을 사용할 수 없다. 식별자는 하나만 존재해야 한다.

```
int a; int a; //안된다.

int main; //main메소드가 이미 식별자로 존재해서 안된다.
```

6. 나중에 배울 메소드와 클래스, 배열 등도 변수 명명법을 따른다.

결국, 알파벳 대소문자, 소문자, 숫자로 기술하는데 반드시 알파벳으로 시작해야 한다.

관용적으로 지켜야 할 것들은 다음과 같다.

1. 변수명, 메소드는 소문자로 시작하고 변수명이 여러 단어로 구성되어 있다면 이후 나오는 단어의 첫번째 알파벳을 대문자로 기술 한다. int catAge=12;

2. 클래스는 대문자로 시작하고 변수명이 여러 단어로 구성되어 있다면 이후 나오는 단어의 첫 번째 알파벳을 대문자로 기술한다. class라는 키워드 다음에 오는 문자열이 class 이름이다. public class MyCat{}

3. 패키지는 모두 소문자를 사용해야 한다. package com.human.ex 패키지여서 모두

소문자로 기술 하였다.

4. 심볼릭상수를 선언하여 사용할 때 모두 대문자로 기술하고 새로운 단어가 나올 때마다 _로 구성한다.

MY_COLLOR, CAT_EYE, DOG_NAME,IP_ADDRESS ,SITE_NAM

다음 명명된 예제들을 확인해 보자.

오늘날씨맑음

해석: Today's weather is clear

변수명: todayWeatherClear

클래스명: TodayWeatherClear

상수명: TODAY_WEATHER_CLEAR

내친구는영리하다

해석: My friend is smart

변수명: myFriendIsSmart

클래스명: MyFriendIsSmart

상수명: MY_FRIEND_IS_SMART

집에서휴식하기

해석: Rest at home

변수명: restAtHome

클래스명: RestAtHome

상수명: REST_AT_HOME

내이름은홍길동

해석: My name is Hong Gildong

변수명: myNameIsHongGildong

클래스명: MyNameIsHongGildong

상수명: MY_NAME_IS_HONG_GILDONG

다음 예제를 변수 , 클래스, 상수 명을 만들어 보자.

강아지산책시키기　해석: Walk the dog

자바코딩연습중　해석: Practicing Java coding

주말에영화보기　해석: Watch a movie on the weekend

새로운책읽기　해석: Read a new book

아침운동하기　해석: Do morning exercise

저녁식사준비하기　해석: Prepare dinner

친구들과만남　해석: Meet with friends

학교숙제완료　해석: Complete school homework

온라인수업참가　해석: Attend online class

연습문제

1. 변수이름 지을때 유의사항 6가지를 이야기해 보자.
2. 변수 이름 지을때 관용적으로 지켜야 할 것들 4가지를 기술해 보자.
3. 다음 변수가 정상인지 잘못 되었는지 판별하고 이유를 설명 하시오.

```
int 3i=5;//x      int i3i=5;//o     int __hello=55;//o

int hello=5.3;//x       int i+j=4;//특수기호는 x       int public=5;//키워드 x

int i#2=5;//x     int MyCatAge=10;//관용적 X    int mycatage=20;//관용적 X

public class myCat(){}//관용적 X
```

4. 이름:홍길동 나이:29 키:170.1를 변수에 넣어서 화면에 출력하는 프로그램을 만들어보자.
5. 다음 자료형이 무슨 자료형 상수인지 확인해 보자.

   10, 1., 6F, .23, ‘a’

6. 다음을 실행시켜 결과를 출력해 보자. a에 5,b=10를 넣은 다음 a,b의 합을 화면에 출력해 보자.

7. 다음을 변수명,클래스명,상수명으로 만들어보자.

해석: My computer specifications

해석: Play with the cat

> 07. 변수 실습 2

정수형 변수 사용

```
int age = 25;

System.out.println("나이: " + age);
```

실수형 변수 사용

```
double height = 175.5;

System.out.println("키: " + height);
```

문자열 변수 사용

```
String name = "John";

System.out.println("이름: " + name);
```

불리언 변수 사용

```
boolean isStudent = true;

System.out.println("학생 여부: " + isStudent);
```

두 정수 합산

```
int num1 = 10;        int num2 = 20;

int sum = num1 + num2;

System.out.println("합계: " + sum);
```

정수와 실수의 곱셈

```
int quantity = 3;          double price = 9.99;

double totalPrice = quantity * price;

System.out.println("총 금액: " + totalPrice);
```

문자열 연결

```
String firstName = "Jane";       String lastName = "Doe";

String fullName = firstName + " " + lastName;

System.out.println("전체 이름: " + fullName);
```

정수의 나눗셈

```
int a = 100;          int b = 4;

int result = a / b;

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

실수의 나눗셈

```
double x = 7.5;       double y = 2.5;

double division = x / y;

System.out.println("실수 나눗셈 결과: " + division);
```

불리언 비교 연산

```
int score = 85;

boolean isPass = score >= 60;

System.out.println("합격 여부: " + isPass);
```

세 정수의 평균 계산

```
int n1 = 10;    int n2 = 20;    int n3 = 30;

int average = (n1 + n2 + n3) / 3;

System.out.println("평균: " + average);
```

변수의 초기화와 값 변경

```
int counter = 0;

System.out.println("초기값: " + counter);

counter = 5;

System.out.println("변경된 값: " + counter);
```

다중 변수 선언 및 초기화

```
int width = 100, height = 50;

System.out.println("넓이: " + width + ", 높이: " + height);
```

정수와 문자열 결합

```
int year = 2024;

String message = "현재 연도는 " + year + "입니다.";

System.out.println(message);
```

실수와 정수 혼합 연산

```
int intNum = 7;       double doubleNum = 3.5;

double result = intNum + doubleNum;

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

상수 사용

```
final double PI = 3.14159;

double radius = 5.0;

double area = PI * radius * radius;

System.out.println("원의 면적: " + area);
```

변수 간의 값 교환

```
int x = 10;     int y = 20;

int temp = x;

x = y;

y = temp;

System.out.println("x: " + x + ", y: " + y);
```

여러 타입의 변수 출력

```
int age = 30;

String city = "서울";

double salary = 5000.50;

System.out.println("나이:"+age + ", 도시: " + city + ", 연봉: " + salary);
```

> 07. 연산자 예제 실습

다음 연산자를 이용한 예제를 쳐보고 결과를 확인해 보자.

;은 코드가 끝났다는 이야기다 지면상 여러 줄로 기술 하였는데 실제 코딩시 한줄에 ;이 나오면 줄을 바꿔서 기술하자.

덧셈 (+):

```
int a = 5;  int b = 3;  int result = a + b; // 5 + 3 = 8

System.out.println(result);
```

곱셈 (*):

```
int a = 5;  int b = 3;  int result = a * b; // 5 * 3 = 15

System.out.println(result);;
```

나눗셈 (/):

```
int a = 15; int b = 3;  int result = a / b; // 15 / 3 = 5

System.out.println(result);
```

나머지 (%):

```
int a = 11; int b = 4;  int result = a % b; // 15

System.out.println(result);
```

컴퓨터 프로그래밍에서 비교 연산자는 두 개의 값을 비교하는 연산자입니다. 비교 연산자를 사용하면 조건문과 반복문에서 원하는 조건을 만족하는지 여부를 판단할 수 있습니다. 실행 결과 true, false 값을 가지는 boolean값을 가진다.

같음 (==):

```
int a = 5;  int b = 3;  boolean result = (a == b); // false

System.out.println(result);
```

같지 않음 (!=):

```
int a = 5;  int b = 3;  boolean result = (a != b); // true

System.out.println(result);
```

크거나 같음 (>=):

```
int a = 5;  int b = 3;  boolean result = (a >= b); // true
System.out.println(result);
```

작거나 같음 (<=):

```
int a = 5;  int b = 3;  boolean result = (a <= b); // false
System.out.println(result);
```

크다 (>):

```
int a = 5;  int b = 3;  boolean result = (a > b); // true
System.out.println(result);
```

작다 (<):

```
int a = 5;  int b = 3;  boolean result = (a < b); // false
System.out.println(result);
```

논리 곱(&&): 두조건이 true일때만 ture를 생성하는 연산자.

```
boolean a = true; boolean b = false;      boolean result = (a && b); // false
System.out.println(result);
boolean a = true; boolean b = true; boolean result = (a && b); // true
System.out.println(result);
```

논리 합(||): 두 조건중 하나라도 true라면 true를 생성한다.

```
boolean a = true; boolean b = false;      boolean result = (a || b); // true
System.out.println(result);
```

논리 부정(!):true를 false로 false를 true로 반전 한다.

```
boolean a = true; boolean result = !a; // false
System.out.println(result);
```

응용결과:

```
boolean a=(23 > 11); boolean b=(23 < 11); boolean result = a&&b ;
System.out.println(result);
System.out.println((23 > 11)||(23 < 11));
```

대입 연산자(=):

```
int a = 5;  int b = a;

System.out.println(a);  System.out.println(b);
```

덧셈 후 대입(+=):

```
int a = 5;  a += 3; // a = a + 3;

System.out.println(a);
```

뺄셈 후 대입(-=):

```
int a = 5;  a -= 3; // a = a - 3;

System.out.println(a);
```

곱셈 후 대입(*=):

```
int a = 5;  a *= 3; // a = a * 3;

System.out.println(a);
```

나눗셈 후 대입(/=):

```
int a = 5;  a /= 2; // a = a / 2;

System.out.println(a);
```

증가 연산자(++):변수의 값을 1씩 증가시킨다.

```
int a = 5;  a++; // a = a + 1;      System.out.println(a);
```

감소 연산자(--):변수의 값을 1씩 감소시킨다.

```
int a = 5;  a--; // a = a - 1;      System.out.println(a);
```

전위 증가 연산자(++a):

```
int a = 5;  int b = ++a; // a = a + 1, b = a;

System.out.println(a); System.out.println(b); //6, 6
```

후위 증가 연산자(a++):

```
int a = 5;  int b = a++; // b = a, a = a + 1;

System.out.println(a);  System.out.println(b); //6 5
```

> 08. 다양한 연산자 사용법

연산자란? 약속된 처리를 기호로 표시한 것을 의미 한다. 수학에 +, *, -, /가 연산자에 해당 한다. 특별한 의미를 가지는 기호를 연산자라 한다.

연산자는 다양한 종류가 있지만, 대표적으로 다음과 같은 종류가 있습니다.

1.산술 연산자: 덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%) 등의 연산을 수행하는 연산자입니다.

2.대입 연산자: 변수에 값을 할당하는 연산자입니다. 등호(=)를 기본적으로 사용합니다. 또한, 복합 대입 연산자로 +=, -=, *=, /= 등이 있습니다.

3.비교 연산자: 두 값의 크기를 비교하는 연산자입니다. >큰지, <작은지, 크거나 같음(>=), 작거나 같음(<=), 같음(==), 같지 않음(!=) 등의 연산자가 있습니다. 비교연산자의 실행 결과는 boolean값을 가진다.

4.논리 연산자: 논리적인 연산을 수행하는 연산자입니다. AND(&&), OR(||), NOT(!) 등의 연산자가 있습니다. boolean자료형 true, false를 가지고 연산하는 방법을 의미한다. 진리표를 검색해서 확인해 보자.

5.증감 연산자: 변수의 값을 1 증가시키거나 1 감소시키는 연산자입니다. 전위(++x)와 후위(x++)의 두 가지 형태가 있습니다.

6.삼항 연산자: 조건문과 비슷한 기능을 수행하는 연산자입니다. 조건식 ? 참일 때의 값 : 거짓일 때의 값 형태로 사용됩니다.

1.산술 연산자

산술연산자는 +, -, /, *, % 와 같이 수학적 계산을 수행하는 연산자입니다. 이들 모두 이항 연산자이다. 2개의 데이터를 가지고 연산한다는 의미이다.

변수명으로 변수에 들어 있는 값에 접근할 수 있다. 따라서, a=5, b=6 이라면 5+6, a+6, a+b, 5+b은 다 같은 의미이다.

2.대입 연산자

대입 연산자는 변수에 값을 넣을 때 사용하는 연산자이고 = 기호를 사용해서 표시한다. 변수에 값을 넣는다는 의미로 a=1;이라 기술하면 a라는 변수의 값이 1이 된다.

‘x=3’ 은 x에 3를 넣는다는 의미이다. x에 3을 할당 한다라 하기도 한다.

= 은 기존의 데이터를 지우고 새로운 데이터를 넣는다는 의미이다.

a=c는 c의 값을 a에 넣는다는 의미이다. 결국 a에 c를 넣으니 a, c 둘다 같아 진다는 의미이다. 오른 쪽에 있는 값을 왼쪽에 넣는다. a=b=c라 기술하면 abc모두 c값으로 바뀐다.

'x = 10; x= 5;' 2가지 문구를 순서대로 적용 한다면 x에 어떤 결과가 남을 것 인지 생각해 보자. = 은 기존 값에 새로운 값을 할당 하라는 의미이다. 따라서, x에는 마지막 값 5가 들어 있다.

초임들이 많이 하는 질문을 살펴 보면 다음과 같은 것이 있다. 이전에 x에 담겨 있는 '10'은 어디로 갔는가? x를 비우고 다시 담아서 이전 x의 값은 사라진다. 이전 x에 들어 있는 10값이 지속적으로 필요 하다면 기존 변수를 사용할 것이 아니고 기존 변수는 놔두고 새로운 변수를 생성해서 처리해야 한다.

a=5, a=6을 순서대로 실행 시키면 처음에는 a값이 5였으나 이후에 6이 들어가 이전의 5값은 접근할 수 있는 방법이 사라지고 a를 통해서 6만 접근할 수 있다. 과거의 5값은 어떻게 되느냐 사라진다고 생각해도 무방하다. 만약에 과거의 5값을 접근할 수 있는 방법이 필요 했다면 다음과 같이 다른 변수에 저장 했어야 한다.

a=5, b=6 이렇게 하면 a를 통해서는 5, b를 통해서는 6 값을 접근할 수 있다. 이렇게 변수에 저장하고 나면 a=5, 이므로 5대신에 a, 6 대신에 b를 사용할 수 있다.

기존 데이터 x의 값을 읽어와 x의 값을 변경하고 싶다면 'x=x+20' 으로 기술하면 된다.

```
int a=5; a=a+20; System.out.println(a);  //출력값은 25가 된다.
```

변수의 기존 값에 3을 더하고 싶으면 a=1, a=a+3, a 출력을 실행 시키면 a는 4가 된다. 지금 소수점이 없는 수를 가지고 연산을 하고 있으니 변수 a를 정수 자료형으로 선언하여 자바 프로그램을 구현하면 다음과 같다.

```
int a; a=1; a=a+3; System.out.println(a);//출력결과 4

int a=1; a=a+3; System.out.println(a);//다음과 같이 줄여서 쓸 수도 있다.
```

c값에 3을 넣은 다음 c를 4로 변경하고 다시 c를 6으로 변경하고 c를 출력하면 6이 된다.

```
int c=3; c=4; c=6;  System.out.println(c);
```

a값에 6을 넣고 a의 값을 2증가 시킨 다음 a를 화면에 출력하면 a가 8이 된다.

```
int a; a=6; a=a+2; System.out.println(a);
```

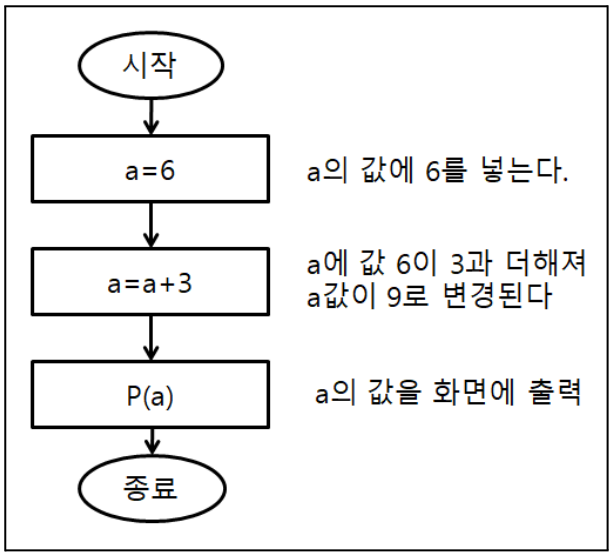
변수 a에 2가 들어 있는데 4를 더하여 a에 값을 5로 변경하려면 다음과 같이 하면 된다. 'a=2; a=a+4;' 과 같이 하면 된다.

= 연산자는 나중에 배울 ==과 해갈리면 안된다. ==은 비교연산자로 두수가 같은지 비교하여 true, false를 생성하는 연산자이다.

프로그램 코드에서는 c=c+3, b=b+2와 같은 형태의 명령어를 많이 사용하여 다음과 같이 c+=3, b+=2로 줄여 쓰는 방법을 제공한다. = 연산과 +, -, /, *, % 연산을 결합하여 줄여 쓸 수도 있다. c-=1은 c=c-1과 같다. a=a%4를 줄여서 쓰면 a%=4로 쓸수 있다. a=a+1은 a+=1 a%=b은 a=a%b등으로 줄여서 사용할 수 있다. 되도록 사용하지 않는 것이 좋다. %연산자는 나머지 연산자이다. a%b 라 기술하면 a를 b로 나눈 나머지를 의미한다. a가 7이고 b가 3이면 실행결과는 나머지 1이된다.

7=x+3;코드는 컴파일 오류가 난다. 상수는 값을 변경할 수 없어서 나는 오류이다. 7은 변경할 수 없다.

다음 순서도를 보고 자바 코드로 구현해 보자.



a값에 6을 넣고 a의 값을 3증가 시킨 다음 a를 화면에 출력하는 방법을 순서도로 작성하면 왼쪽 이미지와 같다. 자바코드로 구성해 보면 아래와 같다. 순서도에서 p는 System.out.println를 의미한다.

```
int a=6;

a=a+3;

System.out.println(a);
```

문자열 더하기 문자열은 하나의 문자열이 된다.

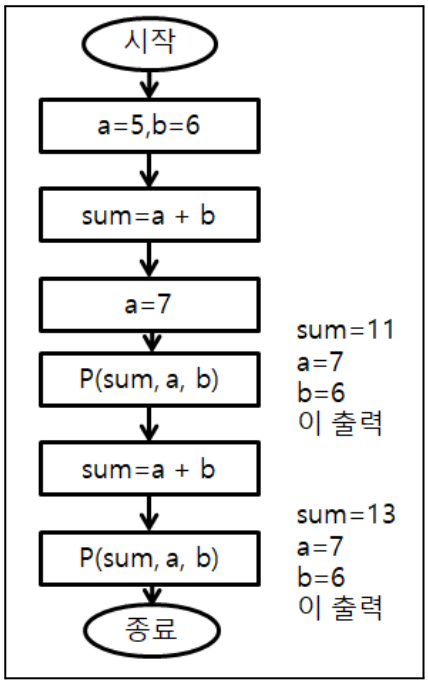
System.out.println("문자열 : "+ "안녕"); // 하나의 문자열 "문자열 : 안녕" 이 된다.

문자열에 다른 자료형을 더하면 결과는 문자열이 된다.

System.out.println("문자열"+ a);은 "문자열9"가 출력 된다.

출력 결과를 확인해 보자. "문자열"+ 10과 같은 문자열 + 숫자 연산은 "문자열10"과 같은 문자열이 된다는 것을 확인 하자. "문자열"+ 모든 자료형은 문자열이 된다.

다음 순서도를 보고 자바 코드로 구현 후 다음 내용을 읽어 보자.



왼쪽 순서도 처럼 a에 5를 넣고 b에 6를 넣은 다음 a, b를 더한 수를 sum에 넣고 a=7로 변경한 다음 a, b, sum를 출력하는 코드를 작성해 보자.

a는 7이 출력 될 것이고 b는 6이 출력될 것이고 sum은 a가 바뀌기 전에 b에 더해져 들어 갔기 때문에 11이 된다. 다시 sum=a+b 를 실행하면 a=7,b=6,sum은 13이 된다. 왼쪽 순서도를 확인해 보자.

상위 이미지 순서도에서 p()의 의미는 System.out.println 를 줄여 쓴것으로 소괄호 안의 내용을 화면에 출력하라는 이야기이다.

System.out.println(“잠에서 일어난다”);의 경우 화면에 “잠에서 일어난다” 를 출력하라는 의미이다. p(잠에서 일어난다,20)로 기술 하면 화면에 “잠에서 일어난다”과 20을 출력 하라는 의미이다.

```
System.out.println(“잠에서 일어난다”); System.out.println(20);
```

다음은 구현한 프로그램 코드이다. 출력 결과를 확인해 보자.

```
int a=5;

int b=6;

int sum=a+b;

a=7;

System.out.println(sum+“:”+a+“:”+b);

sum=a+b;

System.out.println(sum+“:”+a+“:”+b);
```

,를 사용해서 여러개의 식을 한줄에 사용할 수도 있다. 되도록 사용하지 말고 한줄에 하나씩 기술하자.

int a=0, b=1; 과 int a=0; int b=1; 는 같은 의미이다.

a=5; b=5; 와 a=5, b=5;는 같은 의미이다.

5=3; 과 같이 상수는 변경할 수 없는 데이터이므로 값을 넣어 변경할 수 없다.

3.비교 연산자

두 수의 관계가 참인지 거짓인지 비교하는 연산자이다. 연산 결과는 불리언 (boolean) 형태의 true 아니면 false가 된다. 조건문과 반복문의 조건식으로 사용된다.

비교 연산자는 두 수 a, b가 있을때 같은지 다른지 큰지 작은지를 판별 boolean형태로 참이면 true 거짓이면 false 값을 생성해 준다. >, <, ==, != 과 같은 기호를 사용한다.

10 > 5은 10이 5보다 크므로 실행결과 true가 생성된다.

10 < 5은 10이 5보다 크므로 실행결과 false가 생성된다.

10 != 5은 10과 5가 같지 않으므로 true가 생성된다. 다를때 true가 생성된다.

10==5은 10과 5가 같지 않으므로 false가 생성된다. 5==5 와 같이 같을때 true가 생성된다. 다음 표를 이해해 보자.

기 호	설명	예제
A > B	A가 B보다 크면 true	3>4:false 3>3:false 4>3:true
A >= B	A가 B보다 같거나 크면 true	3>=4:false 3>=3:true 4>=3:true
A < B	A가 B보다 작으면 true	3<4:true 3<3:false 4<3:false
A <= B	A가 B보다 작거나 같으면 true	3<=4:true 3<=3:true 4<=3:false
A==B	A와 B가 같으면 true	3==4:false 3==3:true
A!=B	A와 B가 다르면 true	3!=4:true 3!=3:false

4.논리 연산자

논리연산자는 boolean자료형을 가지고 연산해서 새로운 boolean자료형으로 만드는 연산이다.

AND(&&) 연결된 수식이 모두 true일때 true를 생성한다. 연결된 수식중 하나라도 false가 존재하면 false가 생성된다.

OR(||) 연결된 수식중 하나라도 true가 있으면 true를 생성한다. 연결된 수식이 모두 false일 때만 false를 생성한다.

NOT(!) 불리언 값을 반전 할때 사용한다. true는 false, false는 true를 생성한다.

다음은 연결결과를 표로 보여준것이다 확인해 보자.

a	b	a&&b	a||b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

1. AND(&&)

```
int x = 5;  int y = 10;

boolean result = (x > 0) && (y > 0);

System.out.println(result);  // 출력: true
```

2.

```
boolean isSunny = true;  boolean hasUmbrella = true;

boolean goingOutside = isSunny && hasUmbrella;

System.out.println(goingOutside);  // 출력: true
```

3.

```
boolean isRaining = true;

boolean hasUmbrella = true;

boolean isWorkday = false;

boolean goOutside = isRaining && hasUmbrella && isWorkday;

System.out.println("Can I go outside? " + goOutside);
```

1. OR(||)

```
int age = 25;     boolean hasLicense = false;
boolean canDrive = (age >= 18) || hasLicense;
System.out.println(canDrive);  // 출력: true
```

2.

```
boolean isWeekend = false;    boolean isHoliday = false;
boolean canRelax = isWeekend || isHoliday;
System.out.println(canRelax);  // 출력: false
```

3.

```
int temperature = 30;        boolean isSunny = true;
boolean isWeekend = true;
boolean goForSwim = (temperature > 25) || isSunny || isWeekend;
System.out.println("Can I go for a swim? " + goForSwim);
```

1. NOT(!)

```
boolean isRaining = true;     boolean notRaining = !isRaining;
System.out.println(notRaining);  // 출력: false
```

2.

```
boolean hasSubscription = false;     boolean canWatchMovie = !hasSubscription;
System.out.println(canWatchMovie);  // 출력: true
```

3.

```
boolean isRaining = true;       boolean hasUmbrella = true;
boolean isWorkday = false;
boolean goOutside = (isRaining && hasUmbrella) || !isWorkday;
System.out.println("Can I go outside? " + goOutside);
```

5.증감 연산자

특정 변수의 값을 하나 증가 하거나 감소 시킬때 사용하는 연산자다.

x++, ++x 는 x=x+1과 같다. x--,--x 는 x=x-1과 같다.

```
a=10; a++; System.out.println(a);  //11이 출력된다.

a=10; a--; System.out.println(a);  //9이 출력된다.

a=10; ++a; System.out.println(a);  //11이 출력된다.

a=10; --a; System.out.println(a);  //9이 출력된다.
```

++가 앞에 있을 경우 해당줄 이전에 이미 x 값이 하나 증가 되어 진행 되고, 뒤에 붙을 경우 다음 줄 부터 x가 하나 증가 되어 진행 된다.

다음 2 코드라인은 동일하다.

```
a=10; System.out.println(a++); System.out.println(a);  //10,11이 출력된다.

a=10; System.out.println(a);  a=a+1; System.out.println(a);  //11이 출력된다.

a=10; System.out.println(++a); System.out.println(a);  //10,11이 출력된다.

a=10;  a=a+1; System.out.println(a);  System.out.println(a);  //11이 출력된다.
```

다음 코드를 기술하고 출력 결과를 확인해 보자.

```
int a=10; int b=0; b=a++; System.out.println(a); System.out.println(b);

int a=10; int b=0; b=++a; System.out.println(a); System.out.println(b);
```

```
int a=5;
System.out.println(a++);//5
System.out.println(++a);//7
System.out.println(a);//7

System.out.println(--a);//6
System.out.println(--a);//5
System.out.println(a--);//5
System.out.println(a);//4
```

```
a=5;
System.out.println(a);//5
a=a+1;
a=a+1;
System.out.println(a);//7
System.out.println(a);//7

a=a-1;
System.out.println(a);//6
a=a-1;
System.out.println(a);//5
System.out.println(a);//5
a=a-1;
System.out.println(a);//4
```

왼쪽 코드를 쳐보고 실행해 본 다음 생각한 결과와 같은지 확인해 보고 ++, --연산자를 이해해 보자.

10++ 와 같이 기술하면 10은 상수여서 변경이 불가능 하여 컴파일 에러가 난다.

int a=5,b,c; b=--a;c=a++;를 실행 시키고 난후 a,b,c를 출력하면 5,4,4 가 출력 된다.

int a=5,b,c; b=++a;c=a--;를 실행 시키고 난후 a,b,c를 출력하면 5,6,6 이 출력 된다.

a=a++; b=a++ + a++; 와 같이 복잡한 형태는 이식성이 떨어지니 사용하지 않는 것이 좋다.

한줄에 여러번 사용하면 이식성에 문제가 발생할 수 있으니 되도록 사용하지 않는 것이 좋다.

이식성이 란? 같은 코드가 다른 언어에서도 동일하게 동작하면 이식성이 높은 것이고 다른 언어에서 같은 코드가 다르게 동작하는 경우 이식성이 낮다고 이야기 한다.

6.삼항 연산자

삼항 연산자는 다음과 같이 사용한다.

조건식? true식: false식;

상위 예에서 삼항 연산자에서 조건식 부분이 true면 true식만 남고 조건식이 false이면 false식만 남는다.

?:; 은 true ? 1 : 0;와 같이 피연산자가 3개여서 삼항 연산자라 한다. ?:;는 ?앞에 부분이 true이면 1이남고 false 이면 0이 남는다. 나중에 배울 if문과 비슷한 연산자

이다.

다음 코드 수식을 실행시켜 변수를 출력 하여 어떤 값이 출력되는지 확인해 보자.

```
a=true?1:0; //a=1

a=false?1:0;  //a=0

a=(11>6)?1:0; //a=1

int b=5;

b=(true)?3+3:3+4;

System.out.println(b); //6
```

다음 예제를 실행해 보자.

```
  int number = 15;

  String result = (number >= 0) ? "양수" : "음수";

  System.out.println(number + "은(는) " + result + "입니다.");
```

연습문제

1. =,+=,-+ 연산자를 설명해 보자.
2. ++ 전위 후위 연산자에 대해서 설명해 보자.
3. 3=2; 7=7;이 에러인이유는?
4. z=z+4;의 의미는 무엇인가?
5. a-=10; c*=10의 의미는 무엇인가?
6. ++a, a++, --a, a-- 의 의미를 설명하시오.
7. 10++이 컴파일 오류가 나는 이유를 설명하시오.
8. 삼항 연산자가 어떤것이 있으며 어떻게 사용하는지 설명하시오.
9. 6개의 비교 연산자를 설명하시오?

> 11. 형변환과 캐스팅 연산

형변환이란? 프로그램에서 특정 자료형을 다른 자료형으로 변환하는 것을 의미한다.

예를 들면 정수 데이터를 실수 데이터로 문자 데이터를 정수 데이터로 정수 데이터를 실수 데이터로 등, 여러 자료형을 다른 자료형으로 형변환 하는 것을 의미한다. int -> double

일반적으로 변수에 데이터를 넣을때 변수의 자료형과 같은 자료형만 넣을 수 있다.

```
int i =10; float f=10.4f; double d=10.9;
```

변수에 다른 자료형을 넣고 싶다면 형변환을 통해서 넣어야 한다.

형변환에는 자동형변환(암시적 형변환)과 강제형변환(명시적 형변환)이 있다. 자동형변환은 프로그램 내에서 자동으로 형변환되고 강제 형변환은 개발자가 원하는 형태로 강제로 형변환하는 것을 의미한다.

```
int i = (int) 10.4;   // 강제 형변환
double d = 10;        // 자동 형변환
```

자동 형변환은 개발자가 다른 작업을 해주지 않아도 알아서 자료형이 변환된다.

강제 형변환은 캐스팅 연산자()와 변경 하고자하는 자료형을 데이터 앞에 기술 하면 된다.

캐스팅 연산자 사용법 : (변경후 자료형)변경전 데이터

예제 : (int)10.1 double상수가 int형으로 강제 형변환 된다.

다음과 같이 상수와 변수 앞에 사용 할 수 있다. (double)a, (double)10

수학적 연산은 같은 자료형만 연산이 가능하고 연산 결과도 같은 데이터가 생성된다.

만약 개발자가 다른 데이터를 연산 한다면, 자동으로 형변환이나 강제 형변환을 통해서 피연산자 데이터의 자료형을 같게 맞춰 주어야 한다. 형변환 이후에도 자료형이 맞춰 주지 않은 상태에서 연산을 하게 된다면 컴파일 에러가 발생한다.

```
double a=10+13.4;    double a=10.+13.4;      실수로 자동 형변환

String s= “10”+10;   String s= “10”+"10”;    문자열로 자동 형변환
```

보통 작은 크기의 자료형을 큰 크기의 자료형에 넣을 때 자동형 변환이 일어난다.

명시적 형변환을 통해서 크기가 큰 자료형을 크기가 작은 자료형에 넣을 수 있다.

명시적 형 변환(강제 형변환) 할 때는 캐스팅 연산자를 사용하여 기술 해야 하고 큰 데이터를 작은 공간에 강제로 넣는 작업을 진행 할 때에 데이터 손실이 발생할 수 있으니 조심해서 사용해야 한다.

```
long a1 =124;        //long형 변수에 int형 상수

double a2=24.5f;     //double형 변수에 float형 상수
```

상위 코드를 확인해 보자. 변수와 상수가 자료형이 다른데 문제없이 동작한다. 자료형에 데이터 손실이 발생하지 않는다면 강제로 형변환을 하지 않아도 자동으로 형변환을 한다. 강제로 형변환 하는것을 명시적 형변환이라 하고 자동으로 형변환 하는것을 암시적 형변환이라고 한다. 코드들은 모두 암시적 형변환이 일어나서 자료형이 달라도 문제없이 동작 한다.

자동형 변환은 보통 작은 자료형에서 큰 자료형에 들어갈때 발생 하는데 아닌 경우도 있어 그때 그때 확인해 가면서 공부하자.

다음은 자동 형변환시 고려되는 부분을 기술 하였다. 확인해 보자.

1. 자료형의 저장 범위 크기는 다음과 같다.

   int(4byte)<long(8byte)<float(4byte)<double(8byte)

   정수보다 실수가 저장범위가 더 넓다.

   같은 수체계에서는 저장 공간이 큰 수가 저장 범위가 더 넓다.

메모리에 저장할때 사용하는 단위는 다음과 같다.

기계장치에 데이터를 저장하려면 0과 1 만 가능하고 이단위를 비트라 한다.

비트 (bit): 0 또는 1의 값을 한개 저장할때 사용하는 데이터의 가장 작은 단위.

바이트 (byte): 8 비트로 구성된 데이터의 기본 단위.

킬로바이트 (KB): 1,024 바이트.

메가바이트 (MB): 1,024 킬로바이트.

기가바이트 (GB): 1,024 메가바이트.

테라바이트 (TB): 1,024 기가바이트.

2. char형은 문자를 저장하는 데이터라서 형변환에서 제외하고 생각하자.
3. int이하 자료형은 int형으로 자동형 변환 된다. 되도록 사용하지 말자.

   놀랍게도 byte + byte는 int형 이하 자료형의 연산 이어서 계산 결과 int형이 된다.

4. int는 long,float,double로 자동 형변환 된다.

   int, long, float, double의 자료형은 형변환시 작은 쪽에서 큰쪽으로는 자동 형변환되고 큰쪽에서 작은 쪽으로는 강제 형변환 해야 한다.

5. long은 float,double로 자동형 변환된다.
6. float은 double로 자동 형변환 된다.
7. boolean자료형은 몇몇 언어에서는 true면 1, false면 0으로 자동 형변환 되기도 하지만 자바에서는 boolean형의 자동 형변환 및 강제 형변환이 불가능 하다.

```
int a1 = 10000000000l;
int a2 = 12.24;
float a3 = 23.5;
```

```
int a1= (int)10000000000l;
int a2 = (int)12.24;
float a3=(float)23.5;
```

다음 강제 형변환 해야 하는 경우를 살펴보자.

다음 코드에서 10000000000l과 같이 기술하면 에러가 나는데 10000000000l이 int형 범위에 맞지 않아서 에러(Error)가 발생한 것이다. error란? 문제가 발생했다는 이야기이다.

큰범위의 자료형을 작은 범위의 자료형에 넣으려 하면 에러가 발생 한다.

데이터의 손실이 생길수 있으면 자동형변화가 발생 하지 않는다. 문제 없이 동작 하게 하려면 강제형변화를 해주어야 한다.

```
System.out.println(a1);
System.out.println(a2);
System.out.println(a3);
```

```
<terminated> JavaClass2 (1
1410065408
12
23.5
```

강제로 형변환을 하게 되면 자료형이 달라도 문제없이 변수에 넣을 수 있다.

강제 형변환된 데이터는 화면에 출력해 보면 손실이 발생 할 수 있으니 주의 해서 사용해야 한다.

보통 실수를 int형으로 형변환 하면 소수점이 사라진다. 정수형에서 실수형으로 변환 할때는 손실이 발생하지 않는다. 작은 범위에서 큰 범위로 변환 할 때는 크게 문제가 없지만 반대의 경우에는 데이터 손실이 발생 할 수 있다.

다른 자료형을 연산하면 암시적으로 둘중 큰쪽의 자료형으로 자동 형변환 된다. int<long<float<double에서 int가 표현 범위가 가장 작고 double이 표현 범위가 가장 큰 자료형이다. 따라서 다음과 같이 연산 된다.

int + float==float int+long==long int+double=double
long+float=float long+double==double float+double==double

다음과 같은 실수를 하지 않도록 조심하자. (int)11.1+11.1은 11.1이 int 11로 변환되어 int+double=double이여서 결과가 22.1이되고 (int)(11.1+11.1)은 22.2가 int형으로 변환되어 22가 된다. 캐스팅 연산자 우선순위가 높아 먼저 실행된다

다음은 다른 두가지 자료형이 연산 되는 방법이다. 기본 적으로 둘 중 더 큰 자료형으로 자동 형변환 된다는 것을 기본으로 다음과 같은 다양한 특이 사항들이 있으니 확인해 보자.

1. 같은 자료형을 연산한 결과는 같은 자료형이 나온다.

   5/2=2.5가 아니다. int/int=int가 되므로 2가 된다.

   2./2.=1이 아니다. double/double=double가 되므로 1.0이 된다.

2. 다른 자료형을 연산하면 같은 자료형으로 변환해서 연산이 된다. 연산시 기본적으로 둘 중 더 큰 자료형으로 자동형 변환 된다.
3. 자동 형변환을 통해 연산 결과를 같은 자료형으로 만들 수 없다면 에러가 난다.
   a. true+1 에러 난다.
4. boolean 자료형은 문자열 자료형을 제외하고는 연산할 수 없다.
   a. 1+true 안된다. “문자열”+true 는 “문자열true”라고 출력되다.
5. int 자료형보다 작은 정수형 자료형과 문자 자료형은 연산 결과 int형이된다. 문자열 자료형이 아니고 문자 자료형임을 주의하자.
   (byte,short,char,int) + (byte,short,char,int) == int
   a. byte+int=int, short+short==int가 된다.
   b. char+int=int로 ‘a’+1==98이 된다. a문자가 숫자형로 저장되어 있어 숫자 97를 읽어와 거기에 1를 더한 결과이다. 이전에도 이야기했지만 char형은 최신언어들에는 지원하지 않는 자료형이니 사용하지 말자.
6. long + (byte,short,char,int)==long형 자료형이 된다.
7. long + float == float, long + double == double형 자료형이 된다.
8. float + double == double형 자료형이 된다.
9. double + (문자열과 boolean를 제외한 모든 자료형)의 결과는 double이 된다.
10. float + (문자열과 boolean,double 를 제외한 모든 자료형)의 결과는 float이 된다.

11. 문자열 + 모든 자료형 연산 결과는 문자열이다.

a. “안녕” + ”하세요” 의 연산결과가 “안녕하세요”가 된다.

b. “1”+1==2가 아니고 문자열 11이 된다. 숫자가 문자로 바뀐 다음 연산되어 2가 아닌 11이 된다.

c. 1 + 1 + “2”은 연산자 우선순위가 적용되어 왼쪽부터 두개씩 계산되어 결과는 문자열 “22”가 된다.

12. 문자열 연산은 + 연산만 가능하다.

13. 문자열은 다른 자료형으로 변환하기 위해서 캐스팅 연산자를 사용할수 없다.

    a. (int) “11”; //안된다.

14. 문자열 를 다른 자료형으로 형 변환 할때 parse메소드를 사용한다.

    a. String str1=“11.11”; double a=Double.parseDouble(str1);

    b. double a=Double.valuOf(str1);

    c. int a=Integer.parseInt(“11”);

    d. int a=Integer.valuOf(“11”);

    e. boolean b=Boolean.parseBoolean(“true”);

15. 다양한 자료형을 문자열로 변경 하려면 “”+자료형 하면 된다.

    a. “”+123== “123”

    b. “”+5.3== “5.3”

연습문제

1. 형변환이란?
2. 캐스팅 연산자란?
3. 자동형변환과 명시적 형변환을 설명하시오.
4. 명시적 형변환시 주의할 점은 무엇인가?
5. 자료형변환 고려사항 6가지를 기술해서 물음표를 체우자.
    a. 정수 실수 자료형의 저장 범위크기 순으로 기술하시오.
    b. int이하 자료형은 ?형으로 자동형 변환 된다.
    c. byte + byte는 ?형이 된다.
    d. int는 l?로 자동 형변환 된다.
    e. long은 ?로 자동형 변환된다.
    f. float은 ?로 자동 형변환 된다.

g. boolean자료형은 몇몇 언어에서는 true면 ?, false면 ?으로 자동 형변환 되기도 하지만 자바에서는 boolean형의 자동 형변환 및 강제 형변환이 ? 하다.

6. 캐스팅 연산자는 산술연산자보다 우선순위가 ? .
7. (int)11.1+11.1과 (int)(11.1+11.1의 연산결과는 무엇이고 이유를 설명하시오.
8. 다른 두가지 자료형이 연산되는 방법을 기술한것이다 물음표를 채우자.
   a. 같은 자료형을 연산한 결과는 ? 자료형이 나온다.
   b. 연산시 기본적으로 둘 중 더 ? 자료형으로 자동형 변환 된다.
   c. 자동 형변환을 통해 연산 결과를 ? 자료형으로 만들 수 없다면 에러가 난다.
   d. boolean 자료형은 ? 자료형을 제외하고는 연산할 수 없다.
   e. int 자료형보다 작은 ? 자료형과 ? 자료형은 연산 결과 int형이된다.
   f. long + (byte,short,char,int)==?형 자료형이 된다.
   g. long + float == float, long + double == ?형 자료형이 된다.
   h. double + (문자열과 boolean를 제외한 모든 자료형)의 결과는 ?이 된다.
   i. float + (문자열과 boolean,double 를 제외한 모든 자료형)의 결과는 ?이 된다.
   j. 문자열 + 모든 자료형은 결과는 ?이다.
   k. ? 연산은 +연산만 가능하다.
   l. 문자열은 다른 자료형으로 변환하기 위해서 캐스팅 연산자를 사용할수 ?.
   m. 문자열 “11” 를 int형변환 할때사용하는 메소드는?
   n. 문자열를 double형변환 할때사용하는 메소드는?
   o. 문자열를 boolean형변환 할때사용하는 메소드는?
   p. 다양한 자료형을 ?로 변경 하려면 “”+자료형 하면 된다.

다음 문제를 풀어 보자.

```
boolean b=true;
int i=1;
short s=2;
char c='a';
float f=1.234;  //1

short s2=s+c;   //2
short s3=s+s;   //3
char c2=c+s;    //4
int i2=c+s;     //5
int i3=b+i;     //6
```

문제1. 다음 코드를 보고 1~6번까지 각각 어떤 문제가 있는지 확인하여 설명해 보고 문제가 발생하지 않도록 변경해 보자.

문제2. 다음 출력 결과가 무엇인지 확인해 보고 이유를 이야기해 보자.

1) 3+3+“hello” 2) 3*“hello” 3)“hello”+4+4

+는 이항연산자로 왼쪽에서 오른쪽으로 2개 항씩 연산 된다.

문제3. 잘못된 점을 설명하시오.

```
int a= “10”; double d=(double) “123.3”;
```

문제4. String name= “홍길동”; int age=5 일때 변수에 값을 넣어서 다음과 같이 출력하는 프로그램을 만들어 보자.

“이름은 홍길동 나이는 5 입니다.”

문제5. ‘a’+1의 결과는 어떤 자료형에 저장해야 되는가?

문제6. 다음 계산결과 자료형의 결과는 어떤 자료형인가?

1) 2.0+1.0/2.0 2)4.2/2+1 3)5+4/5 4)1+2/3.

문제7. 모든 자료형을 기술하고 저장 할 수 있는 데이터 종류와 크기를 설명해 보자.

> 10. 연산자 우선순위

연산자는 순서대로 계산되는 것이 아니고 우선순위에 따라 계산된다.

연산자 우선순위란? 프로그래밍에서 여러 연산자들이 함께 사용될때 먼저 계산되는 순서를 의미 한다. 연산자들은 항상 앞에서 부터 순서대로 실행되는 것이 아나라 우선 순위가 높으면 기술 순서와 관계없이 먼저 실행된다. 생각과 다른 결과가 나오면 연산자 우선순위를 검색해서 확인하는 습관을 가지자.

연산자 우선순위 기초 실행 순서 5가지는 다음과 같다.

1. 단항 연산자 > 이항연산자 > 삼항연산자 순으로 우선순위를 가진다.
2. 우선순위가 같으면 왼쪽에서 부터 순서대로 2개씩 실행된다.
3. * / % 가 - + 보다 우선순위가 높다.
4. = 연산자는 우선순위가 낮으며 오른쪽에서 왼쪽으로 실행된다.
5. () 괄호가 우선순위가 가장 높다.

1.단항 연산자 > 이항연산자 > 삼항연산자 순으로 우선순위를 가진다.

프로그램에서 사용되는 약속된 다양한 기호를 연산자라고 한다. 피연산자는 연산의 대상이 되는 데이터를 의미한다. 3 + 4 라는 수식에서 3은 피연산자 + 는 연산자라고 하며 수학 기호 및 프로그램에 왠만한 기호는 전부 연산자다.

연산시 피연산자의 갯수에 따라 단항 연산자 이항연산자 삼항연산자로 구분된다.

단항 연산자 > 이항연산자 > 삼항연산자 순으로 우선순위를 가진다.

++a나 +4 와 같이 피연산자가 1개일 경우 단항 연산자라고 하고 4*2와 같이 피 연산자가 2개인 경우는 이항 연산자라 하고, ?:;와 같이 항이 3개 이면 삼항 연산자라 한다.
피연산자 개수에 따라 단항, 이항, 삼항 연산자라 하며 단항 연산자가 우선순위가 가장 높고 삼항 연산자가 가장 낮다.

()가 소괄호가 아닌 캐스팅 용도로 사용되는 연산자는 단항 연산자로 우리가 책에서 배운것들 중에서 가장 높은 연산자 순위를 가진다.

2. 우선순위가 같으면 왼쪽에서 부터 순서대로 2개씩 실행된다.

대부분의 연산자는 이항 연산자여서 2개씩 연산되는데 5+4+6 과 같이 연속적으로 기술하면

순서대로 2개씩 왼쪽에서 오른쪽으로 연산 된다. 5+4가 계산되어 9가 되면 다음 남은 6과 연산되어 9+6이 진행되어 최종 결과 15가 된다. 우선순위가 같다면 왼쪽에서 오른쪽으로 순서대로 연산 된다.

3. * / % 가 - + 보다 우선순위가 높다.

+,-,%,/,* 연산자에서 %,/,* 가 우선순위가 +, - 보다 높아 먼저 계산 된다.

4.=연산자는 우선순위가 낮은 편이며 오른쪽에서 왼쪽으로 실행된다.

할당연산자 = 이 산술 연산자(*,-,/,+,%) 보다 우선순위가 낮다.

= 연산자의 우선순위가 + 보다 낮아 a=5+6에서 5+6이 먼저 실행된다.

```
c=5+6;      // +연산 결과를 변수c에 넣을 수도 있다.
c=a+b;      // 변수 a,b를 더한 값을 c에 넣는다.
c=c+1;      // 변수와 상수를 더해서 변수에 넣을 수도 있다.
```

int a=5+3; a=a+3; // = 연산자가 산술 연산자보다 우선순위가 낮아서 산술 연산 결과가 변수a에 들어 간다. System.out.println(a); System.out.println(5+3);

= 연산자의 경우 오른쪽에서 왼쪽으로 연산 된다.

a=b=c 처럼 왼쪽에서 오른쪽이 아닌 오른쪽에서 왼쪽으로 연산되는 연산자도 있다.

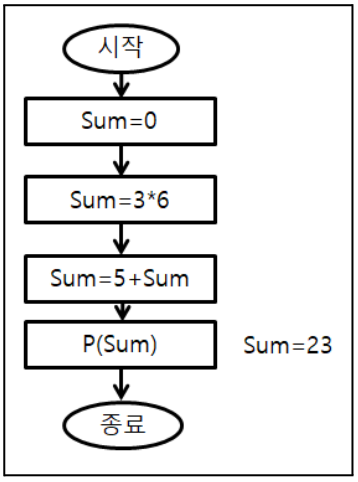
a=b=c 에서 c가 1이면 최종적으로 a, b 모두 1이 된다. 왼쪽에서 오른쪽으로 계산 된다면 a가 2라면 모두 2가 되겠지만 오른쪽에서 왼쪽으로 계산되므로 처음에 a가 2 c가 1이라면 a,b,c 모두 1이 된다. 따라서, a=1 은 올바른 수식이지만 1=a 는 잘못된 수식 이다.
상수는 변경할 수 없고 1은 상수이다. 1을 다른 값으로 변경하는 것은 불가능한 일이다.

= 연산자는 +,-,%,/,* 보다 연산자 우선순위가 낮아서 맨 나중에 실행 된다.

a=4+5 오른쪽 부분이 먼저 실행되어서 9가 된 다음에 a에 9가 들어 가 실행 결과 a는 9이다. 만약 = 연산자가 우선순위가 높아 먼저 실행된다면 a=4가 먼저 실행되어 a에 4가 들어간 다음 a+5가 실행 된다어 실행결과 a는 4가 된다. 하지만 우선순위가 = 이 더 낮기 때문에 4+5의 결과 9가 a에 들어 간다.

자바에서 기호로 되어 있는것 대부분이 연산자이고 연산자는 순서대로 실행되는 것이 아니고 우선순위에 따라 실행 된다. 간단한 예로 1+3*4는 앞에서 부터 순서대로 실행 되는 것이 아니고 3*4가 먼저 실행 된다.

다음 이미지는 한 번에 하나의 연산만 처리하는 방법으로 5+3*6를 연산하는 과정을 순서도로 만든 것이다. *이 우선 순위가 높아서 먼저 실행 되었다. 한번에 하나의 연산 결과를 변수 sum에 저장 하여 최종 연산 결과를 얻어낸 순서도이다. sum=0, sum=3*6, sum=5+sum, p(sum) 마지막에 p(sum)은 sum에 들어있는 값을 출력 하라는 이야기이다. P()에 변수나 문자열 등이 오면 들어 있는 값을 그대로 출력 하라는 의미이다. 자바 코드로 만들어보면 int sum=0; sum=3*6; sum=5+sum; System.out.println(sum); 과 같다.



5. () 괄호가 우선순위가 높다.

()괄호가 있으면 괄호가 우선순위가 높아 먼저 계산 된다.

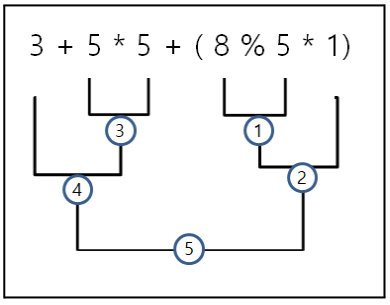
5+3*6, (5+3)*6 다음 두가지 수식을 연산하면 각각 23, 48 이다. 앞의 수식은 +,-,%,/,* 연산자에서 %,/,* 가 우선 순위가 +, - 보다 높아 먼저 계산되어 나온 결과이다. 뒤에 수식은 우선 순위가 괄호가 가장 높아서 소괄호 안의 연산이 먼저 실행 된다. *, /, %는 +, - 보다 우선순위가 높아서 +, - 보다 먼저 계산 된다. %는 나머지를 얻는 계산으로 /,* 와 우선순위가 같다.

c=a+b+1+2+3;과 c=((((a+b)+1)+2)+3)은 같다. // +는 이항 연산자이므로 연산자 우선순위에 따라 왼쪽 부터 오른쪽으로 두개씩 계산 된다.

정리해 보자면 연산자에서 중요한 2가지는 연산자 우선순위와 연산 방향이다. 계산 결과가 생각한 것과 다르게 나오면 웹에서 연산자 우선순위와 연산 방향을 확인해 보는 습관을 기르자.

다음 문제를 풀어보고 연산자 우선순위를 이해해 보자.

1. (5+3)*6의 실행 결과를 출력해 보자.



2. 왼쪽 연산을 실행해서 실행 결과를 확인해 보자.

3. 5%6+(3+3*2)의 실행 결과를 확인해 보자.

4. 2%6+3+(2+5)+3/2 의 실행 결과를 확인해 보자.

5. a=b=40; 20=c=a; a=5+3=8*4; 다음 코드중 잘못된 부분을 찾아서 잘못된 이유를 설명해 보자.

1. 연산자 우선순의 기초실행순서 5가지는 무엇인가?
2. a=b=c를 실행시키면 3개의 값이 c값으로 변경되는 이유는?

> 12. Scanner

Scanner 클래스를 이용해서 일반 사용자로 부터 문자열 입력을 받을 수 있다.

Scanner 클래스에 대해서 복잡하게 고민하지 말고 사용 방법만 이해해 보자.

일단 간단하게 Scanner라는 자료형에 변수를 만들어 변수를 통해서 원하는 작업을 한다고 생각하자.

int a; 와 java.util.Scanner scanner; 는 자료형 + 변수명 이라 생각하면 된다.

이후 코드에서 scanner 변수명 선언후 scanner를 통해서 사용자 입력 값을 받을 수 있다.

```
java.util.Scanner scanner = new java.util.Scanner(System.in);

System.out.println("문자열을 입력해주세요.");
String str1 =scanner.nextLine();
System.out.println("본인이 입력한 문자열은"+str1+"입니다.");
```

```
문자열을 입력해주세요.
안녕
본인이 입력한 문자열은안녕입
```

메인 메소드 부분에 상위 코드를 입력하고 실행하면 오른쪽 화면처럼 “문자열을 입력해주세요.” 라 출력되고 커서가 깜박이고 기다린다. 마우스로 해당 콘솔창을 선택한후 연두색글씨 “안녕”를 입력하고 엔터를 치면 프로그램이 계속 진행되어 본인이 입력한 문자열이 포함된 문자열을 str1 변수에서 받아서 이후 프로그램 안에서 콘솔화면에 str1 변수의 내용을 보여준다.

프로그램을 다시 실행한후 안녕 대신 다른 문자열을 입력하면 다른 문자열이 반영되는 것을 확인 할 수 있다. 종종 한글 입력시 문제가 발생할 수 있다. 그런 경우 영어로 입력하자. 콘솔화면이 일반적으로 사용자에게 많이 사용되고 있지 않아서 방향키를 입력해서 적절한 위치를 찾아 입력하면 입력 문제가 해결될 수 있다.

첫번째 줄 java.util.Scanner scanner =new java.util.Scanner(System.in);의 의미는 java.util.Scanner 자료형을 scanner 변수명으로 생성한 것이다. java.util.Scanner scanner 이 부분은 int a 와 비슷하게 생각하면 된다. Scanner라는 자료형을 변수 scanner로 만든 것이다. java.util.Scanner가 int이고 scanner가 a와 같다. java.util.Scanner은 반드시 기술해야 하고 scanner 대신에 sc 같은 다른 이름을 사용하여도 된다. =new java.util.Scanner(System.in) 이부분은 scanner 변수를 사용할 수 있도록 객체 생성하는 부분인데 자바 문법 공부가 끝날때 쯤 이해가 될 것이다. 그냥 scanner를 사용 할 수 있게 해주는 부분이라고 생각하고 넘어가자.

결론적으로 첫줄에서 사용자 입력을 받을 수 있도록 scanner변수를 선언 하였다고 생각하면 된다.

실제 사용자 입력을 받고 싶으면 코드에 scanner.nextLine();이라 입력하면 실행될때 이 부분에서 사용자 입력을 기다린다. 사용자 입력을 여러번 받고 싶다면 scanner.nextLine()를 여러번 기술 하면 된다.

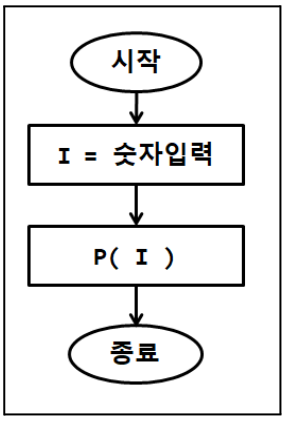
두번째 라인은 공백이고 세번째라인은 문자열을 그대로 화면에 출력 하여서 “문자열을 입력해주세요.” 라는 문자열이 콘솔창에 출력되어 있다.

네 번째 라인 String str1 =scanner.nextLine()에서 = 연산자는 오른쪽에서 왼쪽으로 실행된다. 그래서, scanner.nextLine() 부분이 먼저 실행 되는데 이부분이 사용자가 입력한 문자열을 프로그램 안으로 가져오는 역할을 한다.

실행 과정을 보여주는 콘솔창에서 사용자 입력을 기다리다가 사용자가 문자를 입력하고 엔터를 치면 입력한 문자열을 프로그램 안으로 가져온다. 그래서, 상위 프로그램 String 변수 str1에 사용자가 입력한 문자열이 들어 간다. str1변수를 통해서 사용자가 입력한 문자열을 프로그램에서 사용할 수 있다.

사용자 입력은 항상 문자열 이여서 String 에 담아야 한다. 10를 입력 하면 숫자 10이 아니고 문자열 “10” 이 입력 된다. 문자열을 다른 자료형으로 형변환 하려고 할때 문자열을 int로 형변환 하는 방법은 Integer.parseInt(“10”), 문자열로 double형으로 만드는 방법은 Double.parseDouble(“1.2”), boolean형은 Boolean.parseBoolean(“true”)을 사용하면 된다.

다음 Integer.parseInt(“10”) 처럼 “100원”이라는 문자열을 숫자로 바꾸려 하면 java.lang.NumberFormatException 에러가 나는데 문자열을 숫자로 바꿀수 없어서 나는 에러이다. 예외 처리라는 것을 해주어서 문제가 발생해도 프로그램이 중지되지 않고 동작하게 할 수 있는데 예외처리 부분은 나중에 배울 예정이다. 사용자가 약속된 형태의 데이터를 입력하지 않으면 문제가 발생한다. 일단은 사용자가 올바르게 사용한다고 생각하고 프로그램을 구현 하자.



순서도에서 사용자 입력을 받는 방법은 왼쪽 순서도 처럼 기술 한다.

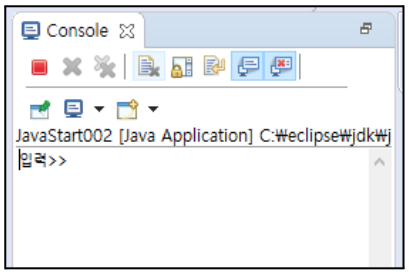
변수를 이용하여 사용자 원하는 수를 입력 받아 입력 받은 수를 출력하는 코드 이다. 다음 코드를 기술후 실행해 보자.

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

        java.util.Scanner sc=new java.util.Scanner(System.in);

        System.out.print("정수입력>>");

        String st=sc.nextLine();
```

```
        int i=Integer.parseInt(st);

        System.out.println(i);

    }
```

다음 코드 처럼 줄여 쓸수 있다.

```
    int i=Integer.parseInt(scanner.nextLine());

    System.out.println(i);
```



scanner.nextLine();이 부분에서 프로그램은 정지하고 사용자 입력을 기다 린다. 사용자가 문자열 11을 입력하면 문자열 “11”이 생성된다. 사용자가 숫자를 입력 하더라도 문자열 처리된다. 사용자 입력은 모두 문자열이다.

Integer.parseInt() 메소드는 소괄호 안에 오는 문자열을 정수로 바꿔 준다. Integer.parseInt(“11”);과 같이 사용하면 실행 결과는 정수 11이 되어 int형 변수에 넣을 수 있다. 정수가 아닌 다른 문자열을 입력하면 문제가 발생하므로 주의해서 입력 하자. 사용자의 잘못된 입력을 처리하는 방법은 나중에 예외 처리에서 공부할 것이다.

실행하면 왼쪽 처럼 console 창에 ‘입력>>’이 출력된 후 커서가 깜빡이고 있다. 이는 사용자 입력을 기다리고 있는 것이다. 마우스로 클릭한 후 본인이 원하는 아무 정수를 입력한 후 엔터를 치면 본인이 입력한 값이 출력 된다.

사용자 입력을 i에 여러번 입력 받아 화면에 출력하려면 다음과 같이 하면 된다.

```
java.util.Scanner scanner = new java.util.Scanner(System.in);
int i=0;

System.out.println(“입력>>”);  i=Integer.parseInt(scanner.nextLine());
System.out.println(i);

System.out.println(“입력>>”);  i=Integer.parseInt(scanner.nextLine());
System.out.println(i);

System.out.println(“입력>>”);  i=Integer.parseInt(scanner.nextLine());
System.out.println(i);
```

다음 코드도 확인해 보고 이후 나오는 연습문제를 풀어보자.

15,16라인에서 i1값은 10, d1값은 1.2가 될 것이고 차후 사용자가 입력한 숫자로 값이 바뀔 것이다. 17,25라인에서 i1,d1 값을 출력해 보자. 하나의 변수에 반복해서 변수값을 입력 받고 싶다면 18,19,20번을 반복 하면된다.

```
11 java.util.Scanner scanner = new java.util.Scanner(System.in);
12 int i1;
13 double d1;
14 String str1;
15 i1=Integer.parseInt("10");
16 d1=Double.parseDouble("1.2");
17
18 System.out.println("정수입력>>");
19 str1=scanner.nextLine();
20 i1 = Integer.parseInt(str1);
21
22 System.out.println("실수입력>>");
23 str1=scanner.nextLine();
24 d1 = Double.parseDouble(str1);
```

정수,실수,boolean,문자열을 사용자에게 입력받아 각 변수에 저장해서 출력하는 프로그램을 구현해 보자.

사용자가 입력한 두수를 더한 결과를 출력하는 프로그램은 다음과 같다.

```
int a=Integer.parseInt(scanner.nextLine());

int b=Integer.parseInt(scanner.nextLine());

int sum=a+b;

System.out.println(sum);
```

scanner.nextLine(); 사용자 입력을 문자열로 받아서 문자열을 생성해 준다.

scanner.nextInt(); 사용자 입력한 숫자 문자열을 int로 생성해 준다.

다음 코드를 확인해 보면 사용자 입력을 2번 받고 있다. 하지만, 코드를 입력하고 실행한 다음 콘솔창에 15를 입력 하고 엔터를 치면 사용자 입력을 1번만 받는다.

```
public static void main(String[] args) {
	java.util.Scanner sc =new java.util.Scanner(System.in);
	System.out.println("입력>>");
	int a=sc.nextInt();
	String str =sc.nextLine();
	System.out.println("종료"+a+str);
}
```

.nextInt인 경우 사용자가 숫자를 입력하고 엔터를 치면 임시 저장 공간 버퍼에 사용자 입력 문자열에서 숫자만 읽어가고 숫자가 아닌 문자열은 남겨두는 특징이 있어 .nextInt 실행후 버퍼에 숫자만 읽어가고 엔터가 남아 있다가 다음 사용자 입력 .nextLine에서

사용자로 부터 입력을 새로 받지 않고 버퍼에 남아있는 엔터 문자열을 받아 프로그램을 계속 진행해 나간다. 만약 숫자와 문자 데이터 2개 다 받아야 한다면 원하는 것과 다른 결과를 얻을 수 있다. 이런 문제가 발생하지 않도록 되도록 nextLine메소드만 사용하자.

연습문제

1. 콘솔창이란?
2. java.util.Scanner scanner =new java.util.Scanner(System.in) 의 의미는 무엇인가?
3. scanner.nextLine()의 의미는 무엇인가?
4. 프로그램에서 사용자가 입력한 데이터는 모두 어떤 자료형이 되는가?

다음 문제들을 사용자 입력을 받아서 처리 해보자. 기억나지 않는 수학공식은 웹에 검색해서 알아 보자.

5. 세로와 가로를 입력 받아 사각형의 넓이를 만드는 순서도와 프로그램을 만들어 보자.

6. 세로 가로 높이를 입력 받아 사각기둥의 부피를 만드는 순서도와 프로그램을 만들어 보자.

7. a 센치미터가 몇 미터 몇 센치인지 출력하는 프로그램을 구현 하시오. 100cm=1m

8. 잔디의 가격은 1 제곱 미터에 a원일 때 윗변이 b,아래변이 c, 높이d인 사다리꼴에 잔디를 심으려면 얼마의 비용이 드는지 구하는 순서도를 작성하시오.

9. 이름 나이 키를 입력받은 화면에 출력하는 프로그램을 작성해 보자. 출력결과: 당신의 이름은 홍길동 입니다.당신의 나이는 23입니다. 당신의 키는 165.5 입니다.

10. 두수를 입력받아 두수의 차를 출력하는 프로그램을 작성해보자.

11. 다음은 국어,영어,수학 점수를 입력 받아 평균을 구하여 다음과 같이 출력하는 프로그램을 만들어보자. 출력결과: 국어:80 영어:70 수학:90 평균:80

12. 키를 m로 소수점 2째 자리까지 입력받아 cm로 바꿔주는 프로그램을 만들어 보자.

13. 상자의 가로, 세로, 높이를 입력받아 부피를 구하는 프로그램을 만들어 보자.

14. 연필 한박스에 12자루의 연필이 들어 있고 연필 1자루는 1000원 이다. 소비자가 몇 박스와 연필 몇자루를 구매할 것인지 입력 받아 지불해야 할 돈을 계산해주는 프로그램을 만들어 보자.

15. 전체 박스 갯수, 한박스의 연필 갯수, 연필 가격을 입력 받아 전체비용를 계산해 보자.

문제풀이

7.

```
double kor=0;
double eng=0;
double math=0;
double avg=0;

java.util.Scanner sc=new java.util.Scanner(System.in);
System.out.println("국어성적입력");
kor=Double.parseDouble(sc.nextLine());
System.out.println("영어성적입력");
eng=Double.parseDouble(sc.nextLine());
System.out.println("수학성적입력");
math=Double.parseDouble(sc.nextLine());
avg=(kor+eng+math)/3.;

System.out.println(kor+":"+eng+":"+math+":"+avg);


double width=0;
double depth=0;
double height=0;
double result=0;
```

> 13. String.Format 메서드 사용법

사용 방법은 다음과 같다. 코드를 기술해서 실행 결과를 확인해 보자.

```
5  String str1= "홍길동";
6  String str2=String.format("이름 : %s", str1);
7  String str3="이름 : "+str1;
8  System.out.println(String.format("이름 : %s", str1));
9  System.out.println(str2);
10 System.out.println(str3);
```

상위와 같이 입력하고 실행해 보면 “이름 : 홍길동”과 같은 문자열이 3개 출력 된다. 8,9,10에서 1번씩 찍힌 것이다.

6번 라인을 살펴 보면 “이름 : %s”에 %s부분을 str1으로 바꿔서 문자열로 만들어 준 것이다. 실행결과는 “이름 : 홍길동” 문자열이 된다.

메서드란? 미리 만들어 놓은 코드 블럭을 호출해서 사용하는 것을 의미한다.

String.format()도 전문 프로그래머가 만들어 놓은 프로그램을 사용하는 메소드이다.

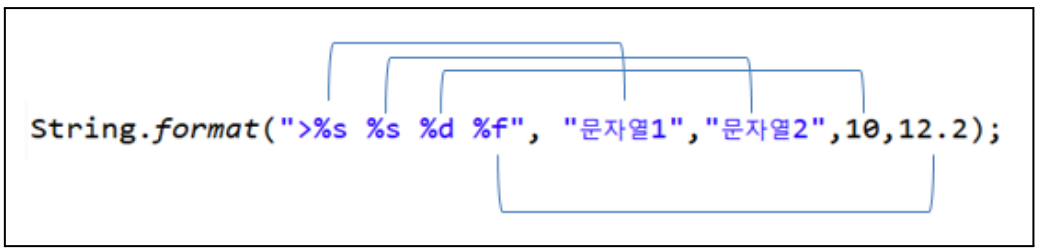
String.format()은 매개변수를 가지고 실행결과 새로운 문자열을 만들어 주는 일을 하는 메소드 이다.

매개변수는 메소드에서 필요한 기능을 실행할 때 필요한 데이터를 넘겨줄때 사용한다. 메소드에서 ()소괄호안에 , 로 구분하여 여러 데이터를 넘겨줄 수 있는데 이 데이터들을 매개변수라 한다. 메소드의 사용 방법은 메소드 이름 뒤에 소괄호를 붙이고 메소드 마다 필요한 매개 변수를 넣으면 된다.

System.out.println(“hello world”); 에서 System.out.println 은 메소드 이름이고 “hello world”은 매개 변수이다. 이것은 화면에 매개변수를 출력해 주는 메소드이다.

String.format()은 매개변수를 이용하여 새로운 문자열을 만들어주는 메소드여서 해당 메소드를 문자열 처럼 사용 할 수 있다.

다음 String.format문법을 이해해 보자.



상위 코드로 생성된 문자열은 다음과 같다. “>문자열1 문자열2 10 12.2”

```
String.format("문자: %s 숫자: %d 실수:%f … ",
                  첫번째 %문자에넣을값1, 두번째문자열에넣을값2,...)
```

첫번째 매개변수의 문자열 데이터 안에 %문자(%s,%d,%f)의 개수 만큼 다음 매개변수에 %문자를 대처할 데이터들을 순서대로 추가 하면 첫번째 매개변수의 문자열의 %문자 부분에 두번째 매개변수부터 순서대로 삽입된다.

이때 매개변수는 %s는 문자열 , %d는 정수형 %f는 실수형 자료형과 짝을 이루어야 한다.

문자열1에 %s, %d, %f가 10개 존재하면 문자열에 넣을 매개변수 값도 ,로 구분하여 10개 넣는다. 처음 나온 %부터 순서대로 매개변수가 들어간다. 매개변수의 자료형과 개수가 맞지 않으면 문제가 발생 한다.

String.format 메소드는 사용 예제들을 확인해 보자.

```
String name = "Alice";

int age = 25;

String formattedString = String.format("이름: %s, 나이: %d", name, age);

System.out.println(formattedString);
```

%s는 문자열을 삽입할 때, %d는 정수를 삽입할 때 사용됩니다.

소수점 조정을 하고 싶다면, %f 형식을 사용하여 원하는 소수점 자리수를 지정할 수 있습니다.

```
double price = 1234.56789;

String formattedPrice = String.format("가격: %.2f원", price);

System.out.println(formattedPrice);
```

%.2f는 소수점 이하 2자리까지 표현하도록 지정합니다.

또한, 출력 문자열의 너비를 설정하고 좌우 정렬도 가능합니다.

```
String leftAligned = String.format("%-10s: %d", "ID", 123);
```

```
String rightAligned = String.format("%10s: %d", "ID", 123);

System.out.println(leftAligned);

System.out.println(rightAligned);
```

%-10s는 왼쪽 정렬을 하고, 총 10자리 너비로 출력합니다. 반대로 %10s는 오른쪽 정렬을 하며, 총 10자리 너비로 출력합니다.

다음 코드는 String.format메소드를 이용하여 다양한 형태의 문자열을 만드는 코드이다. 입력한 다음 실행해 보자.

```
str1=String.format("안녕 내이름은 %s이고 사는곳은 %s"
        , "홍길동","대전");
System.out.println(str1);
str1=String.format("안녕 내이름은 %s이고 사는곳은 %s 다니는"+
        "학교는 %s 입니다.", "홍길동","대전","휴먼");
System.out.println(str1);

str1=String.format("나이는 %d", 20);
System.out.println(str1);
//%s는 문자열
//%d는 정수
//%f는 실수
str1=String.format("나이는 %d, 키는 %f", 20,170.3);
System.out.println(str1);
```

다음 코드는 %f출력시 좀더 다양한 모양으로 출력하고 싶을때 사용하는 방법이다. f앞에 +는 양수일때 +를 포함해서 출력하라는 이야기이고 10은 총10자리로 출력하라는 이야기이고 - 는 빈 공간 정렬 할 때 빈 공간을 오른쪽에 놓으라는 이야기이고 .3은 소수점 3자리까지만 출력하라는 이야기이다. 반올림되는 것을 확인할 수 있다. 아래 코드의 맨 마지막 줄은 %를 그대로 문자열로 만들고 싶을때 사용하는 것이다. %f라는 문자열을 출력하고 싶다면 아래 처럼 %%f라고 기술해야 출력할 수 있다. %%를 사용하지 않으면 %를 문자열로 출력할 수 없다. :문자열은 출력 문자열안의 공백여부를 확인하기 위해서 추가 하였다.

```
System.out.println(String.format(":%f:", 3.141592));
System.out.println(String.format(":%+f:", 3.141592));
System.out.println(String.format(":%10f:", 3.141592));
System.out.println(String.format(":%10.3f:", 3.141592));
System.out.println(String.format(":%-10.3f:", 3.141592));
System.out.println(String.format(":%%f:", 3.141592));
```

```
:3.141592:
:+3.141592:
:  3.141592:
:     3.142:
:3.142     :
:%f:
```

string.format안에서 사용되는 방법들은 다양한 곳에서 사용되어 응용해서 쓸수 있다.

연습문제

1. 메소드와 매개변수란?
2. String.format()은 어디에 사용하는 메소드 인지 기술하고 %s,%d,%f에 매칭되는 자료형은 무엇인지 기술하시오.

3. int age=156, String name = “hong”, double height=175.3 다음 데이터와 String.format를 이용해서 문자열로 만들어 출력해 보자.

4. 사용자로부터 2개의 정수를 받아서 첫번째 정수를 두번째 정수로 나누었을때의 몫과 나머지를 계산하는 프로그램을 작성하시오. ex)실행결과: 몫은 2 이고, 나머지는 1이다.

5. 3자리 숫자를 입력하여 각자리의 숫자를 출력하시오. 힌트) %연산자와 /연산자를 이용하여 만들수 있다. 423를 100으로 나누면 몫은 4고 나머지는 23이다. 23를 10으로 나눈 몫은 2이고 나머지는 3이다. ex)423를 입력하였다면 백의 자리:4 십의 자리:2 일의 자리:3 이 출력 되도록 만들어 보자.

6. 두점을 입력 받아 두점의 거리를 구하는 프로그램을 만들어보자. 두점의 거리를 구하는 방법을 웹에서 검색해보자. 힌트) Math.sqrt(25)=5 이다. x제곱은 x*x 이다.

> 14. 지역변수와 전역 변수

```
public class Example {
    // 전역변수
    public static int globalVar = 10;

    public static void main(String[] args) {
        // 지역변수
        int localVar = 5;
        System.out.println("globalVar:" + Example.globalVar);//전역변수 10
        //같은 클래스에서는  클래스이름. 를 생략 할 수 있다.
        System.out.println("globalVar:" + globalVar);
        System.out.println("localVar:" + localVar); // 5 출력 지역변수
    }
}
```

전역변수와 지역변수를 이해해 보자.

전역변수 : 프로그램 모든 곳에서 사용할 수 있는 변수이다. 클래스안에 static를 붙여서 변수를 선언하면 된다. 보통 클래스 내부 상단에 기술한다.

지역변수 : 메소드의 중괄호 안에 선언된 변수를 지역 변수라 한다. 해당 중괄호 블럭 안에서만 사용할 수 있다.

전역변수 사용방법

클래스 선언부와 main메소드 사이에 public static 를 붙여서 선언한 변수가 전역 변수이다.

지역변수 사용방법

메소드 안에 선언된 변수가 지역 변수인데 메소드에 대해서 배우지 않았는데 관련 있는 코드 블럭을 묶어서 표현한 것이라 생각하면 된다. 함수와 메소드는 비슷한 의미이다.

대표적인 메소드로는 public static void main(...){...}이 있고 중괄호안에 기술한 변수는 모두 지역 변수라고 생각하면 된다. 지금 까지 선언하여 사용한 변수들이 모두 지역 변수라 생각하면 된다.

전역변수는 클래스이름.변수명으로 접근 할 수 있다. 지역변수는 이전 처럼 변수명으로 접근 할 수 있다. 같은 클래스안에서는 클래스이름. 를 생략할 수 있지만 생략하지 않는 것이 좋다. 일반적으로 같은 이름의 변수를 2개 선언 할 수 없지만, 전역변수와 지역변수의 이름을 같게 선언 할 수 있다. 이럴 경우 클래스이름. 를 생략하면 전역변수가 아닌

지역변수에 접근하게 되는 경우가 생기는데 이럴 경우에 클래스.를 생략하면 안된다.

```
public class Example {
    // 전역변수
    public static int globalVar = 15;

    public static void main(String[] args) {
        // 지역변수
        int localVar = 5;
        int globalVar = 10;
        System.out.println("globalVar:" + Example.globalVar); //전역변수 15
        //같은 클래스에서는  클래스이름. 를 생략 할 수 있다.
        System.out.println("globalVar:" + globalVar); //지역변수 10
        System.out.println("localVar:" + localVar); //지역변수 5
    }
}
```

다른 클래스에서 전역변수는 접근할 수 있지만 지역변수는 접근할 수 없다. 지역 변수는 선언된 중괄호 블럭 안에서만 사용할 수 있다.

다음 예제는 main이 포함되어 있는 새로운 NewExample클래스가 포함된 파일을 같은 패키지에 만들어 전역변수 Example.globalVar를 출력해 보는 예제이다.
 전역 변수 값에 접근할 수 있는 것을 확인 할 수 있다. 지역변수들은 접근할 수 있는 방법이 없다.

```
// NewExample.java
public class NewExample {
    public static void main(String[] args) {
        // Example 클래스의 전역 변수 globalVar에 접근
        System.out.println("Example 클래스의 전역 변수: " + Example.globalVar);

        // Example 클래스의 main 메서드와 독립적으로 NewExample 클래스에서는
localVar와 지역 globalVar에 접근할 수 없습니다.
        // Example 클래스의 전역 변수만 접근이 가능합니다.
    }
}
```

아래 코드를 확인해 보자. 전역변수 a 는 클래스이름.변수명으로 모든 지역에서 접근 가능하다. 지역변수 a,b,c는 특정 중괄호 지역에서만 사용할 수있다.

```
package com.human.ex;
public class JavaStart10 {
      public static int a=22;//전역변수는 모든 지역에서 사용할 수 있다.

      public static void main(String[] args) {
            //전역변수 사용
```

```
		System.out.println(JavaStart10.a);//클래스이름.변수명으로 접근
		System.out.println(a);//클래스명을 생략할 수 있지만 되도록 하지 말자.

		int a=10;//main 메소드 안에서만 사용할 수 있는 지역변수
		//if,for문 등 자바문법을 사용하다 보면 중괄호가 생기는데
		// 중괄호 안에 선언한 변수는 해당 중괄호에 지역변수여서
		// 해당 중괄호 안에서만 사용할 수 있다.

		System.out.println(a);//이름이 같으면 지역변수가 우선순위를 가진다.
		//애매함을  없에게 위해서 전역변수는 클래스 이름.변수명으로 기술한다.
		{
			int b=20;
//int b=20;를 2번 선언하는 것은 불가능하다. 메소드 기준 같은 블럭에서 같은 이름의 지역 변수를 선언 할 수없다.
//int a=30;은 main메소드 블럭에서 선언되어 있어서 해당 블럭에서 선언하면 문제가 발생한다.
a가 지역변수로 이미 선언 되어 있어서 여기에 또 선언 하면 식별이 불가능 하여 a를 다시 선언 하는 것을 허용하지 않는다
			System.out.println(a);//main의 지역변수
			System.out.println(b);//해당 블록의 지역변수
			System.out.println(JavaStart10.a);//전역변수
		}
		{
//같은 이름의 변수를 다른 블록이면 같은 이름으로 지역변수를 선언할 수있다.
			int b=30;
			System.out.println(a);//main의 지역변수
			System.out.println(b);//해당 블록의 지역변수
			System.out.println(JavaStart10.a);//전역변수
		}
		//b가 지역변수 이어서 다른 블록에서 사용할수 없다.
		//System.out.println(b);
		System.out.println(a);//지역변수 10
		System.out.println(JavaStart10.a);//전역변수 22
	}
}
```

연습 문제

1) 지역변수와 전역변수를 설명해보자.

2) 다음 코드가 잘못된 이유를 설명하시오.

```
package com.human.ex;
public class JavaStart10 {
	public static int a=22;
	public static void main(String[] args) {
		int a=10;
```

```
            {
                  int a=20;
                  System.out.println(a);
            }
      }
}
```

3) 다음 코드가 정상적인 코드 인지 아닌지 설명하고 정상적으로 고쳐보자.

```
package com.human.ex;
public class JavaStart10 {
      public static void main(String[] args) {
            {
                  int a=21;
                  System.out.println(a);
            }
            System.out.println(a);
            {
                  int a=22;
            }
            System.out.println(a);
      }
}
```

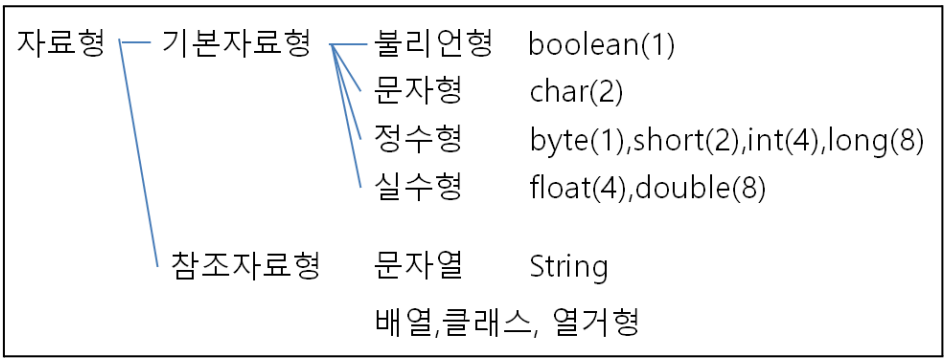
4) 다음 코드가 정상적인 코드 인지 아닌지 설명하고 정상적으로 고쳐보자.

```
package com.human.ex;
public class JavaStart10 {
      public static int a=22;//전역변수는 모든 지역에서 사용할 수 있다.
      public static void main(String[] args) {
            {
                  int a=21;
                  System.out.println(a);
            }
            System.out.println(a);
            {
                  int a=22;
                  System.out.println(a);
            }
            System.out.println(a);
      }
}
```

> 15. 기본형 자료형과 참조형 자료형

데이터 자료형은 메모리에 올라가야 프로그램에서 사용할 수 있는데 자료형 종류마다 올라가는 방식이 다르다.

다음 박스 안의 내용은 외워 보자. 소괄호 안의 숫자는 해당 자료형이 메모리에 올라 갔을 때의 메모리 크기이다. 단위는 byte이다.

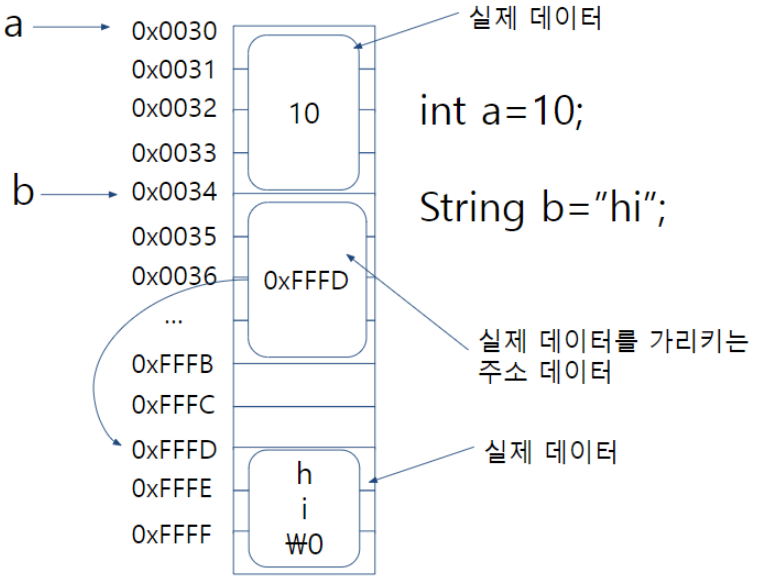


자료형의 종류는 기본 자료형과 참조자료형이 있다.

기본 자료형은 숫자, 참/거짓 등 고정 크기를 가진 자료형에 해당하고, 참조 자료형은 문자열 처럼 크기가 일정치 않은 자료형에 해당 한다.

지금까지 설명한 자료형 중 String를 제외하고 모두 기본 자료형이고 이후에 배울 다른 것들은 모두 참조 자료형이다. String, 배열, 클래스, 열거형 등이 참조 자료형에 속한다.

String은 문자열을 다루는 자료형, 배열은 같은 자료형을 한번에 여러개 생성하는 자료형, 클래스는 새로운 자료형을 만들때 사용하는 자료형, 열거형은 관련있는 상수를 묶어서 표현할때 사용하는 자료형이다.



컴퓨터의 메모리는 프로그램이 접근 할 수 있도록 0x0000 0000 부터 ~0XFFFF FFFF까지 16진수 형태의 고유 주소를 가진다. 변수명은 저 메모리 주소중 하나의 주소를 가진다.

변수명은 실제 메모리 주소를 가리킨다.

변수명 a 로 실제 저장공간에 접근하려면 쉽지만 0x0030이라는 주소로 저장 공간에 접근

하려면 a보다 어렵다. 그래서 프로그램에서는 주소 0x0030보다는 변수명 a로 접근 한다.

변수를 통해서 특정 메모리 위치를 찾아 갈수 있는데 기본 자료형은 변수가 가리키는 주소의 메모리 위치에 가면 실제 데이터가 저장되어 있고, 참조 자료형은 변수가 가리키는 주소의 메모리 위치에 가면 실제 데이터가 있는 것이 아니라 실제 데이터 위치를 가리키는 주소 데이터가 들어 있다.

기본 자료형은 실제 데이터를 다루는 것이고, 참조 자료형은 실제 데이터가 저장되 있는 주소 데이터를 다룬다. 기본 자료형은 데이터 자체를 직접 다루는 반면, 참조 자료형은 데이터가 저장된 위치를 가리키는 주소를 다루고 해당 주소에 가면 실제 주소가 있다.

보관함과 열쇠가 있다면, 보관함은 데이터를 저장할 수 있는 메모리에 해당하고, 열쇠는 데이터를 가리키는 주소 혹은 변수명에 해당 한다. 변수명이 곧 메모리 주소와 같기 때문이다.

열쇠로 보관함를 열었는데 보관함에 물품이 들어 있다면 기본 자료형 이고, 열쇠가 들어 있다면 참조 자료형이다.
참조 자료형에서 데이터를 얻으려면 열쇠로 연 보관함 안에 열쇠가 가리키는 실제 물품이 있는 보관함에 찾아가야 실제 물품을 얻을 수 있다.

다음은 이해를 돕기 위한 또다른 예제 설명이다.

목욕탕에 갔을때 입구에 신발 보관함이 있으면 신발을 벗어 보관함에 넣고 열쇠로 잠근 다음 목욕탕에 들어가 탕에 들어 가기전 옷 보관함에 옷과 신발 보관함 열쇠를 넣은 다음 사물함을 잠그고 옷 보관함 열쇠를 가지고 탕에 들어 간다. 이때, 옷 보관함 열쇠는 메모리 주소 혹은 변수에 해당하고, 열쇠를 이용해서 보관함을 열었을때 옷이 있다면 실제 데이터에 접근 하는 기본 자료형에 해당하고 열쇠가 있다면 참조 자료형에 해당 한다.

메모리는 크게 스택 메모리와 힙 메모리로 저장 영역이 구분되어 있다.

스택은 지역변수를 저장하는 메모리 공간이다.

힙 메모리는 참조 데이터의 실제 데이터를 동적으로 할당할때 사용한다.

double A =10.1; int B=53; String C= “문자열데이터”; 코드가 메인 메소드에 기술되어 있으면 각각의 데이터 (A,B,C,10.1,53, “문자열데이터”)가 어디에 저장되어 있는지 생각해 보자.

저장 위치 요약:

1. **변수 A (`double A = 10.1;`)**:
    - **저장 위치**: 스택 메모리
    - **설명**: A는 메인 메소드의 로컬 변수로, 10.1이라는 값이 A에 할당되며, 이

값은 스택 메모리에 저장됩니다.

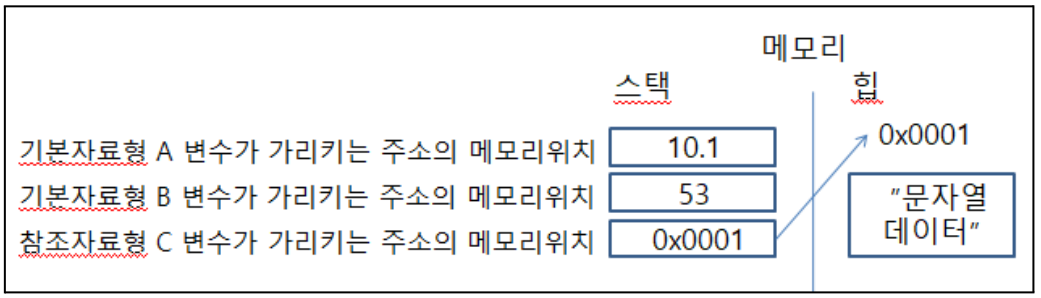
2. **변수 B (`int B = 53;`):**
    - **저장 위치**: 스택 메모리
    - **설명**: B도 메인 메소드의 로컬 변수로, 53이라는 값이 B에 할당되며, 이 값은 스택 메모리에 저장됩니다.
3. **변수 C (`String C = "문자열데이터";`):**
    - **저장 위치**: 스택 메모리
    - **설명**: C는 메인 메소드의 로컬 변수로, "문자열데이터"라는 문자열 리터럴을 가리키는 메모리 주소(참조)를 스택에 저장합니다.

값 요약:

- **A,B,C : 지역변수여서 스택에 생성된다.**
- **10.1**: 스택 메모리 (A의 초기값) **에 생성된다.**
- **53**: 스택 메모리 (B의 초기값) **에 생성된다.**
- **"문자열데이터"**: 힙 메모리의 문자열 상수 풀 **에 생성된다.**

힙 메모리의 문자열 상수 풀은 상수데이터를 저장하는 공간이라고 생각하면 된다.

코드를 간단히 그린 메모리 그림은 다음과 같다.



참조 데이터의 변수는 주소 데이터 저장 공간 이어서 열쇠 크기가 모두 동일 한것처럼 메모리 크기가 모두 4byte로 동일하다. 따라서, 참조 데이터의 변수는 고정 크기이고 실제 데이터의 크기는 가변 크기이다. 문자열 데이터를 생각해 보면 그때 그때 크기가 다르다.

실제 프로그램에서 자료형을 기본 자료형과 참조 자료형으로 분리한 이유는 속도 향상을 위해서다. 현실 세계에서 크기가 다른 물건을 정리하는 것 보다 크기가 동일한 물건을 정리하는 것이 빠르고 더 쉽다.

다음은 초밥집을 예를 들어 왜 메모리가 스택과 힙으로 나눠져 있는지 설명 하였다.

컴퓨터에 프로그램이 실행 되려면 데이터가 모두 메모리에 올라가야 하는데 지역 변수는 접근성이 용이한 스택이라는 메모리 영역에 저장하고 크기가 일정하지 않은 참조형 자료형의 실제 데이터는 힙에 저장한다.

두 종류의 자료형을 구분하는 가장 명확한 방법은 프로그램이 실행되어 메모리에 올라갔을 때 변수의 주소 위치에 실제 데이터가 들어 있는지 실제 데이터가 들어 있는 주소가 있는지에 따라 결정 된다. 즉 변수가 가리키는 곳에 갔을때 값이 들어 있으면 기본 자료형이고 주소가 들어 있으면 참조 자료형이 된다. 목욕탕 보관함 안에 물품이 들어 있으면 기본자료형, 다른 보관함의 열쇠가 들어 있다면 참조 자료형이 된다.

좀더 이해를 높이기 위해서 왼쪽 이미지를 보자.

회전 초밥집에 가보면 접시 위에 초밥이 회전 하고 있고 먹고 싶으면 손님이 직접 접시를 가져다 먹을수 있다. 이런 시스템이 존재하는 이유는 요리사가 만든 초밥을 쉽고 빠르게 손님에게 전달해 줄 수 있기 때문이다. 하지만 접시 위에 올려 놓을 수 없는 형태의 서비스를 손님에게 제공하려면 사진의 오른쪽 중간 부분 접시 위에 케이크 팻말 처럼 손님이 그 팻말을 선택 하면 주방장이 냉장고에서 케이크를 손님에게 가져다 주는 방법이다. 접시 위에 실제 케이크는 없지만 접시 위에 있는 팻말을 확인하여 손님에게 주방에 있는 케이크를 가져다 줄 수 있게 될 것이다. 여기서 접시 위에 실제 다양한 형태의 초밥이 기본 자료형에 해당하고 팻말이 올려있는 접시가 참조형 데이터에 해당한다. 기본 데이터의 경우 저장 위치에 실제 데이터가 있듯 초밥이 있고 참조 데이터에는 저장 위치에 데이터가 저장된 주소가 들어 있듯 음식 위치 정보 팻말이 접시위에 있는 것과 같은 상태 이다.

접시위에 초밥은 실제 데이터가 있는 것이고 팻말은 실제 데이터가 있는 주소가 있는 것이다. 팻말을 통해서 냉장고에는 실제 케이크를 확인해서 손님에게 가져다 줄수 있다.

이렇게 복잡 한 구조로 만든 이유는 접시에 회전 초밥을 올려 놓아야 쉽고 빠르게 초밥를 제공할 수 있는데, 접시에 부피가 큰 케이크가 있으면 요리사나 손님에게 방해가 되고 일하기 불편 하기 때문이다. 쉽고 빠르게 손님들에게 초밥을 제공하기 위해서 회전 초밥과 팻말을 사용하는 것 처럼 컴퓨터도 기본 자료형 데이터와 참조 데이터를 분리 사용해야 메모리에 올린 데이터를 빠르고 쉽게 사용할 수 있다.

기본자료형이란? 변수가 가리키는 위치에 실제 데이터가 들어 있는 자료형이다.

참조자료형이란? 변수가 가리키는 위치에 실제 데이터가 있는 주소가 들어 있는 자료형이다.

프로그램에서 모든 데이터는 메모리에 올라가야 실행 되는데, 메모리는 스택 영역, 힙 영역 등 다양한 영역으로 구분되어 있다. 이중 초밥 접시는 사람들에게 초밥을 빠르고 쉽게

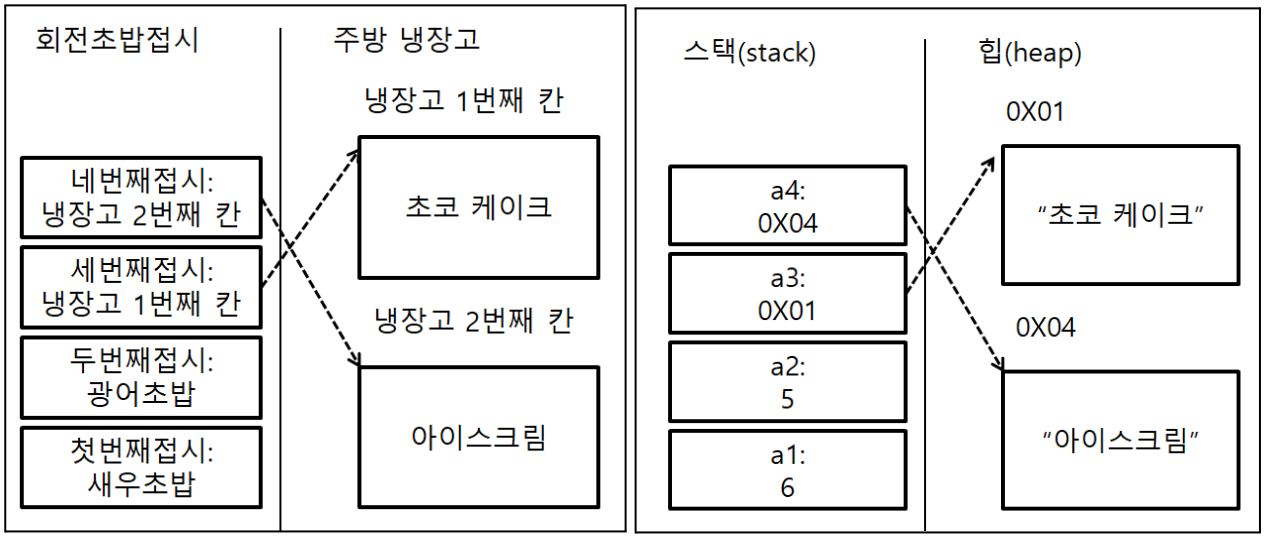
삽입,삭제 하면서 고객에게 초밥을 제공해 준다. 이는 메모리 스택 영역이 고정 크기 데이터를 프로그램에게 빠르고 쉽게 삽입,삭제 할 수 있도록 제공해 주는 것과 같다. 힙 영역은 냉장고 처럼 크기가 일정치 않은 문자열이나 클래스, 배열등의 실제 데이터를 저장하는 영역이다.

모든 메모리는 초밥 접시가 첫번째 접시 두번째 접시 … 냉장고의 첫번째 칸 두번째 칸 … 처럼 순서대로 주소가 있다. 메모리 저장공간은 16진수 형태로 0x0부터 0xffff까지 고유한 주소를 가지고 있고 모든 데이터가 적절할 주소를 찾아서 스택 힙 등의 메모리 영역에 저장된다. 저장된 데이터는 고유한 주소를 가지고 있고 변수를 통해서 해당 주소를 찾아간다. 우리가 보통 32비트 컴퓨터, 64비트 컴퓨터라고 이야기하는 것은 메모리에 접근할 수 있는 최대 주소 크기가 32비트, 64비트 라는 이야기이다. 메모리 주소는 크기가 커서 보통 16진수 표기한다.

스택(회전초밥접시)에 저장된 변수 중 기본 자료형은 해당 저장 위치에 찾아가면 실제 데이터가 있고(초밥), 참조자료형은 저장 위치에 데이터가 있는 주소(팻말)가 있다. 참조자료형의 실제 데이터는 다른 메모리 영역인 힙(냉장고)에 저장된다. 이렇게 구분해서 사용하는 이유는 성능 향상을 위해서이다. 참조 데이터는 기본 데이터와 달리 크기가 명확하지 않아 실제 데이터를 힙에 저장하고 스택에는 주소만 저장 한다.

초밥집 주방이 컴퓨터라 본다면, 초밥집 주방 초밥 접시는 메모리의 스택영역, 초밥 접시위에 초밥은 스택의 기본 자료형 데이터, 초밥 접시위에 팻말은 스택의 참조 자료형의 주소 데이터, 주방의 냉장고는 메모리의 힙 영역, 케이크는 참조 데이터의 실제 데이터에 해당 한다. 요리사는 cpu, 고객은 프로그램에 해당한다.

그동안 우리가 배운 자료형중 String만 제외하면 모두 기본 자료형이고, String를 포함해서 클래스, 인터페이스, 배열 등 이후에 배울 모든 것들은 참조형 자료형이다.



상위 오른쪽 이미지는 다음과 같은 코드를 컴퓨터에서 실행 시켰을때 메모리에 어떻게 올라가는지 그린 그림이다. 다음 코드를 확인해 보자.

int a1=6; int a2=5; String a3= “초코 케이크”; String a3= “아이스크림”;

상위 이미지는 우리가 초밥집에서 초밥을 먹을 때의 상황을 그림으로 그린 것이다. 초밥집에서 가게 문을 연다고 생각해 보자. 회전초밥 접시는 돌고 있을 것이고 그 접시 위에 초밥이 놓여 있을 것이다. 냉장고 안에는 아이스크림과 초코 케이크가 있을 것이다. 회전 초밥접시 에는 종종 초밥 대신에 아이스크림과 초코 케이크를 선택할 수 있는 팻말이 접시 위에 있을 것이다. 초밥집에서 장사를 하려면 모든 음식이 준비돼 있어야 손님에게 제공해 줄 수 있다. 초밥, 아이스크림, 케이크가 컴퓨터 프로그램에서는 프로그램 데이터에 해당한다. 음식을 미리 준비해 둬야 손님에게 음식을 팔 수 있듯 프로그램에서 데이터를 미리 준비해 놓아야 프로그램에서 데이터를 사용할 수 있다. 음식들은 회전초밥 접시나 냉장고에 준비해 놓는데 프로그램 데이터는 메모리에 준비 한다. 이때 회전초밥 접시는 메모리 중에서도 스택에 해당한다 냉장고는 메모리 중에서도 힙에 해당한다. 회전초밥 접시 위에 있는 초밥은 손님들이 회전 초밥 접시을 통해서 직접 가져 올수 있고, 푯말은 무엇이 쓰여 있느냐에 따라 확인후 냉장고에서 제공받을 수 있다.

int, double, long, float 등과 같은 기본 자료형은 실제 스택 메모리에 해당 자료형 데이터가 저장되어 있는 것 처럼 다양한 초밥도 회전초밥 접시위에 존재한다.

String 같은 참조형 데이터는 접시에 팻말이 있고 실제 데이터 아이스크림이 냉장고에 있는 것 처럼 변수를 통해 실제 데이터가 들어 있는 주소를 알 수 있고, 이 주소를 통해 힙메모리 안의 실제 데이터를 확인 할 수 있다.

초밥집에 가면 데이터를 저장하는 공간이 2개 회전 초밥접시와 주방 냉장고가 있듯이 실제 컴퓨터에 데이터를 사용하려면 메모리 스택과 힙이 있다.

if, for, 메소드, 중괄호 안에 선언된 변수들은 중괄호 블럭이 닫히기 전까지 스택에 저장되고 해당 블록이 종료되면 스택 메모리에서 사라진다. 이런 변수들을 지역변수라고 한다. “아이스크림”과 같은 참조 데이터도 중괄호 안에 선언되면 해당 변수는 지역변수 여서 스택 메모리에 생성되고 해당 변수에는 실제 데이터를 가리키는 주소가 들어가 있고, 실제 데이터는 힙에 생성된다.

식당에서 메뉴 정보는 모든 고객과 요리사들이 동일한 값으로 정보를 유지해야 한다. 자바 프로그램에서는 이들을 전역변수에 저장해서 모든 지역에서 접근할 수 있도록 해서 사용하고 있고 전역변수는 메모리 영역중 메소드 영역에 생성해서 사용한다.

연습 문제

1.메모리 영역 3가지를 기술하고 어떤데이터를 올려서 사용하는지 기술하시오.

2. 초밥집에서 다음 회전 접시, 초밥, 냉장고 , 케이크, 푯말, 주방, 요리사, 고객 각각은 프로그램에서 뭐와 비교되는지 기술하고 이유를 설명 하시오.

> 14. 메모리 영역 이해하기

자바의 메모리 영역은 메소드영역, 스택영역, 힙영역이 있다. 메모리 영역을 구분하는 이유는 처리 속력 때문이다.

메소드 영역에는 전역변수를 저장하는 메모리 영역이다.

스택 영역은 지역변수를 저장하는 메모리 영역이다.

힙 영역은 상수데이터를 저장 하는 상수 풀과 사용자가 원할때 원하는 형태로 할당하여 사용할 수 있는 메모리 영역이다.

바로 이해가 어렵더라도 외워두면 이후 설명을 이해하기 편할 것이다.

정리해 보자.

1.데이터를 사용하려면 메모리에 올라가야 한다.

2. 메모리는 3개로 분리되어 있다. (메소드,스택,힙)

3. 각 메모리 영역에 들어갈 데이터는

메소드 영역 - 실행코드,전역변수

스택 영역 - 지역변수

힙영역 - 상수풀, 크기가 정해져 있지 않은 데이터

참조형 데이터의실제 데이터

4. 자바에서 new 키워드를 이용해서 필요할때 힙에 생성하여 사용할 수 있다.

5.메모리를 분리해서 쓰는 이유는 속력향상을 의해서 이다.

메모리 영역별 특징

메소드 영역은 처음 1번 메모리에 잡히고 나면 프로그램이 종료 될 때까지 계속 유지

되어 지속적으로 접근할 수 있다.

스택 영역은 해당 메소드가 (해당 블록이) 실행 중일 때만 메모리에 등록되어 있고,

종료되면 사라진다.

힙영역 new 연산자를 이용해서 사용자가 필요할 때 생성해서 쓸수 있다. 생성된 데이터에 접근할 수 있는 방법이 없으면 메모리를 관리하는 가비지 컬렉션이 알아서 메모리에서 삭제한다. 상수는 힙영역의 상수풀에 저장한다.

String 자료형은 클래스로 참조 자료형인데 이전 코드들을 보면 new를 사용해서 생성하지 않았지만, new 연산자를 사용해서 다음과 같이 사용하는 방법도 있다.

String str2="문자열데이터"; String str=new String("문자열데이터");

복잡한 내용이 있지만 문자열은 특별이 많이 사용되니 str2="문자열데이터";과 같은 형태로 생성하는 방식을 지원 한다고 생각하고 넘어가자.

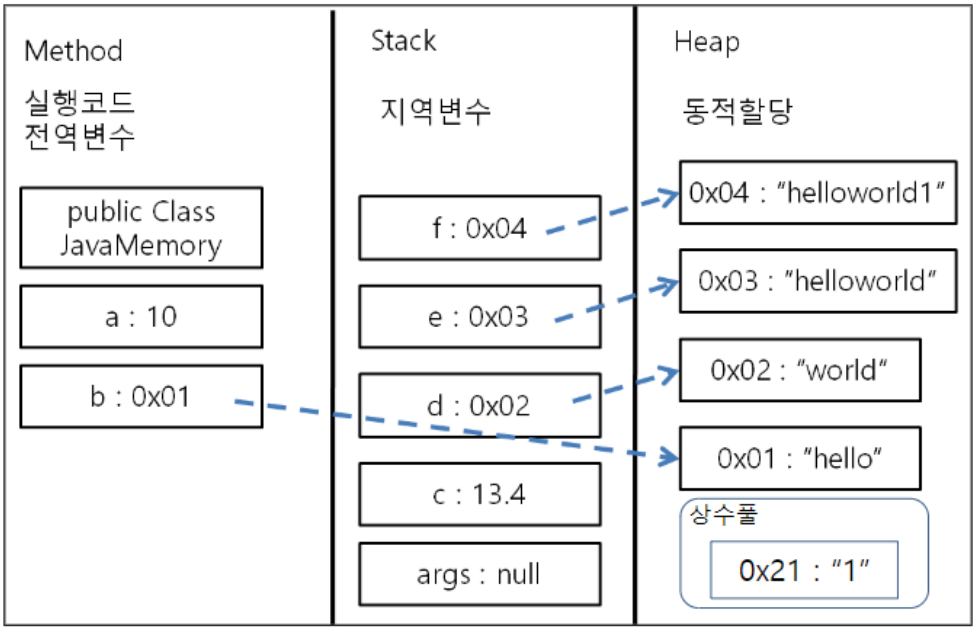
String str=new String("문자열데이터");와 같은 형태를 사용해도 동일하게 동작한다.

String 클래스는 참조 데이터에서 실제 데이터는 둘다 힙에 저장되는데 String str2="문자열데이터"; 에 문자열은 상수여서 상수풀에 new String("문자열데이터"); 은 상수풀이 아닌 메모리 힙공간에 새로 생성되어 저장된다.

String클래스는 참조 데이터여서 두가지 경우다 실제 데이터는 힙 메모리에 생성 된다.

상위 내용을 바탕으로 다음 코드를 프로그램으로 그리면 다음과 같다.

```
public class JavaMemory {
	public static int a=10;
	public static String b=new String("hello");
	public static void main(String[] args) {
		double c=13.4;
		String d=new String("world");
		//문자열은 수정이 안되고 새로 생성되서 메모힙에 생성된다.
		String e=JavaMemory.b+d;
		String f=e+"1";
	}
}
```



연습문제>

1.기본 자료형과 참조 자료형을 설명해 보자.

2.모든 자료형이 기본 자료형인지 참조 자료형인지 설명해 보자.

3.메모리를 메소드,스택, 힙으로 분리하는 이유를 설명하시오.

4.지역변수와 전역변수를 설명하시오.

5.메모리 각 영역에 들어갈 데이터를 설명하시오.

6.메모리 각 영역의 특징을 설명하시오.

> 16. 사용자 정의 자료형 클래스

참조형 데이터 클래스에 대해서 알아보자. 클래스는 관련 있는 데이터를 묶어서 새로운 형태의 자료형을 만든는 사용자 정의 자료형이다.

객체란? 현실 세계에 존재하는 모든 것을 의미한다.

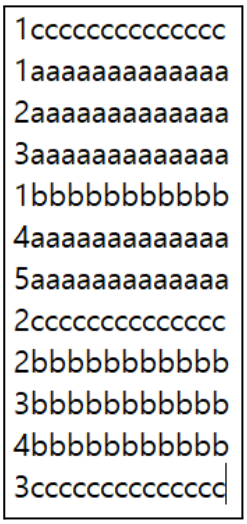
클래스란? 클래스란 **현실 세계의 객체를 프로그램 안에서 사용하기 위해한 방법** 입니다. 관련 있는 코드를 묶어서 프로그램으로 구현한 것이다. 과거에는 관련 없는 여러 프로그램을 하나의 파일로 묶어 기술하여 사용 하였는데 이런 방법은 유지 보수가 어려워 관련있는 코드를 묶어서 만드는 방법을 사용하기 시작하였다. 이때 사용하는 방법이 클래스이다. 자바 프로그램에서 객체를 구현할 때 사용하는 자바 문법이 클래스 이다.

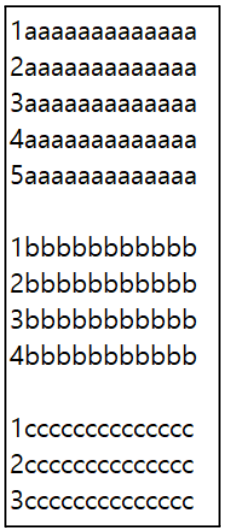
int, double 과 같은 기본 자료형으로 표현 할 수 없는 데이터를 기본 자료형들을 묶어서 새로운 형태의 자료형을 만들때 클래스를 사용한다.

사용시 일반 변수 처럼 선언하여 사용하면 되지만 차이점이 있다면 자료 크기가 명확하지 않아 new 연산을 이용해서 실제 데이터는 메모리 힙에 생성하고 선언한 변수에는 실제 데이터가 들어있는 메모리 주소를 넣는다. 클래스는 참조 데이터이다.

메모리 관리 측면에서, 객체의 실제 데이터는 힙(heap)에 할당되고, 변수는 해당 데이터에 대한 참조(실제 데이터가 있는 주소)데이터를 가지게 됩니다.

다음은 관련있는 코드를 묶어서 프로그램을 클래스로 구현하는 이유를 설명한 예이다.



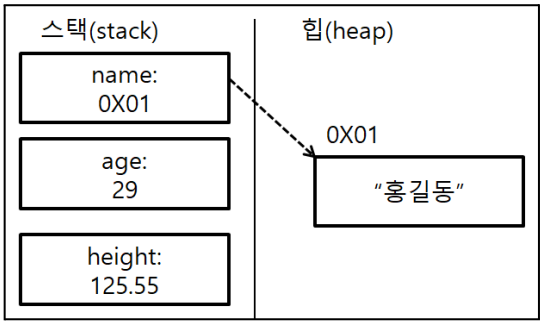


왼쪽 이미지에서 a가 들어 있는 줄은 a관련 코드이고 b가 들어 있는 코드는 b관련 코드이고 c가 들어 있는 코드는 c관련 코드라 할 때 b관련 코드를 읽어보고 수정 해야 한다면 왼쪽 보다 오른쪽 형태가 작업하기 더 쉽다. b관련 프로그램을 삭제한다면 왼쪽 보다 오른쪽 형태가 작업하기 더 쉽다.

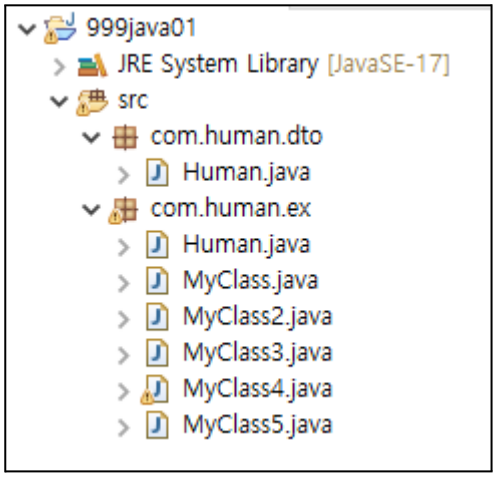
이런 이유로 객체지향 프로그램에서는 관련 있는 데이터를 클래스로 묶어서 구현한다.

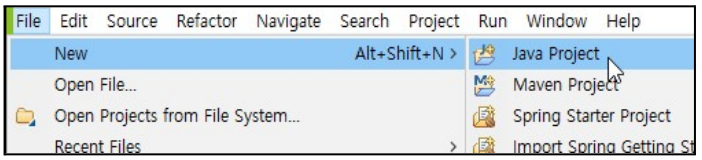
다음 메인 메소드 안에 코드들 이다. 메모리 상태를 그려 보자.

```
String name="홍길동";
int age=29;
double height=125.55;

System.out.println("name:"+name);
System.out.println("age:"+age);
System.out.println("heigt:"+height);
```



기존 사용하던 방식대로 변수를 선언하여 프로그램과 메모리를 그려보면 상위 이미지와 같다. 메인 메소드 안에 기술되어 있는 변수는 지역변수 이므로 모두 스택에 생성되고 참조 데이터 String의 실제 데이터 저장 공간은 힙에 생성 된다. 문자열 변수에는 실제 데이터가 들어있는 주소가 들어 있다.





일단 file 메뉴 >> new 메뉴 >> java Project 메뉴를 선택하여 프로젝트를 만들어 보자.
999java01로 프로젝트를 만들고 실행해 보자.

다음 코드는 일반적인 자바 코드로 사람 데이터를 작성한 것이다 한번 작성해서 실행해 보자.

MyClass.java 파일은 클래스를 사용하지 않은 현실 세계의 사람 데이터를 자료형에 저장해서 출력하는 방법이다. 확인해서 실행해 보자.

```
package com.human.ex;
public class MyClass {
	public static void main(String[] args) {
	// 사람정보를 저장해보자.
	// 사람정보는 이름 나이 키를 가지고 있다.
		String name="홍길동";
		int age = 10;
		double height=169.3;
		System.out.println(String.format(
				"이름 : %s \n나이 : %d \n 키 : %f", name,age,height));
```

//여기 까지 구현후 실행해 보고 다음을 이어나가자.

```
// 고양이한마리와 강아지 한마리에 정보를 입력받아 출력하는 프로그램을 구현해보자.
// 고양이와 강아지가 가지고 있는 데이터는 털색상, 주인이름, 본인 이름, 나이 이다.
            //고양이
            String catColor = "갈색";
            String catOwnerName = "홍길동";
            String catName = "고양이";
            int catAge = 3;
            System.out.println(String.format("털색 : %s \n주인이름 : %s
\n본인이름 : %s \n나이 : %d", catColor,catOwnerName,catName,catAge));
            //강아지
            String dogColor = "흰색";
            String dogOwnerName = "철수";
            String dogName = "강아지";
            int dogAge = 5;
            System.out.println(String.format("털색 : %s \n주인이름 : %s
\n본인이름 : %s \n나이 : %d", dogColor,dogOwnerName,dogName,dogAge));
      }
}
```

상위 코드에 집주소 address 데이터가 추가된다면 address 는 누구의 주소인가?
변수가 많아 져서 데이터가 섞이면 변수들을 사용하는데 어려움이 있고, 이를 해결하기 위해서 변수를 묶어 클래스로 만든다.

따로 따로 선언되어 있는 사람, 강아지, 고양이 관련 변수 들을 관련 있는 코드 끼리 묶어 각각의 클래스로 만들 수 있다.

클래스를 사용하는 방법은 관련있는 변수들을 묶어서 클래스로 정의한 다음 클래스 변수를 선언해서 사용하면 된다.

클래스를 정의하는 방법과 선언하는 방법은 다음과 같다.

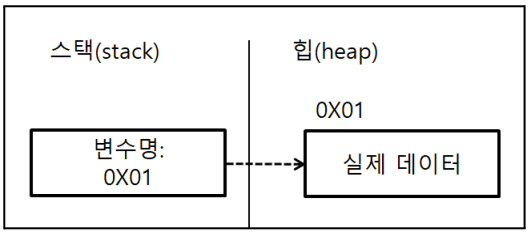
```
//1. 클래스 정의 : 사용자 정의 자료형
public class 클래스이름{//클래스이름은 결국 int와 같은 자료형 종류가 된다.
        public 기본자료형 변수이름; //클래스 안에 선언된 변수를 필드로 부른다.
        public 기본자료형 변수이름;
//본인이 만들 새로운 자료형(클래스)에 포함하고 싶은 자료형(기존 자료형)을 여러개 기술 한다.
}
//2. 클래스 변수 선언
     클래스이름 변수명=new 클래스이름();  //선언과 함께 생성
     클래스이름 변수명; 변수명=new 클래스이름();  //선언후 생성
//저장하기 원하는 객체 개수만큼 클래스 변수로 선언하여 사용 할 수 있다.
```

다음은 Human 관련 데이터를 클래스로 만들어 사용하는 예제이다.

```
1.Human 클래스 정의
public class Human{
       public String name;
       public int age;
```

```
        public double height;
}
2. 클래스 변수 선언
Human h=new Human();    Human h2; h2=new Human();
```

Human 클래스를 정의하고 Human 클래스 변수를 선언한 다음에 new 연산자를 이용해서 저장 공간을 생성한 시작 주소를 h2에 넣으면 h2변수를 통해서 저장 장소에 접근할 수 있다.



클래스 선언 변수가 지역 변수라면 왼쪽과 같이 변수는 스택에 실제 데이터는 힙 메모리에 잡힌다.

클래스 변수 선언 부분을 잘 보면 기본 자료형 선언하는 것과 비슷하다. int a; 에서 처럼 만들 클래스를 자료형으로 Human h;과 같이 기술한다.

new 연산자를 사용하면 클래스안에 선언한 변수(필드)를 저장할 수 있는 공간이 heap(힙) 메모리 영역에 잡힌 다음, 잡힌 메모리 시작 주소가 h에 저장 된다.

Human h=new Human(); 과 같이 기술하면 된다.

참조 데이터에 a=1; 처럼 값을 할당 하고 싶다면 실제 저장 공간은 new 연산자를 이용해서 힙 메모리에 생성하고 생성된 실제 데이터 주소를 변수 h 에 넣은 다음 주소가 들어 있는 h를 통해서 실제 저장된 메모리 공간에 접근할 수 있는데 . 주소 연산자를 이용해서 다음과 같이 값에 접근 하면된다. h.age=10;

new 연산자를 사용하면 실제 데이터 저장 공간이 힙에 생성되고 클래스 변수안에는 실제 생성된 데이터의 시작 주소가 저장 되는데 이때 힙에 생성된 실제 데이터를 객체 또는 인스턴스라 부른다.

인스턴스란? 클래스를 프로그램에서 사용할 수 있도록 메모리에 저장공간을 실제로 생성한 상태를 의미한다. Human h=new Human(); Human클래스가 할당되어 h에 주소가 들어간 클래스 변수를 인스턴스라 한다. Human 클래스의 인스턴스 h 라 이야기 한다. 보통 인스턴스를 객체라 부르기도 한다.

붕어빵 만들 때 붕어 빵틀로 붕어빵를 찍어 내듯이, 클래스로 원하는 데이터를 설계하고 데이터를 사용하고자 할때 인스턴스로 만들어 사용한다. 붕어빵, 클래스는 설계도에 해당하고 붕어빵, 인스턴스는 설계도로 만든 제품에 해당한다.

main 메소드안의 Human h;는 스택에 human데이터를 저장 할 변수 h를 선언한 것이어서 스택에 Human객체의 실제 데이터 주소를 저장할 공간 h 가 생긴다.

new Human()는 메모리 힙영역에 Human 데이터를 저장할 수 있는 데이터 공간을 만들고

해당 저장 공간의 주소를 생성해 준다.

h=new Human(); 라고 기술하면 new Human()에서 생성된 데이터의 시작 주소 값이 h에 들어간다.

다음은 정의한 클래스를 사용할 수 있도록 new 와 = 연산자를 사용 하여 인스턴스를 생성하는 방법을 보여주고 있다. 힙 메모리에 저장 공간을 생성하고 클래스 변수에 힙에 생성된 저장 공간 주소를 넣는 방법은 아래와 같이 new연산자와 = 연산자를 사용하면 된다.

```
//2. Human 클래스 변수 생성
int a=10;              //int 자료형 a 변수명
Human h=new Human();  //Human 은 사용자 정의 자료형 클래스 h 는 클래스 변수명
new Human();           //힙에 저장공간을 만들고 해당 주소를 생성한다.
h                      //힙에 생성된 저장 공간 주소가 들어 있는 인스턴스
h.age=10;              //h변수에 들어있는 주소의 실제데이터에 age값을 10으로 변경
```

클래스를 한번 정의하면 필요할 때 마다 클래스 인스턴스를 여러개 생성 할 수 있다. Human 정보를 저장할 수 있는 여러개의 저장 공간을 다음과 같이 생성 할 수 있다.

Human h1=new Human(); Human h2=new Human(); Human h3=new Human();

클래스 변수 h1, h2, h3를 인스턴스라고 한다. 저장 공간이 필요할때 마다 new 연산자를 사용하여 생성할 수 있다. 사람정보 1개를 저장하고 싶다면 하나의 인스턴스를 생성하면 되고 여러개 생성하고 싶다면 여러개 만들면 된다. 상위 처럼 h1,h2,h3를 생성 하였다면 사람 3명의 정보를 저장할 수 있다.

접근 방법

생성된 클래스 인스턴스의 실제 값에 접근하는 방법은 클래스 인스턴스(클래스 주소)에 . 를 찍어 인스턴스에 저장된 주소의 실제 값에 접근 할 수 있다. 이 점은 주소 연산자로 인스턴스가 가리키는 주소에 가서 원하는 데이터를 가져 오라는 의미이다.

다음은 해당 인스턴스의 이름, 나이, 키 데이터를 저장 하고 읽어오는 방법을 기술 한 것이다.

```
//3. 클래스 사용
Human h=new Human();

h.name="홍길동";    // 인스턴스에 .를 찍어서 변경하고자 하는 값에 접근할 수
h.age=14;          // 있다.
h.height=123.24;   // =연산자를 사용하여 새로운 값을 할당 할 수 있다.

System.out.println(h);  //실제 데이터가 들어있는 주소 찍기
System.out.println(h.name);  //클래스에 저장된 데이터를 읽어오는 방법이다.
```

```
System.out.println(h.age);
System.out.println(h.height);
```

기본 자료형을 사용하려면 int a=10; 이라고 선언하는 것 처럼 참조 자료형 클래스을 사용하려면 Human h = new Human();로 선언 한다. Human은 이미 존재하는 자료형이 아니여서 int 처럼 선언하여 사용 하려면, 구현부 Human 클래스를 정의한 부분이 있어야 한다.

선언한 변수를 사용하는 방법은 a를 통해서 데이터를 접근 하듯이 h를 통해서 데이터를 접근하여 사용한다.
사용 방법은 값을 변경할 때는 a=20, h.age=30 과 같이 사용 한다. 값을 읽어 올때는 a, h.age와 같이 사용한다. h.age에서 .은 주소연산자로 h에 들어있는 주소로 가서 age값을 읽어 오라는 이야기이다.

다음 MyClass2.java파일은 클래스로 관련 변수를 묶어서 표현한 것이다. 상위 이미지를 확인해서 com.human.ex에 추가해 보자.

```
package com.human.ex;
//int, double같은 내가 만든 새로운 자료형 사용자 정의 자료형 Human class
/*class Human{//외부에 Human 클래스 관련 부분을 만들때 이 클래스를 주석하자.
	public String name="홍길동";
	public int age = 10;
	public double height=169.3;
}*/
public class MyClass2 {
  public static void main(String[] args) {
	// 자료형 사용할때 int a=10;
	//class를 자료형으로 사용하려면 new연산자를 이용해서 생성한다.
	Human h = new Human();
	//클래스에 데이터가 할당된 변수를 인스턴스라한다.
	System.out.println(h);//참조자료형이여서 h에는 주소가 들어 있다.
	System.out.println(h.name);//클래스의 실제데이터는 .를 찍어서 접근
	System.out.println(h.age);//클래스 만들때의 초기 기본값이 들어 있다
	System.out.println(h.height);
	//int a=10;  a를 변경하고 싶으면 a를 변경하면 된다. a=11;
	//h.age=10;  h.age를 변경하고 싶으면 h.age를 변경하면된다. h.age=11;
	h.name="홍길남";//기존 들어있는 값을 새로운 값으로 변경한다.
	h.age=20;
	h.height=150;
	System.out.println("변경후");
	System.out.println(h.name);
	System.out.println(h.age);
	System.out.println(h.height);
//홍길동 객체를 추가로 생성해서 홍길동과 홍길남의 나이를 더한 합을 출력해보자.
//홍길남 h  홍길동 h2 의 각각의 인스턴스가 있어야 한다.
```

```
//인스턴스란 클래스의 변수에 실제 데이터를 저장할 수 있는 공간이 할당된 상태를
의미한다.
//홍길동의 정보는 기본값으로 디폴트값으로 제공되고 있다.
//새로운 변수를 선언하여 저장 공간이 2개가 되었다.
      Human h2 =new Human();//홍길동 정보
      System.out.println("두사람 나이합산:"+(h.age+h2.age));
  }
}
```

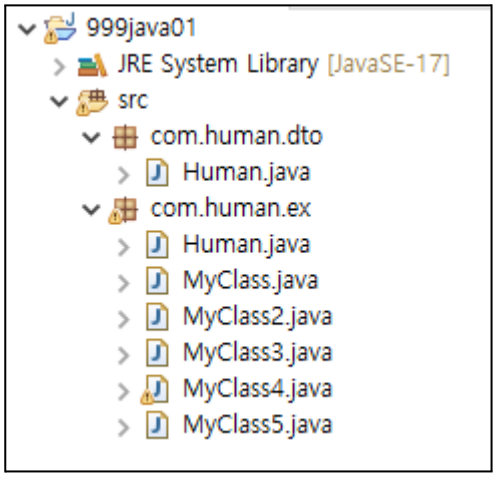
하나의 파일에는 하나의 클래스만 생성하는 것을 기본으로 한다.

여러개의 클래스를 생성 하려면 메인 메소드가 포함된 클래스만 public 를 붙이고 나머지는 public를 삭제하면 된다.

MyClass2.java 파일을 확인해 보면 Human클래스와 MyClass2 클래스가 2개 존재 한다. 한 파일에 2개의 클래스가 존재 하려면 public 클래스는 하나만 존재해야 한다. 그래서, 메인이 포함된 클래스는 public 를 붙이고 나머지는 public 키워드를 제외한 클래스로 만들어야 한다. 일반적으로 하나의 파일에는 하나의 클래스를 만드는 것이 관용적 문법이다. 교육할때는 클래스 파일마다 파일을 분리하면 만들기 귀찮아서 하나의 파일에 기술한 것이다. 하나의 파일에 기술할때는 public class가 하나만 존재해야한다. 하나의 클래스에 여러개의 클래스를 만들 때 main메소드가 없는 클래스는 public를 생략하자.

클래스의 이름은 식별자여서 같은 이름으로 하나만 생성할 수 있다.

이전 코드 MyClass2.java파일 안에 Human 클래스를 주석하고 같은 com.human.ex 패키지에 Human.java 파일을 만들어 public를 붙여서 Human클래스를 만들어 보자. 하나의 패키지에는 이름이 같은 클래스가 존재하면 안되기 때문에 이전에 만든 Human 클래스를 주석을 하지 않으면 하나의 패키지에 2개의 같은 이름 클래스가 존재하게 되어 문제가 발생하게 된다. 2개의 같은 이름의 클래스가 있으면 문제가 되어서 MyClass2.java안에 Human클래스를 주석 하였다.



다음은 Human.java 과 MyClass3.java 2개의 파일 내용이다. Human 클래스의 내용을 하나의 자바 파일로 분리해서 작업하는 예제이다.

//다음은 Human.java 파일 내용

```java
package com.human.ex;
public class Human {
	public String name="홍길동";
	public int age = 10;
	public double height=169.3;
}
```

MyClass3.java 파일을 만들어 Human.java에 만든 Human클래스를 사용해보자.

//다음은 MyClass3.java파일이다.

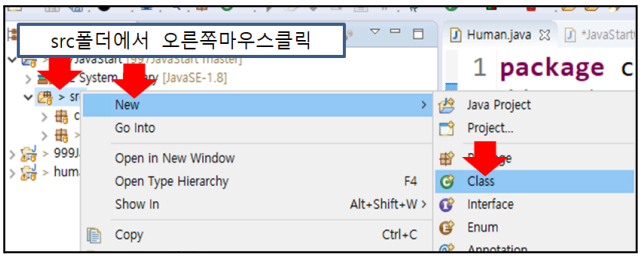
```java
package com.human.ex;
public class MyClass3 {
  public static void main(String[] args) {
//여기에 작성한 코드는 Human.java파일에 기술된 Human 클래스를 사용하였다.
	Human h = new Human();
	System.out.println(h);//참조자료형이여서 h에는 주소가 들어 있다.
	System.out.println(h.name);//클래스의 실제데이터는 .를 찍어서 접근
	System.out.println(h.age);//클래스 만들때의 초기 기본값이 들어 있다
	System.out.println(h.height);
	h.name="홍길남";//기존 들어있는 값을 새로운 값으로 변경한다.
	h.age=20;
	h.height=150;
	System.out.println("변경후");
	System.out.println(h.name);
	System.out.println(h.age);
	System.out.println(h.height);
  }
}
//다음을 만들어 보자.
//Cat Dog 클래스를 각각의 파일로 만들어서 데이터를 추가해서 내용을 출력해 보자.
```

패키지가 다르면 같은 이름의 클래스를 만들수 있다. 같은 이름의 클래스 2개를 만들어 사용해 보자.

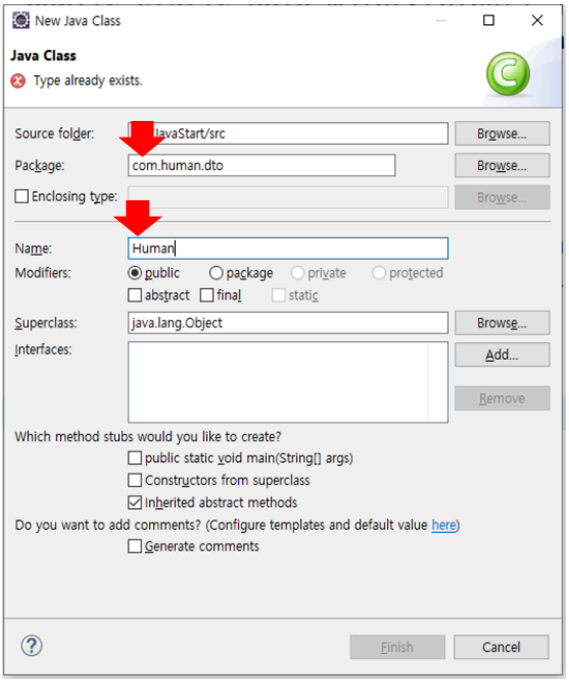
같은 패키지에 같은 이름 클래스를 만들수 없지만 패키지가 다르면 이름이 같은 클래스를 만들 수 있다. com.human.dto에 Human클래스를 만들어 보자. 다음에 코드와 함께 만드는 방법을 정리해 두었다. 확인한 후 만들어 실행해 보자.

```java
package com.human.dto;
public class Human {
	public String name="홍길남";
	public int age = 11;
	public double height=169.4;
}
```

1.일단 file 메뉴 >> new 메뉴 >> java Project 메뉴를 선택하여 프로젝트를 만들어 보자. 이미 999java01로 프로젝트를 만들었으니 확인하고 다음으로 넘어 가자.

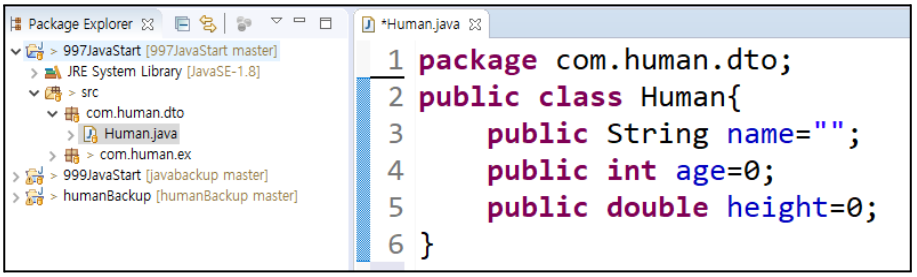


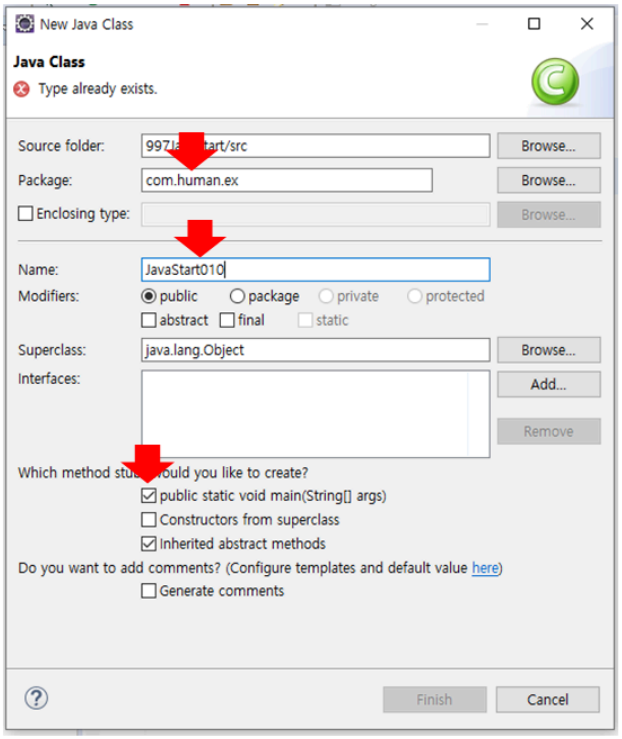
2. 프로젝트안의 src폴더에서 오른쪽 마우스를 클릭한 다음 new 다음에 class파일을 선택하면 새로운 클래스 파일을 만들 수 있다. 원하는 메뉴 classs가 없으면 other메뉴를 선택해서 검색하자.



3.왼쪽 처럼 new java class창이 뜨면 Package 부분에 com.human.dto name부분에 Human를 입력한 다음 맨아래 finish버튼을 클릭하면 새로운 창이 열리고 java코드를 추가하면 된다.

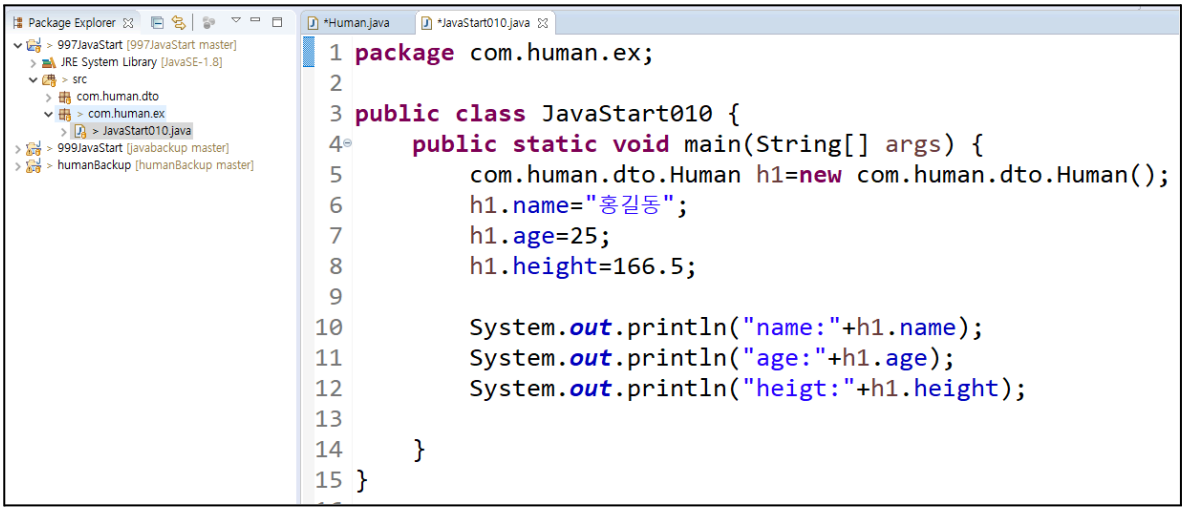
4.생성된 Human.java파일을 클릭해서 아래 처림 Human클래스를 만들어 보자. 작업을 마무리하면 패키지가 다른 2개의 Human클래스가 만들어 진다.





5. main이 포함된 새로운 자바 클래스 파일을 하나더 등록해 보자. 왼쪽 이미지 처럼 “new java class” 창이 뜨면 package에 com.human.ex를 입력하고 name창에 JavaStart010 대신에 MyClass04을 입력 하고 public static void main 체크박스를 클릭하여 체크한 다음 finish버튼을 누른다. finish 버튼을 클릭 할 수 없으면 이미 같은 이름의 데이터들이 존재 하거나 잘못된 입력이 있어서 그럴 수 있다. 변수 명명법을 잘 확인해 보자.

6. MyClass04.java를 클릭해서 입력 창이 뜨면 아래 코드를 입력하고 실행해 보면 파일은 다른 패키지로 분리되어 있지만 동작하고 있는 프로그램을 확인 할 수 있다.



이미지 5번 라인을 보면 파일에서 분리한 클래스를 패키지 까지 포함해서 전체 이름으로 사용하고 있다. 파일이 분리되면 패키지를 포함한 전체 이름으로 적어야 사용 할 수 있다. Human클래스 패키지를 생략하고 클래스를 구현하면 dto에 있는 클래스가 생성된것이 아니라 ex에 있는 Human클래스가 생성된 것이다.

클래스 이름만 같을뿐 다른 클래스를 만들어 사용하고 있다.

실질적으로 클래스를 사용할 때에는 패키지를 포함한 전체 주소를 사용해야 한다.

```
com.human.dto.Human h3=new com.human.dto.Human();
com.human.ex.Human h2=new com.human.ex.Human();
```

Human이라고 기술해서 쓰고 싶다면 import를 사용하면 된다.

import com.human.dto.Human; 로 기술하면 해당 파일에서는 Human클래스는 패키지를 생략하고 Human human = new Human();로 기술해서 사용 할 수 있다.

import com.human.dto.*; 로 기술하면 com.human.dto패키지 안에 있는 모든 클래스를 해당 파일에서 기술할때 생략해서 사용할 수 있다.

현재는 com.human.dto패키지 안에 Human클래스 밖에 없지만 다른 클래스들이 있다면 다른 클래스들도 패키지명을 생략 할 수 있게 된다.

클래스에 import 사용방법

1. 클래스생성시 패키지를 포함한 전체이름으로 클래스를 생성해야 한다.
2. 같은 패키지 안에서는 패키지 명을 생략하고 클래스 이름만 사용할 수 있다.
3. 다른 패키지 일때는 패키지명을 포함한 클래스 이름으로 반드시 기술해야 한다. 긴 패키지 명을 사용하고 싶지 않으면 import를 사용한다.
4. import com.human.dto.클래스이름 (import com.human.dto.Human;)의 경우 해당 기술된 클래스(Human)만 패키지명을 생략하고 기술 할 수 있다.
5. import com.human.dto.* 일때는 해당 패키지에 있는 모든 클래스에 적용 된다.
6. import로 등록했다 하더라도 식별이 불가능하면 풀이름으로 클래스 명을 기술해야 한다.

```
package com.human.ex;
import com.human.dto.Human; //4. com.human.dto밑에 있는 Human클래스 등록
import com.human.dto.*;     //5. com.human.dto밑에 있는 모든 클래스 등록
public class MyClass4 {
  public static void main(String[] args) {
      //1.원래 com.human.dto.Human 사용하려면 다음과 같다.
      com.human.dto.Human h=new com.human.dto.Human();//3
      //2.import하면 com.human.dto.Human 생성시 다음과 같이 사용할 수 있다.
      Human h1=new Human();

      //6. 만약에 Human클래스를 식별 할 수 없으면 전체 이름으로 기술
      //com.human.ex.Human 클래스 사용시 반드시 풀이름으로 사용해야 한다.
      com.human.ex.Human h2 = com.human.ex.Human();
      //Human h1=new Human();은 com.human.dto.Human 클래스만 생성된다.
      com.human.ex.Human 클래스를 사용하려면 전체 기술를 해야 한다.
  }
}
```

이전에 사용자 입력으로 사용한 Scanner 클래스를 java.util.Scanner로 선언하는 것과

같은 이유이다. Scanner 클래스가 다른 패키지에 선언되어 있어서 해당 파일이 존재하는 패키지 전체를 기술한 것이다. 이것도 상단에 import java.util.* 를 추가 하면 메인에서 java.util.Scanner가 아닌 Scanner라고 기술 하여도 큰 문제 없이 사용할 수 있다.

```
Scanner sc=new Scanner(System.in);
```

```
1 package com.human.ex;
2 import com.human.dto.*;
3 import com.human.dto.Human;
4 public class JavaStart010 {
5     public static void main(String[] args) {
6         Human h1=new Human();
7         h1.name="홍길동";
8         h1.age=25;
9         h1.height=166.5;
```

왼쪽 이미지 6번 라인 처럼 클래스 이름으로만 사용하고 싶다면 2,3번 처럼 import명령어를 사용해서 등록해야 한다. 2번의 * 의 의미는 com.human.dto 밑에 정의된 모든 클래스를 패키지명 없이 사용할 수 있다는 의미이고 3번 라인은 com.human.dto 밑에 Human클래스를 패키지명 없이 사용 할 수 있다는 의미이다. 여기서는 Human클래스만 사용하고 있기 때문에 2번과 3번중 하나만 기술해도 동작한다.

import를 사용해서 식별이 가능한 클래스는 패키지명을 생략할 수 있다. 여기서 2개의 Human클래스는 import를 사용하여도 두 클래스를 식별할 수 없어 하나만 import로 등록하면 등록된 패키지로 Human클래스를 생략해서 사용할 수 있지만, 둘다 import하면 식별이 불가능 하게 되어 패키지명을 생략 할 수 없다.

같은 폴더에 있는 클래스의 경우 선언시 패키지명을 생략 할 수 있어서 그동안 import없이도 Human h1=new Human();과 같이 패키지명을 선언하여 사용하였다.

Scanner클래스를 다음과 같이 선언해서 사용한 이유도 이런 이유 이다.

```
java.util.Scanner sc =new java.util.Scanner(System.in);
```

누군가 java.util패키지에 Scanner라는 클래스를 만들어 놓은 것이다.

너무 길기 때문에 줄여서 사용하려면 import 를 사용해야 한다.

package밑에 기술하면 되고 import com.human.dto.*; 라 기술하면 com.human.dto패키지 폴더에 있는 모든 클래스를 해당 파일에서 com.human.dto를 생략하고 사용할 수 있다.

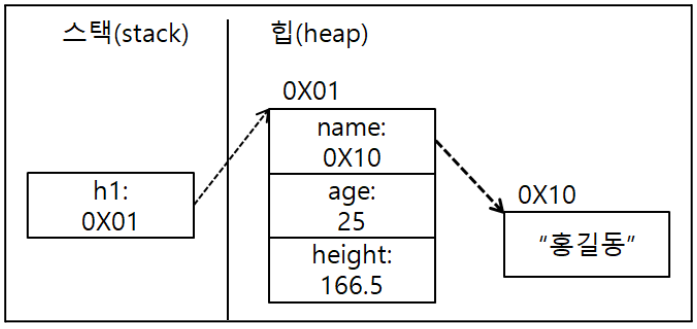
import com.human.dto.Human; 이라 기술하면 com.human.dto패키지 폴더에 있는 Human 클래스만 사용할때 com.human.dto를 생략할 수 있다.

```java
package com.human.ex;
//1. 사용자 정의 데이터 클래스 정의
class Human{
    public String name="";
    public int age=0;
    public double height=0;
}

public class JavaStart007 {
    public static void main(String[] args) {
        //2. 사용을 위한 클래스 선언
        Human h1=new Human();
        //3. 클래스 데이터 입력
        h1.name="홍길동";
        h1.age=25;
        h1.height=166.5;
        //4. 클래스 데이터 출력
        System.out.println("name:"+h1.name);
        System.out.println("age:"+h1.age);
        System.out.println("heigt:"+h1.height);

    }
}
```

다음은 왼쪽 이미지는 Human 클래스를 조작하는 코드이다. 이클립스에서 프로젝트를 새로 생성해서 스스로 작성해서 구현해 보자.

구현시 주의할 점은 1번 라인 패키지는 class 만들때 사용한 패키지 정보를 기술한 것이다. 본인이 만든 패키지와 일치해야 한다. 3번 라인에 public이 생략되어 있다. 같은 파일에 클래스가 2개여서 그렇다.

이전에 Human 클래스를 Human.java로 만들어 놓았다면 Human클래스를 주석해야 한다. 같은 패키지에 같은 이름의 클래스를 만들수 없다.



10번 라인이 메인 메소드 부분이다. 따라서, 해당 메소드의 클래스와 파일 이름은 동일해야 하므로 파일 이름은 JavaStart007.java 이여야 한다.

상위 코드는 사람 관련 변수를 클래스로 작성한 결과물이다. 실행 시켜 보고 코드를 보고 메모리 그림을 그려보자. 왼쪽 이미지는 해당 코드의 메모리 상태를 그린 이미지다 보지 말고 그려보고 어렵다면 확인해 보고 안보고 그릴 수 있을때 까지 반복해서 그려보자.

String은 참조 데이터여서 클래스에 저장 공간이 잡힐때 여전히 실제 데이터를 저장 할 수 있는 주소만 잡혔다는 사실을 잊지말자.

```
Human h1=new Human();
```

위의 코드를 설명하면 new Human()을 통해 Human 클래스의 새 인스턴스를 생성하는 과정입니다. 이때, 메모리에서 어떤 일이 일어나는지 쉽게 풀어볼게요:

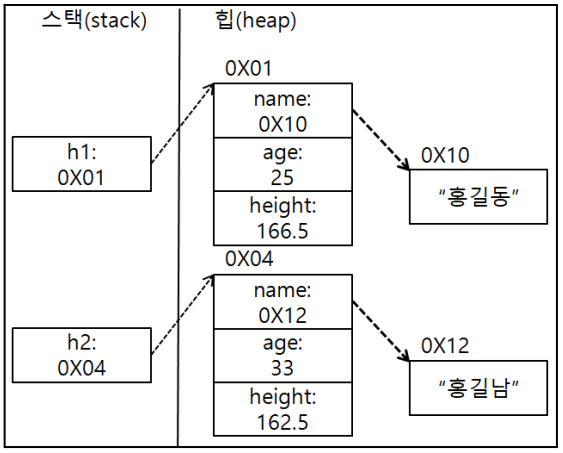
1. **힙 메모리에서의 할당**:
    - new Human()을 호출하면, Human 클래스의 필드(예를 들어 age, name, height, birthday)가 힙 메모리에서 할당됩니다. 이 필드들은 클래스의 속성(변수)이고, 생성된 인스턴스만이 가지고 있는 고유한 값들입니다.

2. **메모리 주소 저장**:
    - 힙 메모리에 Human 객체가 생성되면, 이 객체의 시작 위치(메모리 주소)가 할당됩니다.
    - 이 메모리 주소가 바로 객체를 가리키는 주소이고, h1 변수는 이 주소를 참조합니다. 즉, h1이 실제 데이터(객체)의 위치를 알려주는 역할을 하며, 이 덕분에 h1을 통해 Human 객체의 필드들을 사용할 수 있게 됩니다.
3. **간단히 말하면**:
    - h1은 Human 객체의 메모리 주소를 가지고 있는 변수이고, 그 주소를 통해 힙 메모리의 Human 객체에 접근할 수 있습니다.

결국, h1은 힙에 있는 Human 객체의 위치(주소)를 가리키는 역할을 하고, 이를 통해 그 객체에 접근하고 필드 값을 읽거나 수정할 수 있게 됩니다.

다음 코드는 클래스 인스턴스 2개 생성 하는 코드이다. 클래스 변수가 필요 하다면 필요한 만큼 새로운 인스턴스를 new 연산자를 이용해 힙에 생성해서 사용할 수 있다. 다음 기술한 코드를 확인하고 메모리에 생성된 데이터를 그려보자. 안보고 그릴 수 있을 때 까지 확인해 보자.

```
Human h1=new Human();
h1.name="홍길동";
h1.age=25;
h1.height=166.5;
System.out.println("name:"+h1.name);
System.out.println("age:"+h1.age);
System.out.println("heigt:"+h1.height);

Human h2=new Human();
h2.name="홍길남";
h2.age=33;
h2.height=162.5;
System.out.println("name:"+h2.name);
System.out.println("age:"+h2.age);
System.out.println("heigt:"+h2.height);
```



```java
class Human{
    public String name="";
    public int age=0;
    public double height=0;
}
class Dog{
    public String name="";
    public int age;
    public String eyeColor="";
}
public class JavaStart009 {
    public static void main(String[] args) {
        Dog dog=new Dog();
        dog.name="홍";
        dog.age=23;
        dog.eyeColor="노랑";
        System.out.println("name:"+dog.name);
        System.out.println("age:"+dog.age);
        System.out.println("eyeColor:"+dog.eyeColor);
        Human h1=new Human();
        h1.name="홍길동";
        h1.age=25;
        h1.height=166.5;
```

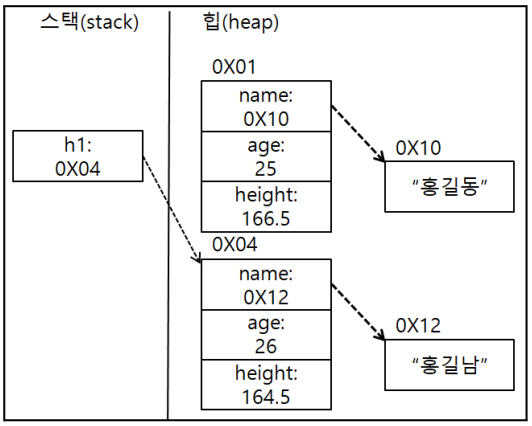
왼쪽 코드는 새로운 Dog클래스를 추가해서 프로그램을 구현한 코드이다. 프로그램 클래스 추가가 필요하면 왼쪽처럼 추가하여 사용하면 된다. 같은 패키지에 같은 이름으로 클래스를 만들면 문제가 발생할 수 있으니 클래스에 특별한 이유없이 에러 표시가 된다면 이미 존재하는 클래스가 아닌지 확인해 보자.

7,8,9 라인 처럼 클래스 필드에서 값을 할당하여 선언 할 수 도 있다.

코드를 실행해보고 메모리를 그려보자.

```java
Human h1=new Human();
h1.name="홍길동";
h1.age=25;
h1.height=166.5;
System.out.println("name:"+h1.name);
System.out.println("age:"+h1.age);
System.out.println("heigt:"+h1.height);

h1=new Human();
h1.name="홍길남";
h1.age=26;
h1.height=164.5;
System.out.println("name:"+h1.name);
System.out.println("age:"+h1.age);
System.out.println("heigt:"+h1.height);
```

다음 코드는 Human 클래스 인스턴스 h1에 new 연산자를 사용해서 같은 변수 h1에 2번 할당하여 사용하는 경우를 보여주는 예제이다. h1에 클래스 필드를 두번 할당하면 h1에 처음 할당한 주소는 두번째에 할당한 주소로 변경되어 이전 데이터에 접근할 수 없게되어 메모리에서 사라진다. 코드를 확인하고 메모리 상태를 안보고 그릴 수 있을 때 까지 그려보자.



12라인과 20라인을 확인해 보면 h1변수에 새로운 변수를 선언하지 않고 기존 변수 h1에 다시 Human객체를 생성 하였다. 이럴 경우 12라인에서 h1에 저장되어 있는 Human 객체의 주소는 20라인에서 새롭게 만들어진 Human객체의 주소로 변경되어 12라인에서 생성한 Human객체는 접근할 수 있는 방법이 없어져서 더이상 접근할 수 없는 데이터가 된다.

다음 코드에서 변수를 통해서 접근할 수있는 정수는 12 뿐이다. 변수를 통해서 접근할 수 있는 Human 인스턴스는 총 3개가 생성되지만 new Human()으로 생성된 마지막 Human 인스턴스만 접근 가능하다.

다음질문에 답해보자. 현재 메모리에 저장되어 있는 데이터는 어떤것인가?

```
int a=10; a=11; a=12;

Human h=new Human(); h=new Human(); h=new Human();
```

다음에는 클래스 필드에 new 연산자를 사용하지 않고 이미 생성해 놓은 클래스 필드 주소를 넣었을때 어떻게 되는지 확인해 보자.

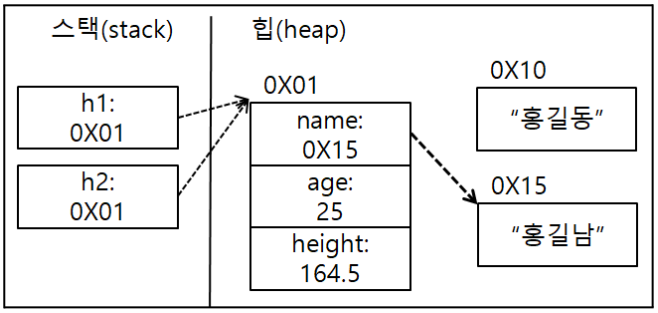
(Human h2=h1;)

다음 코드를 확인해 보자.

12라인과 20라인을 살펴 보면 20라인에 h2=h1은 h1이 가지고 있는 값 즉 h1의 실제 데이터가 있는 주소를 h2에 저장 한다. 결과적으로 h1이 가리키는 데이터의 주소와 h2가 가리키는 데이터 주소는 같아져 메모리의 상태는 다음 이미지 처럼 된다.

```
12      Human h1=new Human();
13      h1.name="홍길동";
14      h1.age=25;
15      h1.height=166.5;
16      System.out.println("name:"+h1.name);
17      System.out.println("age:"+h1.age);
18      System.out.println("heigt:"+h1.height);
19
20      Human h2=h1;
21      h2.name="홍길남";
22      h2.height=164.5;
23      System.out.println("h1.name:"+h1.name);
24      System.out.println("h1.age:"+h1.age);
25      System.out.println("h1.heigt:"+h1.height);
26      System.out.println("h2.name:"+h2.name);
27      System.out.println("h2.age:"+h2.age);
28      System.out.println("h2.heigt:"+h2.height);
```

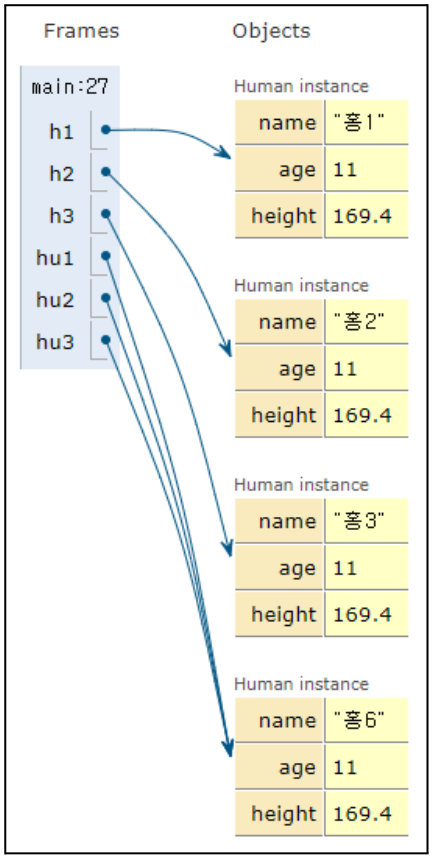
13라인에서 h1.name에 홍길동을 넣고 20라인에서 h1에 h2를 넣고, 21라인에서 h2.name에 홍길남을 넣고, 23라인에서 h1.name 26라인에서 h2.name을 출력해보면 홍길동, 홍길남이 출력될거 같지만 실제로 실행해 보면 홍길남이 2번 출력되는 것을 확인할 수 있다. h1.height와 h2.height도 같은 값이 출력되는 것을 확인 할 수 있다.



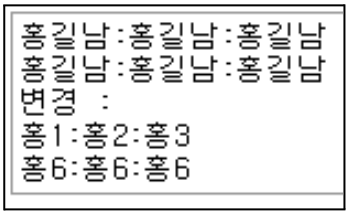
이유는 h1, h2 인스턴스가 같은 주소를 가지고 있기 때문에 h1,h2 각각 다른 변수로 인스턴스에 접근 하여도 같은 인스턴스에 해당한다. 따라서, 한쪽 변수의 값을 변경하면 다른쪽 변수에 영향을 준다. 이와 같은 문제가 발생하지 않도록 신경써서 코딩 해야 한다.

상위 이미지는 프로그램 실행결과 메모리 상태를 그림으로 그린 것이다. 이미지를 가지고 어떤 문제가 발생하는지 확인해 보고 코드를 보고 이미지를 그릴 수 있도록 숙지 하자.

다음은 MyClass5.java파일을 실행 시켜서 결과를 확인하고 생각과 같게 출력 되었는지 확인해 보자.

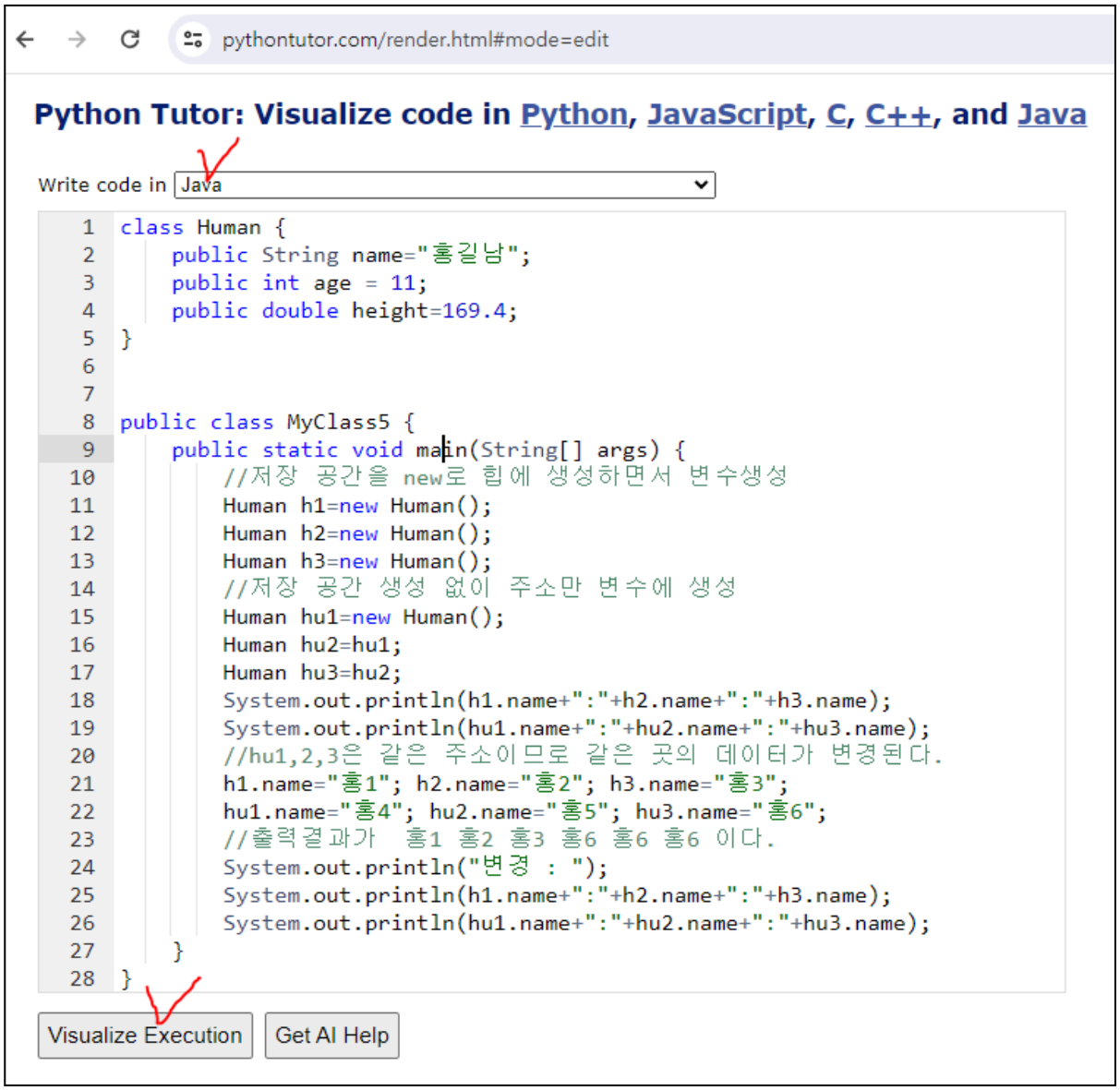



```java
package com.human.ex;
public class MyClass5 {
   public static void main(String[] args) {
          //저장 공간을 new로 힙에 생성하면서 변수생성
          Human h1=new Human();
          Human h2=new Human();
          Human h3=new Human();
          //저장 공간 생성 없이 주소만 변수에 생성
          Human hu1=new Human();
          Human hu2=hu1;
          Human hu3=hu2;

System.out.println(h1.name+":"+h2.name+":"+h3.name);
System.out.println(hu1.name+":"+hu2.name+":"+hu3.name);
//hu1,2,3은 같은 주소이므로 같은 곳의 데이터가 변경된다.
h1.name="홍1"; h2.name="홍2"; h3.name="홍3";
hu1.name="홍4"; hu2.name="홍5"; hu3.name="홍6";
//출력결과가  홍1 홍2 홍3 홍6 홍6 홍6 이다.
System.out.println("변경 : ");
System.out.println(h1.name+":"+h2.name+":"+h3.name);
System.out.println(hu1.name+":"+hu2.name+":"+hu3.name);
   }
}
```

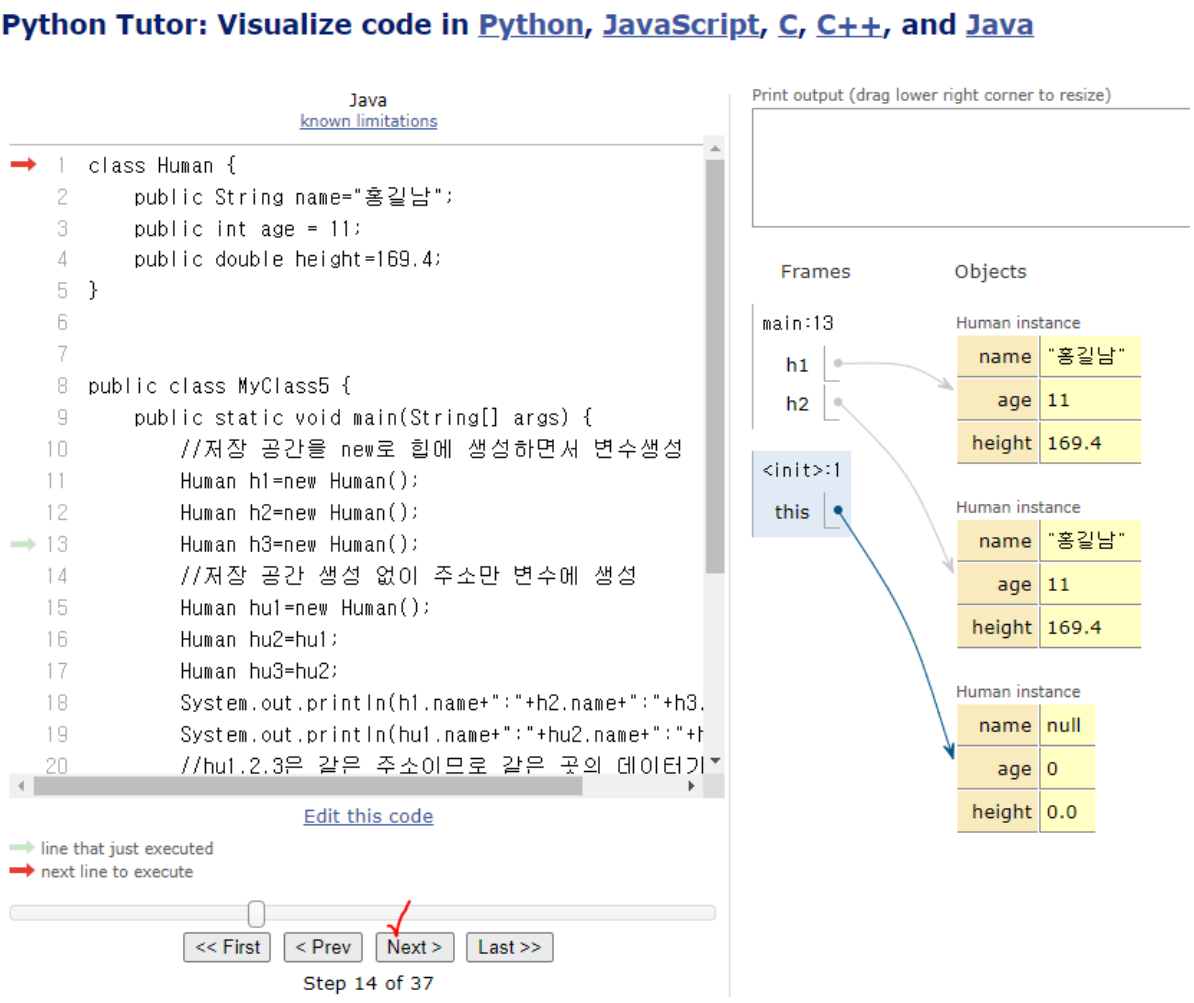


왼쪽 이미지는 출력 결과이다.

상위 메모리를 그린 이미지는 https://pythontutor.com/visualize.html#mode=edit 에 들어가서 아래 이미지 처럼 write code in에서 java를 선택한 다음 에디트 박스에 실행 하고자 하는 자바 코드를 넣은 후 하단의 Visualize Execution 버튼을 누르면 메모리를 순서대로 그려주는 화면으로 이동한다.



다음 이미지 하단의 next 버튼을 누를 때마다 메모리 생태가 화면에 그려진다.



연습문제

1.클래스, 인스턴스에 대해서 설명하시오.

2. 클래스를 선언하고 사용하는 방법을 기술하시오.

3. Human h3;와 new Human();와 h3=new Human();의 의미를 기술 하시오.

4. . 연산자에 대해서 설명하시오.

5. public class Human 이아닌 class Human이라고 선언한 이유를 설명 하시오.

6.Human h=new Human()에서 Human클래스는 패키지를 포함한 전체 경로를 사용해야 하는데 해당 코드가 동작하는 이유는 무엇인가?

7. import를 사용하는 이유와 * 의 의미를 기술 하시오.

8. 우리 주위의 객체를 클래스로 만들어서 데이터를 넣고 출력해 보자. 클래스 파일을 분리해서 만들어 보자. 핸드폰, 학생, 자동차, 엘리베이터 본인이 생각한 객체 총 5개를 만들어 보자.

9. 다음 이미지를 보고 데이터를 뽑아 Tree 클래스로 만들어 보자. 어려우면 다음 힌트들을 확인해 보자.

힌트 - 필요한 데이터 선정 6개

나무 이름,나무 가격, 키우는데 걸리는 시간, 경험치, 현재 심은 나무수, 최대 심을 수 있는 나무수

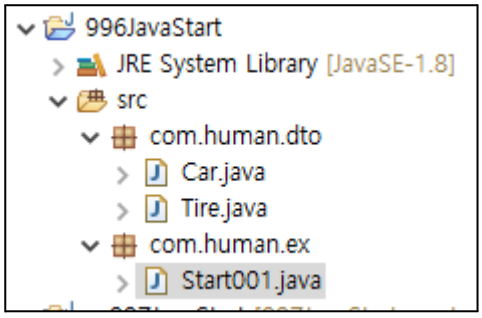
10. 상위 이미지는 노트북 관련 데이터들이다. 원하는 데이터를 뽑아서 클래스를 만들고 다음 이미지의 데이터를 넣어 화면에 출력해 보자.

클래스는 사용자 정의 변수로 생각해도 된다. 기존에 없는 자료형을 만들어 사람이나 강아지등의 정보를 저장 하고, 복잡해 질 수 있는 관련 데이터를 잘 묶어 프로그램을 쉽게 개발 할 수 있게 해준다.

.은 주소 연산자이다. 주소가 들어있는 변수에 찍어서 변수에 들어 있는 주소 위치의 실제 데이터에 접근할 수있다.

클래스 안에 변수(필드)로 다른 클래스를 사용 할 수도 있다.

다음과 같은 자동차를 클래스로 만들어 데이터를 저장 하자. 자동차의 색상, 바퀴, 제조사, 최고속도, 차종을 저장할 수 있어야 하고 바퀴는 제조사, 가격, 주행 거리 정보가 들어 있어야 한다. 3대의 자동차 정보를 입력하고 출력하는 프로그램을 만들어 보자.




```java
package com.human.dto;
public class Car {
    public String color="";
    public String company="";
    public int maxSpeed=0;
    public String model="";
    public Tire tire=new Tire();
}
```

```java
package com.human.dto;
// 바퀴는 제조사,가격,주행거리
public class Tire {
    public String company="";
    public double price=0;
    public double mileage=0;
}
```

1. 이미지 처럼 Car, Tire클래스를 com.human.dto에 만들자. Car클래스 안에서 Tire클래스를 사용하고 있으므로 Tire클래스를 먼저 만들어야 문제가 생기지 않는다.

2. Car, Tire클래스와 프로그램을 실행하기 위한 Start001클스를 구성해 보자.

*Start001.java

```java
1  package com.human.ex;
2  import com.human.dto.Car;
3  public class Start001 {
4      public static void main(String[] args) {
5          Car car=new Car();
6          car.color="빨강";
7          car.company="현대";
8          car.maxSpeed=200;
9          car.model="소나타";
10         car.tire.company="금호타이어";
11         car.tire.mileage=20;
12         car.tire.price=50000;
13
```

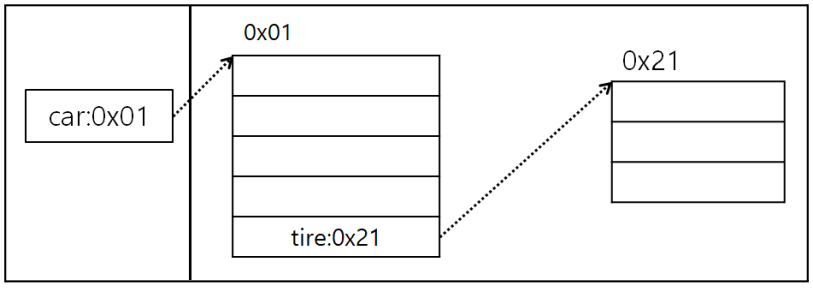
Car클래스를 만들어 사용 하려면 일단 옆 이미지 5번 라인처럼 객체 생성을 해야 한다. 생성한 객체에 값을 넣으려면 6~12라인 처럼 처리를 해야 한다.

클래스안에 클래스로 선언된 값에 접근 하려면 10~13, 21~23줄에서 car.tire.price 처럼 지속적으로 .를 찍어서 접근하면 된다. 14~23라인에서 값이 분리된 상태의 객체를 만들어 값을 복사하고 있고 25~28 라인에서 객체의 값을 출력하고 있다.

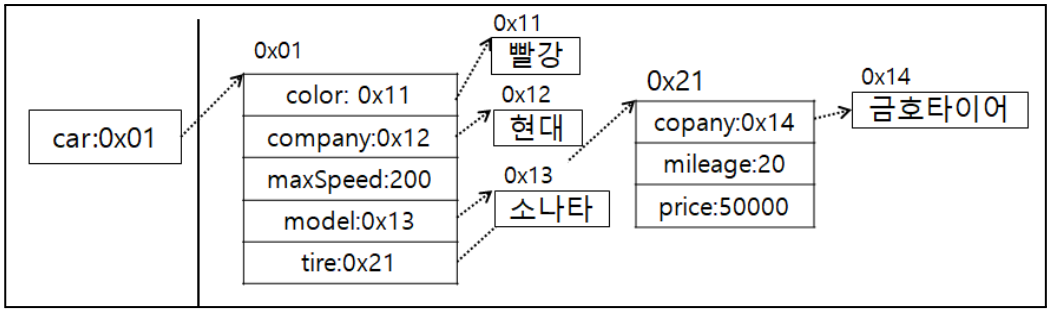
```
14          Car newCar=new Car();
15          // newCar=car;
16          newCar.color=car.color;
17          newCar.company=car.company;
18          newCar.maxSpeed=car.maxSpeed;
19          newCar.model=car.model;
20          //newCar.tire=car.tire;
21          newCar.tire.company=car.tire.company;
22          newCar.tire.mileage=car.tire.mileage;
23          newCar.tire.price=car.tire.price;
24
25          System.out.println(car.color);
26          System.out.println(car.tire.company);
27          System.out.println(newCar.color);
28          System.out.println(newCar.tire.company);
29      }
30  }
```

클래스안의 클래스는 각각의 인스턴스가 car와 tire라면 car.tire. 과 같이 계속 .을 연속적으로 찍어서 접근할 수 있다.

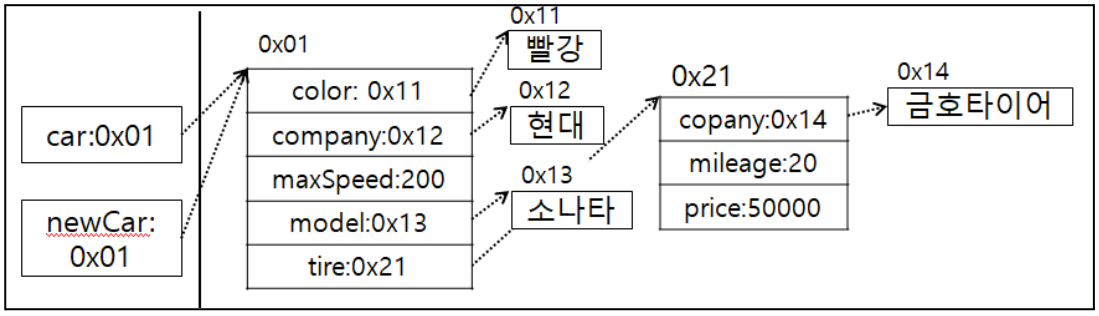
왼쪽이 스택이고 오른쪽이 힙이다. 메모리에 어떻게 생성 되는지 확인해 보자. 5번 라인이 실행되면 메모리에 왼쪽과 같이 잡힌다.

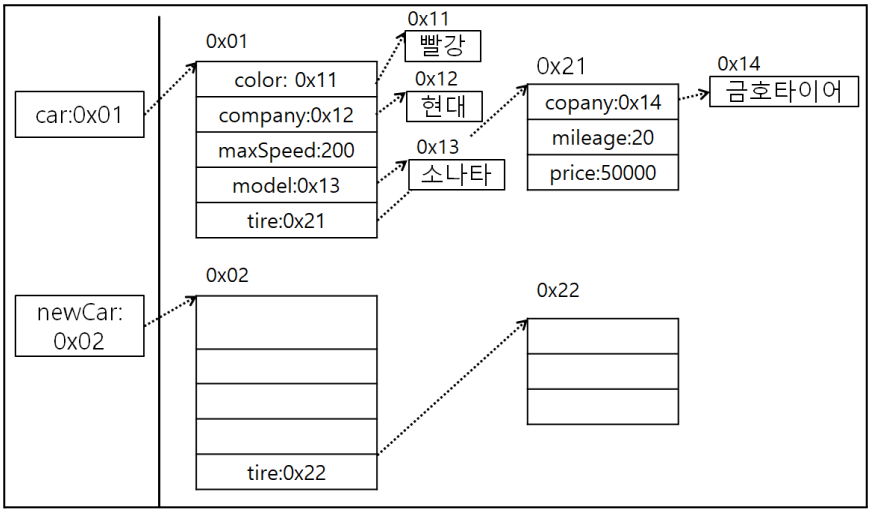


6~12라인까지 실행되면 메모리에 다음 이미지 처럼 생성된다.



14라인 처럼 선언하지 않고 주석되어 있는 15라인처럼 Car newCar=car;로 선언하고 작업을 하게 되면 같은 객체의 값에 접근하게 되어 원하는 것과 다른 결과를 얻을 수 있다. 아래 이미지를 확인하자.

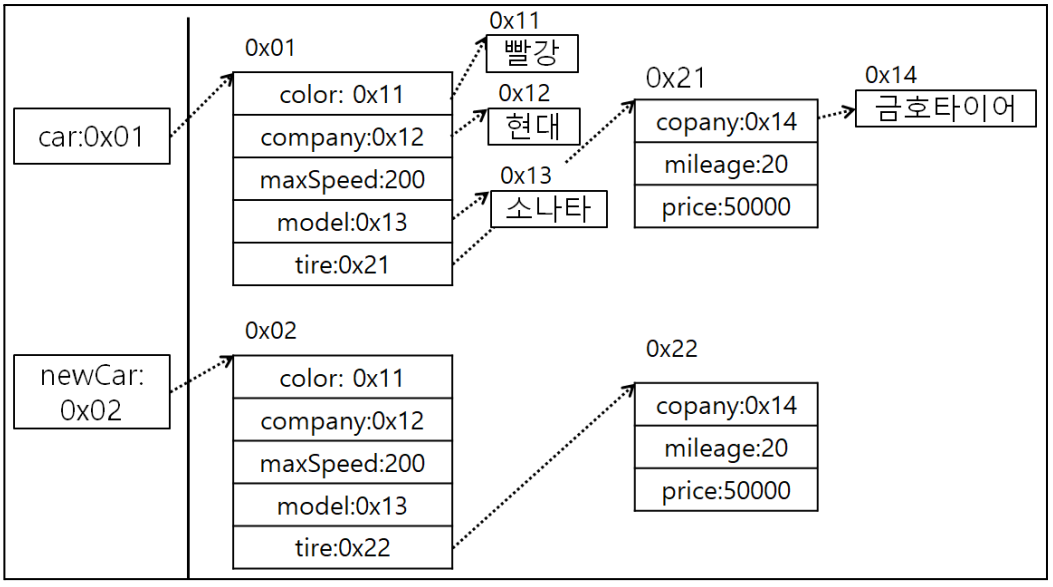


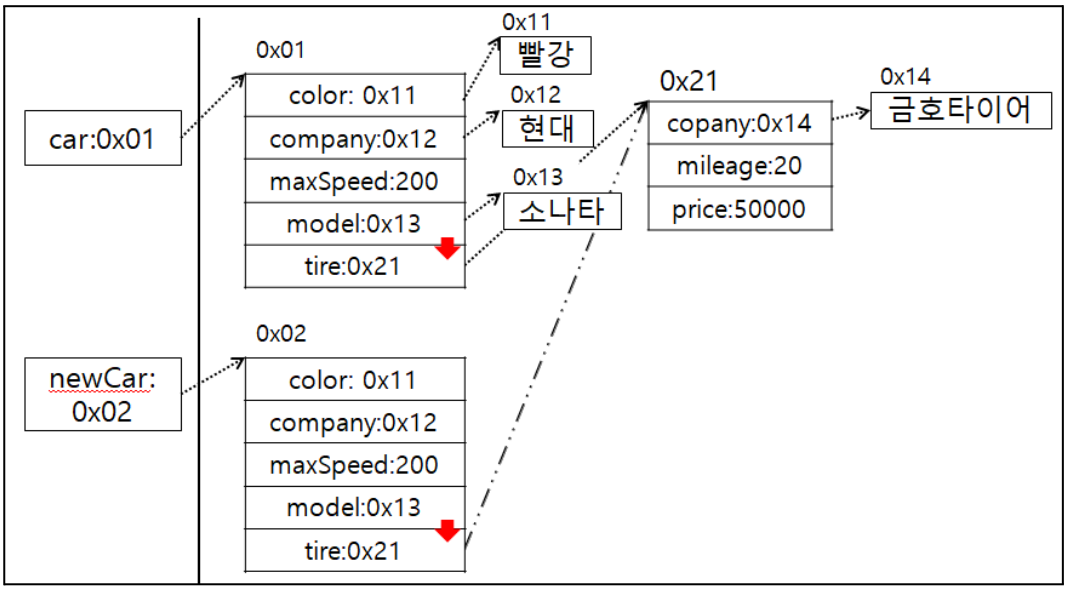


14라인 처럼 선언해서 실행하면 다음과 같은 결과가 메모리에 생성 될 것이다.

20라인 주석부분을 제거하고 기술하면 아래 이미지 처럼 tire관련 객체들이 데이터를 공유하는 문제가 발생한다. 20번째 라인으로 인해서 원하는 것과 다른 결과를 얻을 수 있으니 조심해서 사용하자. 아래 이미지의 빨간색 화살표 부분 처럼 같은 타이어 주소가 들어가서 타이어주소를 공유하게 된다.

16~24라인 처럼 20라인을 주석하고 기술해야 완전히 분리된 결과를 얻을 수 있다.





모든 문제가 해결 되었다. 그런데 상위 그림에서 문자열 상수의 주소를 공유하고 있는 것을 확인할 수 있다. 상수는 변경이 불가능해서 데이터가 같은 상수는 주소를 공유한다.

연습문제

1. 이전에 만든 Tree클래스에 나무의 주인 정보를 저장하는 Owner클래스를 만들어 Tree 클래스에 추가해 보고 메인 메소드가 포함되어 있는 TreeEx 클래스의 메인 메소드에서 데이터를 입력받아 출력해보자. 힌트) Car,Tire클래스와 Tree,Owner클래스의 관계가 같으니 참고해서 만들어 보자. Owner클래스 필드 - 이름, 나이, 연락처

> 17. 배열

다음을 기술하고 실행 결과를 확인해 보자.

```
// 1. 정수형 배열 선언 및 초기화 int[] numbers = {1, 2, 3, 4, 5};
System.out.println("1. 정수 배열: " + Arrays.toString(numbers));

// 2. 특정 길이의 정수형 배열 생성

int[] arrayWithSize = new int[5];

System.out.println("2.빈 배열(초기값 0):"+Arrays.toString(arrayWithSize));

// 3. 배열 요소 접근 및 수정

arrayWithSize[0] = 10;

arrayWithSize[4] = 20;

System.out.println("3. 요소 수정 후: " + Arrays.toString(arrayWithSize));

// 3. 배열 부분요소 읽어오기

System.out.println(arrayWithSize[1] + arrayWithSize[4]);

// 5. 배열 길이 확인

System.out.println("4. 배열 길이: " + arrayWithSize.length);
```

참조형 자료 배열은 여러개의 데이터를 한번에 생성 할때 사용한다. 배열을 생성하면 하나의 변수를 인덱스를 통해서 여러개의 데이터에 접근 할 수 있다.

변수에 5개의 저장 공간을 만들려면 `int a1=0, a2=0, a3=0, a4=0, a5=0;` 이런 식으로 일일이 하나씩 선언 해야 한다. 이런 어려움을 해결하기 위해서 한번에 여러 개의 데이터를 저장 할 수 있도록 선언하는 방법을 제공하는데 이 방법을 배열이라 한다.

일반 변수는 하나의 변수에 하나의 데이터를 저장하기 때문에 변수명 만으로 저장 공간에 접근할 수 있었지만, 배열은 하나의 변수에 여러개의 데이터를 저장해야 하기 때문에 변수명과 인덱스를 이용해서 저장 공간에 접근 할 수 있다.

```
//자료형 변수명[] = new 자료형[데이터생성갯수];

int arr1[] = new int[5];
int []arr2 = new int[5];
int []arr3;
arr3=new int[5];
int arr4[]= {0,0,0,0,0};
```

배열을 생성 하는 방법은 왼쪽과 같다.

배열 선언시 자료형 다음 변수명을 기술하고 변수명 앞이나 뒤에 []를 쓰고 = 다음에 new 연산자 자료형 [생성할 데이터 저장 공간개수];로 기술하면 된다.

배열은 선언과 함께 {}를 사용해서 배열의 내용을 초기화 할 수 있다. int a[]={1,2,3}; 라고 선언 하면 a라는 배열 안에 int자료형을 저장할 수 있는 공간 세 개를 할당하고 할당된 공간에 들어 갈 값이 1,2,3 이라는 것을 의미한다. 프로그램 중간에 사용할 수 없고 선언시에만 사용 가능하다.

int a[]=new int[3]; a[0]=1; a[1]=2; a[2]=3; 와 같은 의미이다.

int a0=1; int a1=2; int a2=3;과 같이 프로그램에서 사용할 수 있는 데이터 1,2,3를 생성해서 메모리에 저장한 상태이다.

배열의 각각의 값에 접근하는 방법은 인덱스를 사용하면 된다. 인덱스는 0부터 시작한다. a[0]의 의미는 a배열의 첫번째 값을 읽어 오라는 이야기 이고 a[0]=5;의 의미는 a배열의 첫번째 값으로 5를 넣으라는 의미이다. a[1]의 의미는 a배열의 두번째 값을 읽어 오라는 이야기 이고 a[1]=5;의 의미는 a배열의 두번째 값으로 5를 넣으라는 의미이다.

상위 이미지의 코드는 배열변수 arr1,arr2,arr3,arr4에 각각 int형으로 5개의 데이터를 저장할 수 있는 공간을 생성한 것이다.

`int a[] = new int[5];`는 정수형 배열을 생성하는 코드입니다. 이 코드가 하는 일을 단계별로 설명해볼게요:

1. 배열 선언:
    - `int a[];`는 `int` 타입의 배열을 선언하는 부분입니다.
    - 여기서 `a`는 정수형 요소들을 담을 배열이라는 것을 알리며, 아직 메모리에 배열 공간을 할당하지는 않습니다.
2. 배열 생성:
    - `new int[5];`는 배열을 실제로 생성하는 부분입니다.
    - `new` 키워드는 힙 메모리에 공간을 할당하고, 이 공간에 `int` 타입의 배열을 만듭니다.
    - `[5]`는 배열의 크기를 나타내며, 5개의 `int` 요소를 담을 수 있도록 메모리가 할당됩니다.
3. 초기화:
    - 배열이 생성되면 모든 요소는 자동으로 `0`으로 초기화됩니다. 그래서 `a[0]`, `a[1]`, ..., `a[4]`의 값은 모두 `0`입니다.
4. 참조 변수에 주소 할당:

- 생성된 배열의 메모리 주소가 a 변수에 저장됩니다.
- 이제 a 변수를 통해 배열의 요소들에 접근할 수 있습니다.

5. 배열 인덱스로 값 배열의 값 변경 및 읽어오기
   - a[2]=10
   - a[2]

결과적으로 a는 길이가 5인 정수형 배열을 가리키게 되며, 각 요소는 초기값으로 0을 가집니다.

다음 배열 예제들을 확인해 보자.

```
1.// 일반변수와 배열에 3명의 나이를 저장해서 출력해보자.
            int humanAge1=10;int humanAge2=12;int humanAge3=15;//선언
//    int humanAgeArr[]=new int [3]; or int []humanAgeArr=new int [3];
//          humanAgeArr[0]=10;
//          humanAgeArr[1]=12;
//          humanAgeArr[2]=15;
            int humanAgeArr[]= {10,12,15};//상위 주석 4개와 같은 의미이다.
            //human1의 나이를 출력해보자.
            System.out.println(humanAge1);
            System.out.println(humanAgeArr[0]); //배열에서 읽어오기
            //human2의 나이를 8로 변경해보자.
            humanAge2=8;
            humanAgeArr[1]=8;//배열의 값 변경하기
            //human3의 기존나이에 3를 더해보자.
            humanAge3=humanAge3+3;
            humanAgeArr[2]=humanAgeArr[2]+3;
            //human정보 출력하기
            System.out.println(humanAge1+":"+humanAge2+":"+humanAge3);
System.out.println(humanAgeArr[0]+":"+humanAgeArr[1]+":"+humanAgeArr[2]);
            System.out.println(humanAgeArr);//배열의 주소가 출력
            System.out.println(Arrays.toString(humanAgeArr));//배열내용출력

//Arrays.toString 메서드는 배열의 요소들을 문자열 형태로 쉽게 확인할 수 있도록 해주는 유용한 메서드
```

2.

```
// 배열 선언과 동시에 값 할당
int[] numbers1 = {10, 20, 30, 40, 50}; // 배열 선언과 동시에 값 할당
int[] numbers2 = new int[] {10, 20, 30, 40, 50}; // new 연산자와 함께 배열

// 배열 선언후 값 할당
```

```
// 타입[] 배열이름 = new 타입[배열크기];
int[] numbers = new int[5]; // 5개의 int형 요소를 저장하는 배열 선언
numbers[0] = 10;numbers[1] = 20;//5개의 공간에 값 할당
numbers[2] = 30;numbers[3] = 40;numbers[4] = 50;
int size = numbers.length; // 5  배열의 사이즈를 얻고 싶을때 기술 한다.
System.out.println(numbers.length); //배열의 크기
System.out.println(numbers);//배열의 주소
System.out.println(java.util.Arrays.toString(numbers));//배열의 내용 출력
```

3.

```
int[] numbers = {1, 2, 3, 4, 5};
// 배열 요소 접근
int num1 = numbers[0];   // 1
int num2 = numbers[2];   // 3
System.out.println(num1 + num2);
// 배열 요소 변경
numbers[1] = 10;         // {1, 10, 3, 4, 5}
System.out.println(java.util.Arrays.toString(numbers));//배열의 내용 출력
numbers[3] = 8;          // {1, 10, 3, 8, 5}
System.out.println(java.util.Arrays.toString(numbers));//배열의 내용 출력
numbers[0] = num1 + numbers[4];// {6, 10, 3, 4, 5}
System.out.println(java.util.Arrays.toString(numbers));//배열의 내용 출력
```

배열 사용 방법은 다음과 같다.

a1, a2, a3 등과 같은 일반 변수는 저장 공간과 변수명이 1개 여서 a1에 1를 넣고 싶으면 a1=1 a2에 2를 넣고 싶으면 a2=2이런 식으로 1:1 매핑해서 접근 가능 하지만 배열 같은 경우에는 배열 이름 1개로 여러 개의 저장 공간을 접근 해야 하므로 인덱스를 사용한다.

배열은 같은 이름으로 여러 저장 공간에 접근하기 위해서 인덱스를 사용 하는데 배열 변수명[인덱스] 즉 a[0], a[1], a[2]이런 식으로 하나의 배열 명으로 여러 값에 접근할 수 있다. a[0], a[1], a[2] 각각이 하나의 데이터를 저장하는 저장 공간을 의미한다. 배열은 배열 변수명과 인덱스로 원하는 여러개의 값에 접근할 수 있다. 이때 사용하는 연산자가 []이며 의미는 변수가 가리키는 주소에 가서 인덱스 번째 있는 값에 접근 하라는 의미이다.

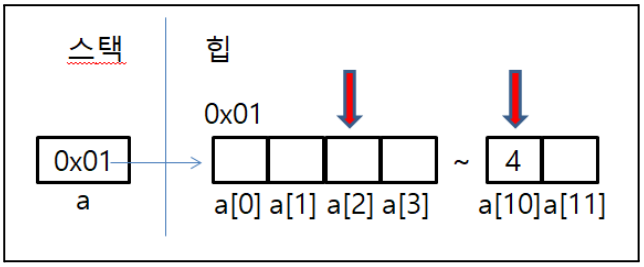
int a[]=new int[12]라 배열을 선언 하면 저장 할 수 있는 공간이 12개 생기며, 인덱스는 0 부터 시작하여 다음 데이터를 가리 킬때 마다 하나씩 증가 한다. 첫번째 값에 접근하려면 a[0] 두번째 값에 접근하려면 a[1], 세번째 값에 접근하려면 a[2]하면 되고 a[12] 값에 접근 하려하면 new int[12]로 선언하였기 때문에 인덱스가 0 부터 11까지 12개 존재하므로 인덱스 12는 존재하지 않는다.

배열의 값을 넣고 읽어 오는 방법은 다음과 같다.

배열 이름이 a이고 인덱스가 2인 배열의 값을 3으로 변경 하려면 a[2]=3으로 변경 할 수 있다. 읽어 올때는 a[2]라고 쓰면 된다.

배열은 참조 자료형이다. int a[]=new int[12]의 경우 실제 데이터는 힙에 연속된 12개의

int 정수를 저장할 수 있는 공간이 생긴다. 여기서 new 는 연산자인데 힙에 데이터를 생성하라는 의미이다. 힙은 메모리의 특정 위치로 힙메모리에 데이터를 저장하고 싶으면 new 연산자를 사용해서 연속된 메모리 공간을 생성해 데이터를 저장 할 수 있다. new 연산자를 사용 할 때마다 힙에 새로운 저장 공간을 할당하고 할당된 데이터의 시작 주소를 얻을 수 있다.



a라는 변수를 배열로 생성하려면 고유한 주소를 가진 힙 메모리에 새롭게 배열 저장 공간을 new로 생성한 다음 생성된 배열의 시작 주소를 변수 a에 넣고 배열 이름 a와 인덱스를 이용해 실제 데이터 저장 공간에 접근 할 수 있다.

[]은 주소 연산자이다. a가 가리키는 주소로 가서 해당 인덱스에 데이터를 읽어 오라는 의미이다. 그림에서 ~ 모양은 12개의 연속된 메모리 공간 중 일부를 그림에서 생략한 것이다. 시작 주소로 부터 다섯 번째 저장되어 있는 값을 읽어 오려면 인덱스가 0부터 시작하므로 a[4]형태로 읽어 오면 되고 값을 변경하고 싶으면 a[4]=10;과 같이 기술하면 된다.

첫 번째 화살표 위치에 숫자 13을 넣으려면 어떻게 해야 하는가?

두 번째 화살표 위치에 4라는 값을 읽어 오려면 어떻게 해야 하는가?

```
//int arr1[] = new int[3];
int arr1[] = {1,2,3};

//double arr2[] = new double[4];
double arr2[] = {1.1,2.1,3.1,4.1};

//String arr3[] = new String[5];
String arr3[] = {"강아지","고양이","붕어"};

//Human arr4[]=new Human[3];
Human arr4[]= {
        new Human(),
        new Human(),
        new Human()
};
```

다양한 자료형 int, double ,String, Human 클래스를 배열로 선언하고 초기화 하는 방법은 왼쪽 이미지와 같다. 주석된 부분이 초기화 없이 선언하는 방법이고 주석 안된 부분이 초기화를 통해서 값을 설정한 형태이다. ㅐ다음 상황들을 코드로 어떻게 기술하는지 확인해 보자.

int 4개를 저장할 수 있는 배열 a를 선언한 다음: int a=new int[4];

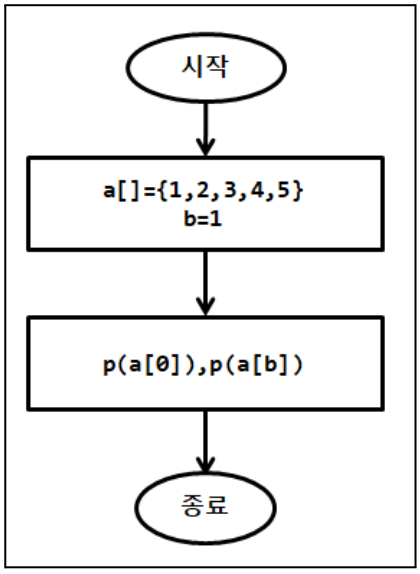
a라는 배열에 5,2,3를 넣은 다음 : a[0]=5, a[1]=2, a[2]=3

a배열의 인덱스[1]번의 값을 b만큼 증가시킨 다음 : a[1]=a[1]+b

a배열의 인덱스 0과 인덱스 1를 더해서 인덱스 2에 넣은 다음 : a[2]=a[0]+a[1]

a배열의 인덱스 0,1,2를 더한 값을 a[3]에 넣은 다음 : a[3]=a[0]+a[1]+a[2]

a 배열의 모든 값의 합을 변수 sum에 넣은 다음:sum=a[0]+a[1]+a[2]+a[3]



sum를 5로 나눈 값을 sum에 넣은 다음:sum=sum/5

sum를 출력해 보자. sum값은 얼마인가?

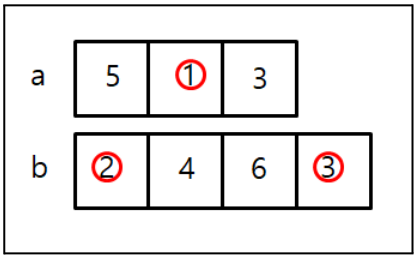
실제 프로그램으로 결과 값을 구해보자.

왼쪽 순서도 처럼 선언과 함께 배열의 값을 설정 하는 방법도 있다.

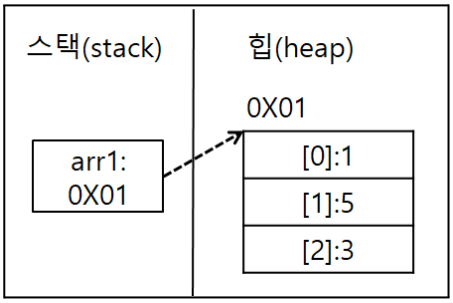
int a[]={1,2,3,4,5} 이렇게 하면 a[0]에 1, a[1]에 2, a[2]에 3, a[3]에 4, a[4]에 5가 들어 있는 것과 같은 결과 가 나올 것이다.

왼쪽 이미지 처럼 인덱스에 상수 대신 숫자 변수를 넣을 수 있다. a[]={1,2,3,4,5}, b=1, p(a[0]), p(a[b]) 와 같이 사용할 수 있고 실행 결과는 1과 2가 출력 된다. 코드를 작성해서 확인해 보자.

다음을 코드로 작성해 보고 실행 해 보자.



변수를 여러개 선언 하듯이 배열도 여러개 선언하여 사용할 수 있다. 왼쪽 이미지는 a 배열의 값 3개, b 배열의 4개 저장 할 수 있는 공간을 선언하여 총 7개의 저장 공간을 선언한 결과이다. a[0]을 이용해서 5값을 읽어 올 수 있고 b[2]을 이용하여 6값을 읽어 올수 있다. 동그라미 1번에 8를 넣으려면 a[1]=8이라고 하면되고 동그라미 2번에 1를 넣으려면 b[0]=1를 빨간동그라미 1,2에 들어 있는 값을 3에 넣으려면 b[3]=a[1]+b[0]하면 된다. 상위 작업들을 프로그램 구현후 배열의 내용을 모두 출력해 보자.



왼쪽 이미지는 다음과 같은 int형 배열 코드 int arr1[] = {1,5,3}; 을 메모리에 그린 것이다. int arr1[] = new int[3];로 선언한 다음 일일이 값을 넣은 arr1[0]=1; arr1[1]=5; arr1[2]=3과 결과물은 동일하다.

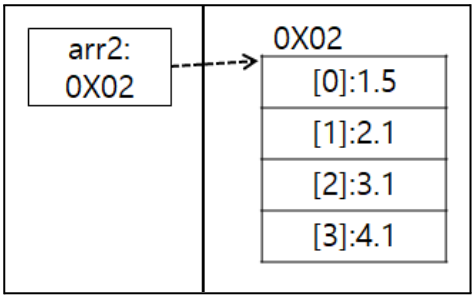
arr1 = {1,5,3}; 과 같은 초기화 방법은 중간에 사용할 수 없고 최초 배열을 선언할 때 만 사용할 수 있다.

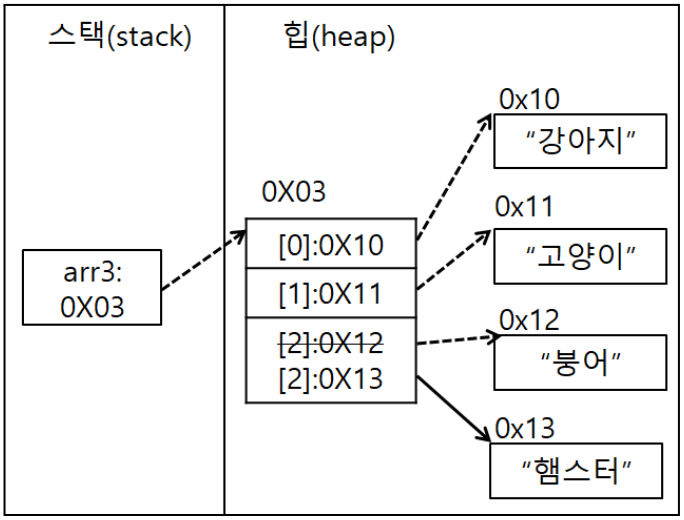
배열도 참조 데이터여서 실제 데이터는 힙에 있다.

아래 왼쪽 이미지는 다음과 같은 double형 배열 코드 double arr2[] = {1.1,2.1,3.1,4.1};를 메모리에 그린 것 이다. double은 기본 자료형 이어서 배열로

할당된 위치에 직접 실수 값이 들어 간다.

아래 오른쪽 이미지는 다음과 같은 String 클래스형 배열 코드 String arr3[] = {"강아지","고양이","붕어"}; arr3[2]= “햄스터”; 를 메모리에 그린 것이다. 배열에 들어 있는 데이터가 참조 데이터여서 arr3배열 변수에는 문자열 주소를 저장할 수 있는 배열의 시작 주소가 들어 있고, 배열안에는 문자열 주소를 저장할 수 있는 공간이 3개 잡히고, 배열안 데이터의 문자열 주소를 찾아가면 실제 문자열에 해당하는 데이터가 있다.





상위 글들을 이미지와 비교해 보며 하나하나 따라가보자.

arr3[2]= “햄스터”;를 통해서 arr[2]의 주소가 0x12 (“붕어”)에서 “햄스터” 문자열의 주소 0x13으로 변경 되었다. 0x12의 붕어 문자열이 햄스터로 변경된 것이 아니고 0x13의 새로 메모리에 “햄스터” 문자열을 저장한 후 해당 주소(0x13)를 arr[2]에 넣었다.

문자열 클래스는 한번 힙에 생성하면 중간에 해당 주소에 들어있는 문자열 자체를 변경할 수 있는 방법이 없다. 변경하고 싶다면 문자열을 새로 힙에 생성해서 넣어줘야 한다. 붕어를 수정해서 햄스타로 변경하는 방법이 없어서 기존 붕어는 메모리에 남겨 두고 햄스터를 메모리에 새로 만들어서 문자열 변수에 새로 생성된 햄스터의 주소를 넣는다.

다음 이미지는 Human 객체 배열을 가지고 작업한 코드이다. 코드 확인후 아래 이미지에서 메모리에 어떻게 잡혀 있는지 확인해 보자. 참조 객체는 18번 처럼 선언 하거나 19~23라인 처럼 초기화 하여 생성 할 수 있다.

주의할 점은 참조 자료형의 배열은 new 연산자에 위해서 힙 메모리에 저장공간이 생기면 실제 객체를 저장할 수 있는 공간이 만들어 지는 것이 아니라 참조 자료형의 주소를 담을 수 있는 공간 여러개가 배열안에 생성되는 것이어서 일일이 실제 데이터 저장 공간을 할당하여야 생성된 주소를 배열에 넣어 주어야 한다.

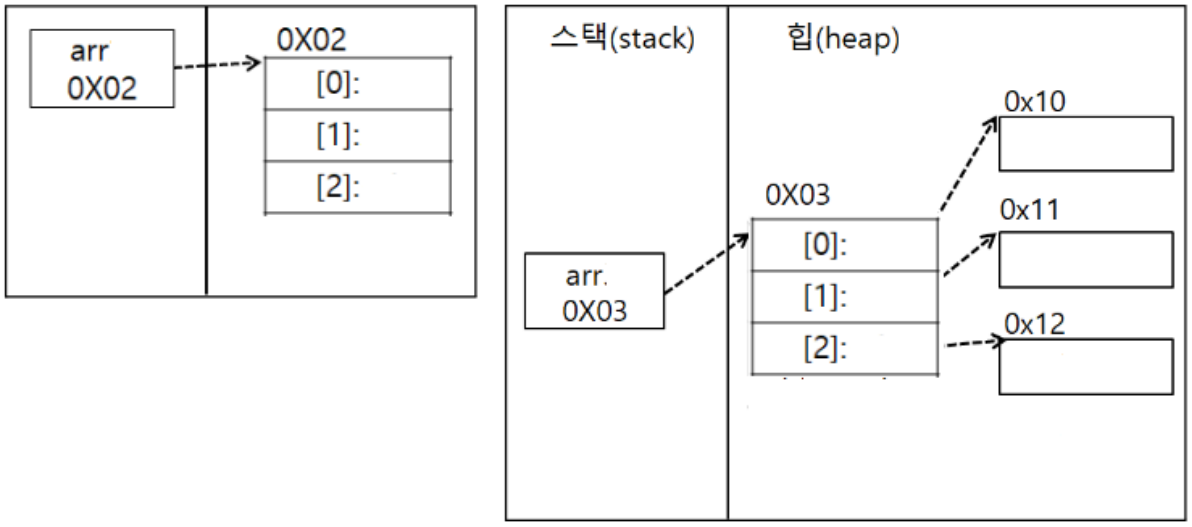
```
18 Human arr4[]=new Human[3];
19 //Human arr4[]= {
20 //   new Human(),
21 //   new Human(),
22 //   new Human()
23 //};
24 arr4[0]=new Human( );
25 arr4[0].name="홍길동"; arr4[0].age=15;
   arr4[0].height=154.1;
26 arr4[1]=new Human();
27 arr4[1].name="홍길남";
28 arr4[1].age=25;
29 arr4[1].height=157.1;
30
31 arr4[2]=arr4[0];
32
33 System.out.println(Arrays.toString(arr4));
34 //[com.human.dto.Human@6d06d69c, com.human
35 System.out.println(arr4[0]);
36
37 arr4[1].name="홍길동";
38 arr4[1].age=15;
39 arr4[1].height=154.1;
40
41 System.out.println(arr4[0]==arr4[1]);
42 System.out.println(arr4[0]==arr4[2]);
```

18라인 처럼 기술하면 Human 인스턴스 주소들을 저장할 수 있는 배열의 공간이 생기고 실제 저장 할 Human 인스턴스의 공간은 생성되지 않는다.

18라인 다음에 arr4[0].name 을 입력하면 null포인트 에러가 발생한다. "null 포인터"는 프로그래밍에서 사용되는 용어로, 메모리상에 특정 객체나 데이터를 가리키는 포인터 변수가 아무 것도 가리키지 않을 때를 나타난다. 19~23라인을 확인해 보면 배열 초기화시 new 연산자로 실제 데이터 저장공간을 생성 하였다. 18라인을 주석하고 19~23 라인의 주석을 해제 하고 arr4[0].name= “홍” 이라고 입력하면 null포인트 에러 없이 동작 한다. 18라인의 경우 실제 저장공 간이 생성되어 있지 않았고, 19~23라인의 경우 new 연산자를 이용해서 실제 저장공간을 만들어 배열을 초기화 하였기 때문이다.

현 코드에서 배열에 실제 저장 공간을 할당 하려면 24, 26, 31라인 처럼 힙에 저장 공간 주소를 배열 안에 넣을 수 있다. 이렇게 하면 배열 안의 주소를 통해서 힙 메모리에 생성된 실제 데이터 저장 공간에 값을 저장 할 수 있다.

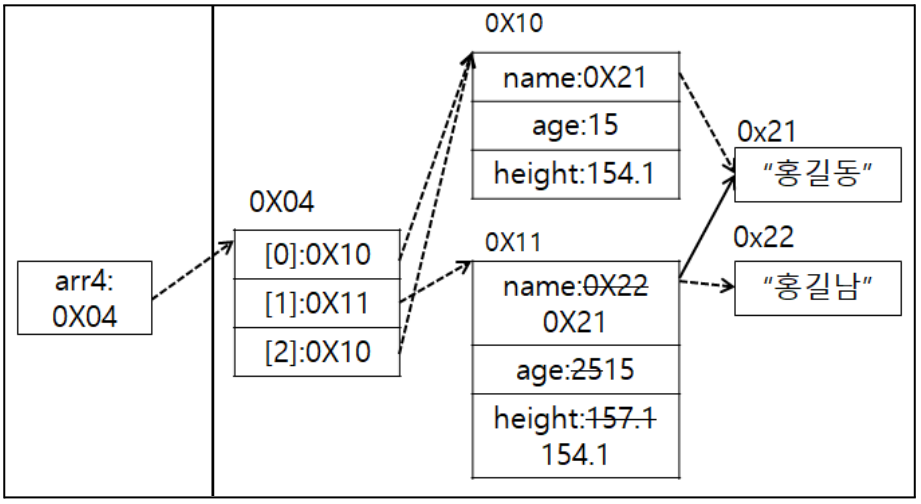
18라인과 19~23라인을 각각 메모리에 그리고 차이점을 확인해 보자. 왼쪽이 18라인의 경우이고 오른쪽이 19~23라인의 경우이다.



25번과 27~29번은 인스턴스 생성 후 값에 접근하는 방법이다.

문자열 객체 주소 데이터로 이루어진 배열 변수에 [index]를 붙이면 해당 배열 index 위치의 주소에 접근 할 수 있다. 현재 배열에 들어 있는 값이 주소이므로 arr4[0]은 주소가 되고 주소에 .를 붙이면(arr4[0].name) 해당 주소에 위치한 인스턴스 필드 값에(클래스 내부에 선언한 변수) 접근 하라는 의미이다.

상위 코드는 다음 이미지 처럼 메모리에 잡힌다.



18번 라인에서 0x04 부분이 만들어져 arr4에 들어간다.

24번 라인에서 0x10 부분이 만들어 져 arr4[0]에 들어 갔다.

26~29라인에서 0x11 부분이 만들어 졌 arr4[1]에 들어 갔다.

31라인에서 arr4[2]에 arr4[0]의 값을 넣어서 둘다 같은 객체 0x10 데이터를 가지게 되었다.

33라인에서 arr4배열의 내용을 찍어 보았다. 배열안에 들어 있는 모든 Human객체가 출력 된다. 들어있는 객체의 주소가 나온다. 35번 라인에서 arr4[0]에 들어있는 Human객체가 출력 된다.

41번 라인 arr4[0]==arr4[1]은 0x10==0x01 으로 들어 있는 주소 데이터가 다르니 false가 출력 된다.

42번 라인 arr4[0]==arr4[2]은 0x10==0x10 으로 들어 있는 주소가 같으니 true가 출력 된다.

arr4[0], arr4[1], arr4[2]에 들어 있는 값을 비교하려면 .를 찍어서 해당 객체가 들어있는 값에 접근해야 한다. arr4[0].age==arr4[1].age, arr4[0].age==arr4[2].age

다음은 야구게임 전광판이다. 데이터들을 다양한 자료형을 선언하여 처리해 보자.

스코어보드 2019.07.29 02

	1	2	3	4	5	6	7	8	9	R
탬파베이	0	0	0	0	1	3	2	3	1	10
토론토	0	2	2	0	4	1	0	0	0	9

상위 정보를 표현하는데 필요한 자료들이 어떠한 것들이 있는지 기술해 보자. 일단 탬파베이, 토론토는 야구 팀이다 두팀의 1~9회까지 점수를 저장할 배열 2개와 총합을 저장할 변수 2개가 있으면 야구점수 관련된 모든 데이터를 저장할 수 있다. 물론 배열 하나에 모든 정보를 기록 할 수 있다. 0~9까지는 탬파베이 점수 10~19까지는 토론토 점수 해도 되지만 이렇게 된다면 배열을 사용하는 것이 불편해지므로 사용하기 편한 형태로는 두팀의 점수 저장 배열2개와 총합 변수 2개로 하여 선언하였다. team1[], team2[], team1TotalScoure=0, team2TotalScore=0 이해하기 쉬운 형태의 변수명으로 선언해 보았다. 배열 team1에 탬파베이 점수값을 넣어보자. 코드 줄수를 줄이기 위해서 배열을 선언하면 기본 값이 0 이라고 생각하고 진행하자. 5회에 1점을 얻었다 team1 배열에 이 정보를 넣으려면 team1배열의 4번째 인덱스에 1를 넣으면 된다. team1[4]=1 6회에 3점을 얻었다. team1[5]=3 … 이런 식으로 진행해 나가면 된다. 토론토의 점수는 다른 배열 team2에 넣으면 된다. team2[1]=2, team2[2]=2 이런 식으로 넣으면 된다. 두팀의 총 취득한 점수를 어떻게 구할 것인지 생각해보자. 첫번째 팀의 경우 다음과 같이 team1의 인덱스 0 부터 8까지 모두 저장하면 된다.

team1TotalScoure=team1[0]+team1[1]+team1[2]+team1[3]+team1[4]+team1[5]+team1[6]+team1[7]+team1[8] 이렇게 하면 팀1의 득점한 총 점수를 출력할 수 있을 것이다.

팀2의 득점한 총 점수를 얻는 방법은 다음과 같다.

team2TotalScoure=team2[0]+team2[1]+team2[2]+team2[3]+team2[4]+team2[5]+team2[6]+team2[7]+team2[8] 이렇게 하면 팀1의 득점한 총 점수를 출력할 수 있을 것이다.
프로그램으로 구현해서 상위 전광판 처럼 출력해보자.

다음 문제를 풀어보자.

1. 배열에 53,6,85,3,5를 넣은 다음 배열의 내용을 화면에 출력해 보자.

2. 배열 a에 1,2,3 을 넣은 후 배열 내의 모든 값에 2를 더한 값인 3,4,5로 변경한 다음에 배열의 내용을 화면에 인덱스 순서대로 출력해 보고 인덱스 역순으로 출력해 보자.

3. 배열 a[]={12,1,53,6,85,3}를 만든 다음에 배열의 모든 내용을 더한 값을 sum에 저장하여 출력하는 코드를 만들어 보자.

4. a[]={12,1,53,6,85,3}에서 a배열에 들어있는 값중 짝수만 화면에 출력하시오.

5. a[]={12,1,53,6,85,3}에서 인덱스가 홀수인 배열의 값을 출력하시오.

6. tree클래스를 배열로 선언하여 여러개의 데이터를 넣은후 출력해 보자.