Java Programming

자바 객체지향 프로그래밍

박매일 강사
bitcocom@empas.com

강의영역
Java, Python
DataBase, Data Modeling
Web, MVC Framework
Spring, Spring Boot, JPAReact.js,Vue.js

-주요이력-

한국전력공사 In-House DT 실무자 교육
한양대학교 ERICA 자바 온라인 강의
정보통신산업진흥원 SW융합 채용연수사업
(현)인공지능사관학교 자바, 웹, 스프링,DB 강의
(현)소프트웨어마이스터고 소프트웨어공학 교육
(현)인프런 자바,웹,스프링,IoT 온라인 콘텐츠 제공
(현)패스트캠퍼스 자바 온라인 콘텐츠 제공

1주차	O.T (5분)	본강의 - 1 (40~50분)	본강의 - 2 (40~50분)	본강의 - 3 (40~50분)	질의응답 (5개 내외)

오리엔테이션(OT)

Java Programming

강의목표

자바 프로그래밍을 통해 **객체지향 프로그래밍 기법**을 이해하고
자바 언어의 기본 원리 및 **문법을 학습**하여 **논리적인 사고력**을
바탕으로 자바언어에 자신감을 부여하기 위한 강의.

개발환경 및 개발도구

- Windows OS
- JDK 1.8 이상 버전 사용
- IntelliJ IDE

오리엔테이션(OT)

Java Programming

강의 커리큘럼

1주차 1. 관계를 이해하라.(relationship)

2주차 2. 클래스를 이해하라.(class)

3주차 3. 상속을 이해하라(inheritance)

4주차 4. 다형성을 활용하라(polymorphism)

5주차 5. API를 활용하라(applications programming interface)

1. 관계를 이해하라.(relationship)

- 1) 자바 개발 환경 만들기
- 2) 프로그래밍 3대 요소를 통한 관계를 이해하기
- 3) 변수와 배열의 관계를 이해하기
- 4) 변수와 메서드의 관계를 이해하기
- 5) JVM의 메모리 모델을 이해하기
- 6) Q&A 질의응답

1주차 | O.T (5분) | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

1) 자바 개발 환경 만들기(IntelliJ 설치 및 구동)

1. Java SE 개발환경 구축(Java™ Platform, Standard Edition) - ①

- IntelliJ IDE 다운로드 및 설치

1. https://www.jetbrains.com/ko-kr/idea/download/?section=windows 접속
2. IntelliJ IDEA Community Edition 다운로드
3. 다운로드 받은 파일을 설치

1주차 | O.T (5분) | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

2) 프로그래밍 3대 요소를 통한 관계를 이해하기

1. 변수, 자료형, 할당

1. 변수(Variable)
   - 데이터를 저장할 메모리 공간의 **이름(symbol)** **연결**
2. 자료형(Data Type)
   - 변수의 **크기**와 변수에 저장될 데이터의 **종류**를 결정하는 것
3. 할당(Assign)
   - 변수에 값을 저장(대입,할당)하는 것

자료형(DataType)

기본자료형(PDT) : 컴파일러에서 기본적으로 제공해주는 자료형

종류	자료형	크기(byte)	예시
정수	short, int, long	2, 4, 8	10, 20
실수	float, double	4, 8	23.4f, 34.567
문자	char	2	'A', 'a'
불	boolean	1	true(참), false(거짓)

VS

class

새로운 자료형을 만드는 도구 ?
(Modeling 도구)

BookDTO
MemberVO
String

Object

사용자정의자료형(UDDT) : **객체** 자료형(Object DataType)
- 필요에 의해서 **새롭게 만들어** 사용하는 자료형
- 만드는 도구, 설계하는 도구, 모델링하는 도구 가 필요하다. : **class**

종류	자료형	예시
책	BookDTO	자바의정석(제목, 가격, 출판사)
회원	MemberVO	김길동(이름, 주소, 전화번호)
문자열	**String**	"APPLE"

Java API
java.lang.*

책
객체(Object)

=

BookDTO

제목	가격	출판사	페이지

→ **class**

class를 이용하여 객체를 설계

```
public class BookDTO{
  public String title;
  public int price;
  public String company;
  public int page;
}
```

우리가 만드는 **객체**는 프로젝트에 따라 다양하므로 **class**로 언제든지 **만들어 사용**하면 된다 !!!

2. 변수선언과 할당

1. **변수(Variable)**
   - 데이터를 저장할 메모리 공간의 이름(symbol)
2. **자료형(Data Type)**
   - 변수의 **크기**와 변수에 저장될 데이터의 **종류**를 결정하는 것

→ **변수선언**

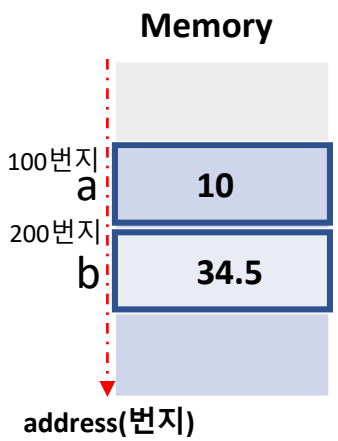
메모리에 변수(기억공간)를 만드는 것
DataType + Variable
변수가 선언되면 **ST(변수테이블)**에 등록이 된다.

```
int a;
float b;

a=10;
b=34.5f;
```

Symbol Table(변수목록표)

변수이름(key)	번지(value)
a	100번지
b	200번지

symbol table을 거쳐서 memory에 접근
symbol table에 변수가 없으면 can not find symbol 에러



※ Symbol Table(변수목록표)
변수가 기억공간을 할당 받으면 변수의 번지가 등록되는 테이블

3. **할당, 대입(Assign, =)**
   - 변수에 값을 대입하는 것

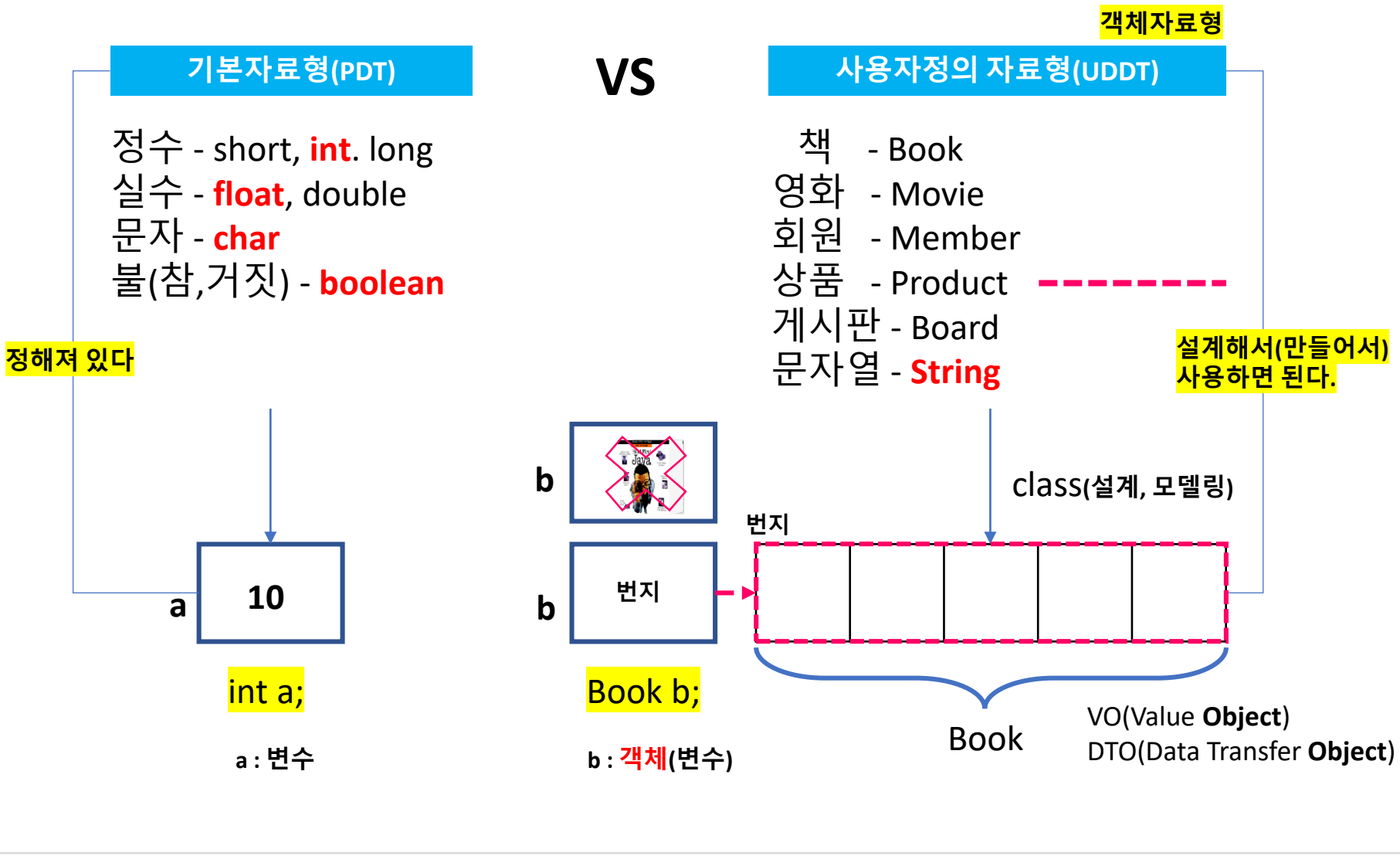
= 해석에 주의

L-Value = R-Value;

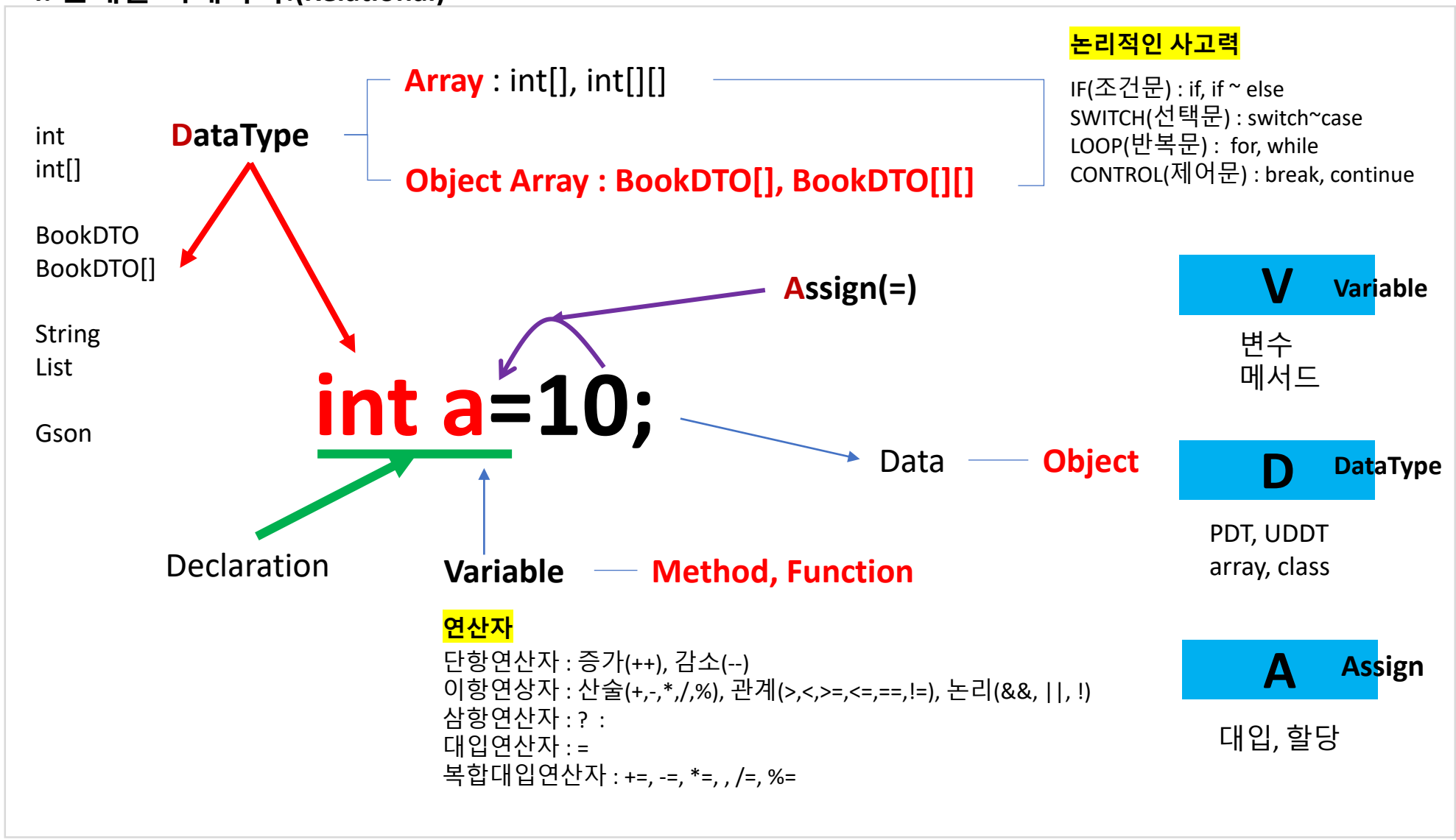
변수 = **값, 변수, 수식, 메서드 호출 문**

```
a=10;
a=b;
a=b+20;
a=sum(b, c);
```

3. 자료형을 이해하라


객체자료형
기본자료형(PDT)
VS
사용자정의 자료형(UDDT)
정수 - short, int. long
실수 - float, double
문자 - char
불(참,거짓) - boolean
책 - Book
영화 - Movie
회원 - Member
상품 - Product
게시판 - Board
문자열 - String
정해져 있다
설계해서(만들어서) 사용하면 된다.
class(설계, 모델링)
번지
a
10
int a;
a : 변수
b
번지
Book b;
b : 객체(변수)
Book
VO(Value Object)
DTO(Data Transfer Object)

4. 관계를 이해하라.(Relational)

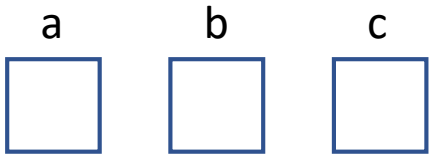


3) 변수와 배열의 관계를 이해하기

1. 변수와 배열(Array)

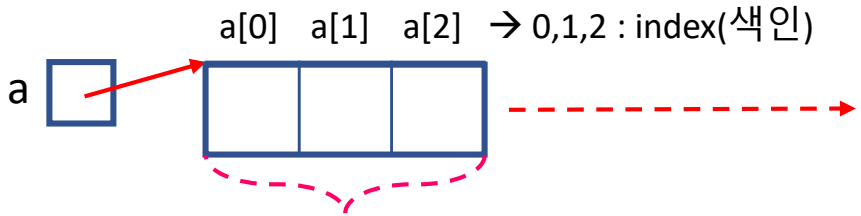
Q) 3개의 정수를 저장하기 위해서 변수 3개를 만드는 방법

변수를 개별적(불연속)으로 만드는 방법




```
int a, b, c;
a=10;
b=20;
c=30;
```

변수를 연속적으로 만드는 방법(Array, 객체)



[ ] 배열기호
[ ] : 1차원
[ ][ ] : 2차원

```
int[] a;
a=new int[3];
    or
int[] a=new int[3];
```

```
a[0]=10;
a[1]=20;
a[2]=30;
a.length=>3
```

배열의 길이?

→데이터 처리가 복잡하다.
→데이터 이동이 어렵다.
→데이터를 한 개만 저장가능 하다.

Array(배열) 특징

→많은 수의 변수를 만들기가 용이하다.
→기억공간 접근이 쉽다(반복 문 사용 가능)
→데이터 이동이 쉽다.
(데이터를 하나의 형태로 담아서 이동 가능)

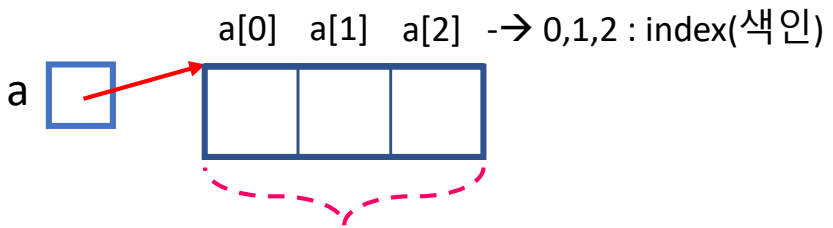
class ← → 서로 다른 데이터 타입(이질적인 구조, 객체)을 저장 할 수 없다
(Array단점)

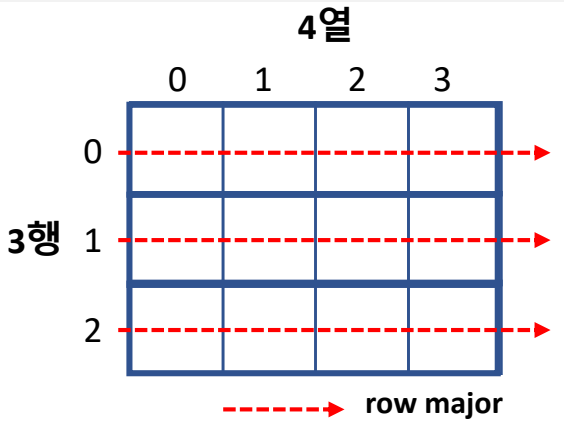
Array(배열) **동일한 타입의** 데이터를 여러 개 저장하기위한 연속적인 메모리 구조 – Array(배열)

2. 1차원, 2차원 배열(Array)

1차원 Array

```
int[] a;
a=new int[3];
      or
int[] a=new int[3];
```



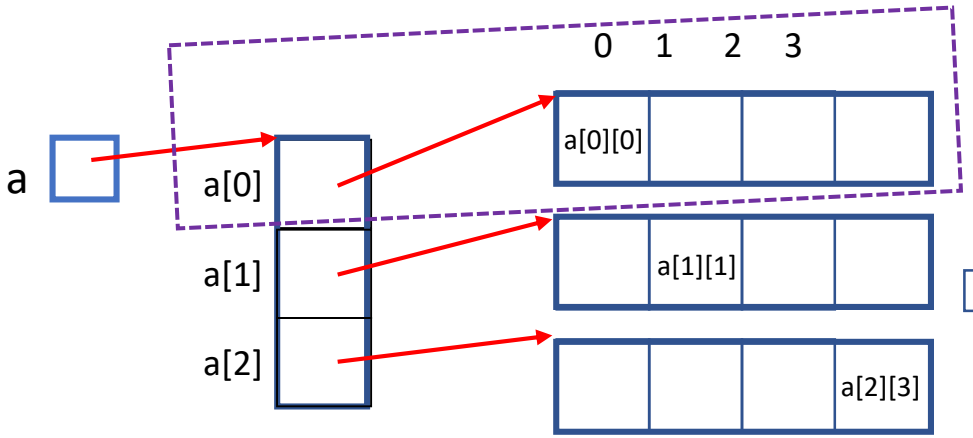


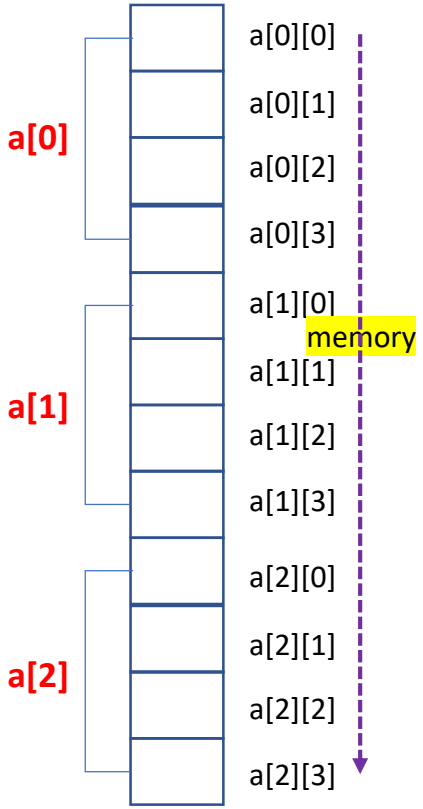
2차원 Array

[3][4] =3행4열

```
int[][] a;
a=new int[3][4];
      or
int[] a=new int[3][4];
```

1차원 배열이 여러 개로 된 형태





a.length=>3
a[0].length=>4
a[1].length=>4
a[2].length=>4

해석: a라는 아파트에 3개(a[0],a[1],a[2]) 동이 있고 각 동은 3층이다.

[가변길이 배열]
int[] a=new int[3][];
a[0]=new int[3]; a[1]=new int[4]; a[2]=new int[5];

4) 변수와 메서드의 관계를 이해하기

1. 변수와 메서드(method)

변수(variable)

변수(Variable) : 데이터를 **한 개 만(한 개의 형태)**저장 기능하다.
→저장만 한다.

메서드(method)→ 메서드 이름이 변수 역할을 한다

메서드(method) : 동작을 한 후에 데이터를 **한 개 만** 만들어 낸다.
→동작 후 저장한다.

DataType
int a=10;

return DataType
int sum=a+b;

변수와 메서드는
결론적으로 데이터를 한 개만
저장하므로 비슷하다.

함축적인 표현

메서드에서 리턴 하는 값을 메서드 이름에 저장한다.
(메서드 이름이 **변수 역할**을 한다)

method 호출문

```
int v=sum(10,30);
```

method Call(호출)

sum=40

return

method 선언문

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

외부로부터 데이터를 받을 때

method 선언
(정의 부 + 구현 부)

```
접근제어자 리턴타입 메서드이름(매개변수 리스트){
    // 처리부분 ....
    // 리턴여부 →return
}
```

2. 메서드의 매개변수 전달기법(parameter passing)

Call By Value(값 전달 기법) 기억공간 개별

```
int a=10; int b=20;
int v=sum(a, b);
```

a,b : value(값)

Call By Value

Call By Reference(번지전달 기법) 기억공간 공유

```
int[] arr={10, 20, 30};
int v=sum(arr);
```

arr : reference(번지)

Call By Reference

method 정의 부

```
public int sum(int a, int b){
    int v=a+b;
    return v;
}
```

```
public int sum(int[] a){
    int v=0;
    for(int i=0;i<a.length;i++){
      v+=a[i];
    }
    return v;
}
```

5) JVM의 메모리 모델을 이해하기

1. JVM Memory Model 1

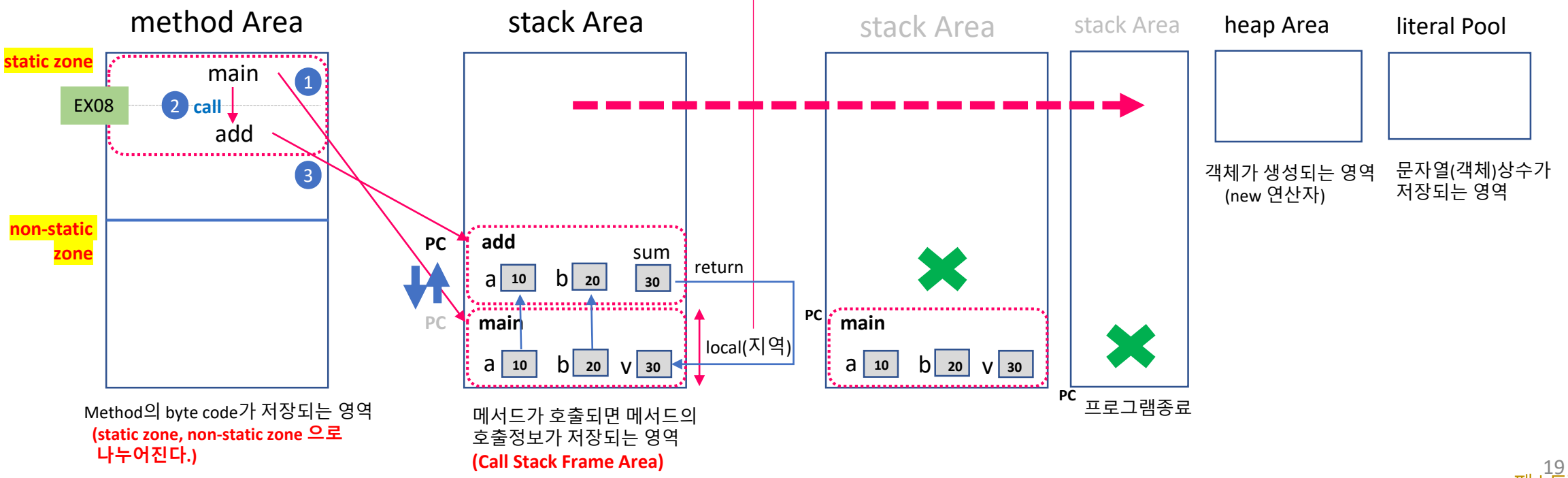
[JVM이 EX08 class(실행클래스)를 실행하는 절차]

1. 해당클래스를 현재 디렉토리에서 찾는다.
2. 찾으면 클래스 내부에 있는 **static 키워드**가 있는 메서드를 메모리로 로딩 한다.
   - method Area의 static zone에 로딩 한다. **main() , add() method**
3. static zone에서 main() 메소드를 실행한다(호출, 시작)
   - main() method가 호출되면 main() method의 호출정보가 Stack Area에 들어간다(push)
   - 프로그램이 시작되는 부분이다.(PC의 위치가 현재 동작되고 있는 메서드다.)
4. Stack Area가 비어 있으면 프로그램이 종료된 것이다.

```
public class EX08{
  public static void main(String[] args){
    int a=10; int b=20;
    int v=add(a, b);
    System.out.println(v);
  }
  public static int add(int a, int b){
    int sum= a+b;
    return sum;
  }
}
```

local(지역)

add() call



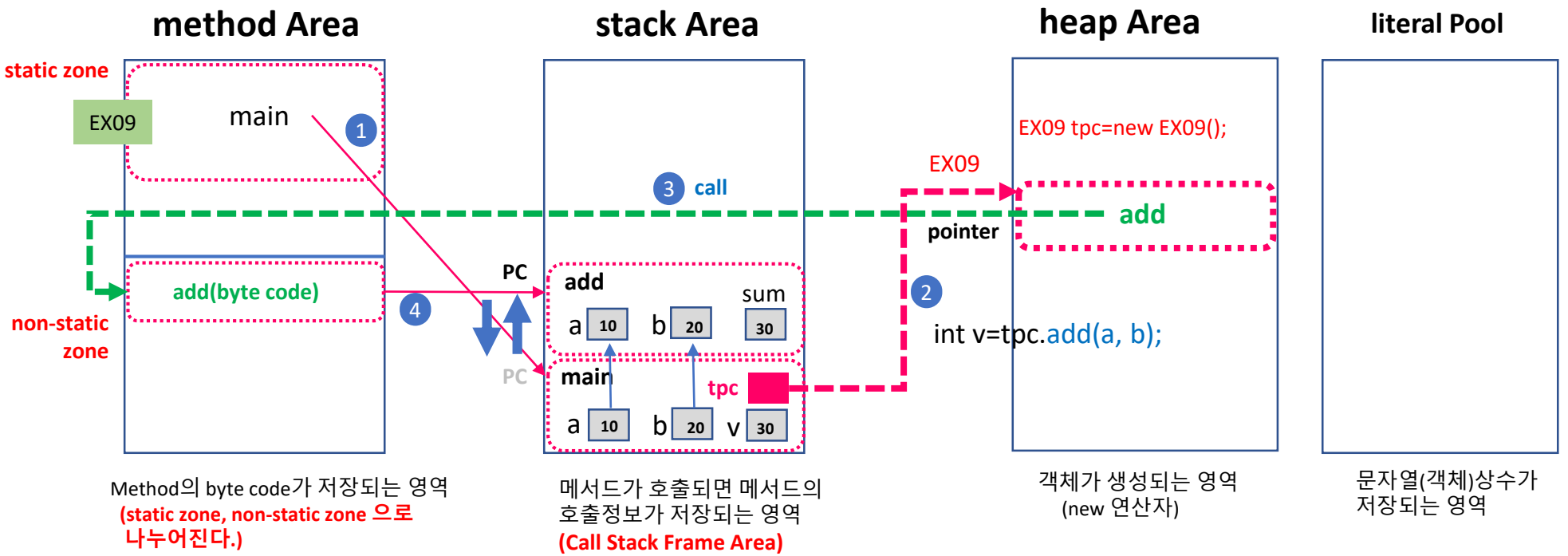
2. JVM Memory Model 2

[JVM이 EX09 class(실행클래스)를 실행하는 절차]

1. 해당클래스를 현재 디렉토리에서 찾는다.
2. 찾으면 클래스 내부에 있는 **static 키워드**가 있는 메서드를 메모리로 로딩 한다.
   - method Area의 static zone에 로딩 한다. main() method
3. static zone에서 main() 메소드를 실행한다(호출, 시작)
   - main() method가 호출되면 main() method의 호출정보가 Stack Area에 들어간다(push)
   - 프로그램이 시작되는 부분이다.(PC의 위치가 현재 동작되고 있는 메서드다.)
4. Stack Area가 비어 있으면 프로그램이 종료된 것이다.

```
public class EX09{
   public static void main(String[] args){
      int a=10; int b=20;
      EX09 tpc=new EX09();
      int v=tpc.add(a, b);
      System.out.println(v);
   }
   public int add(int a, int b){
      int sum= a+b;
      return sum;
   }
}
```

add() call



1주차	O.T (5분)	본강의 - 1 (40~50분)	본강의 - 2 (40~50분)	본강의 - 3 (40~50분)	질의응답 (5개 내외)

6) Q&A 질의응답

질의응답(Q&A)

Q. Java 개발 툴이 이클립스만 있나요?

- Eclipse IDE , STS, eGovFrame(전자정부표준프레임워크)
- IntelliJ IDEA
- Visual Studio Code

Q. Java로 어떤 종류의 프로그램을 만들 수 있나요?

- 응용프로그래밍(**Desktop** 응용 프로그램)
- 웹 응용프로그램(**Web** 기반 기업 업무용 ERP 시스템)
- 안드로이드 앱 응용프로그램(**Mobile**기반 응용 프로그램)

Q. 다른 객체지향 프로그래밍 언어가 있나요?

- **Java, C++, C#, Python**, PHP, Ruby, Object-C는 객체지향 프로그래밍을 지원한다

Q. 자바 언어의 장점이 무엇인가요?

- 운영체제(OS)에 **독립적**인 프로그램을 개발 할 수 있다.(플랫폼에 독립적)

2. 클래스를 이해하라.(class)

- 1) **클래스(객체)**란 무엇인가?
- 2) 객체가 **메모리에 생성되는 과정**을 이해하기
- 3) class, object, instance의 상호관계
- 4) **잘 설계된 클래스** 이란 무엇인가(Model : DTO,VO)
- 5) 메서드의 **오버로딩(overload)**이란?
- 6) Q&A 질의응답

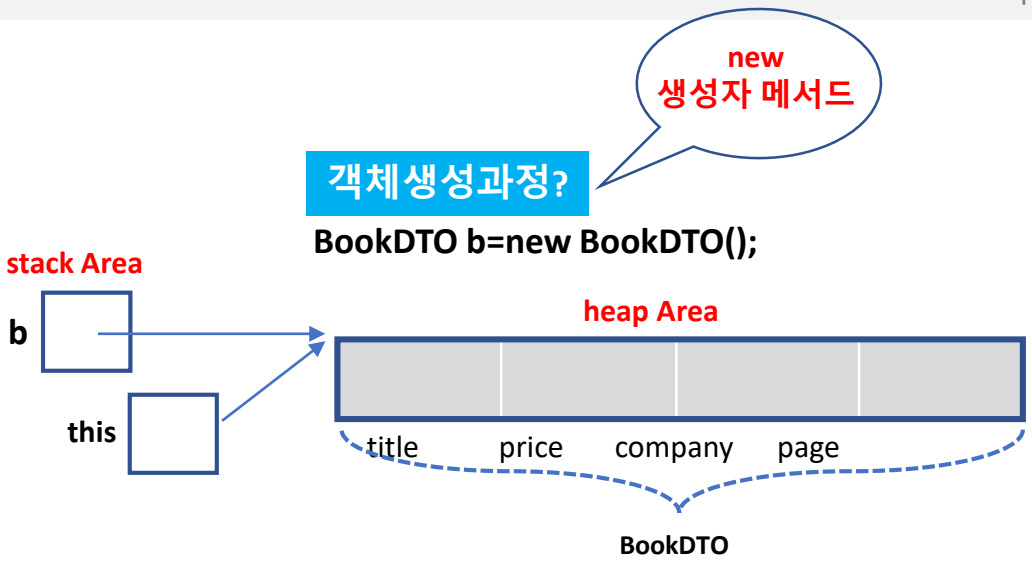
2주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

1) 클래스(객체)란 무엇인가?

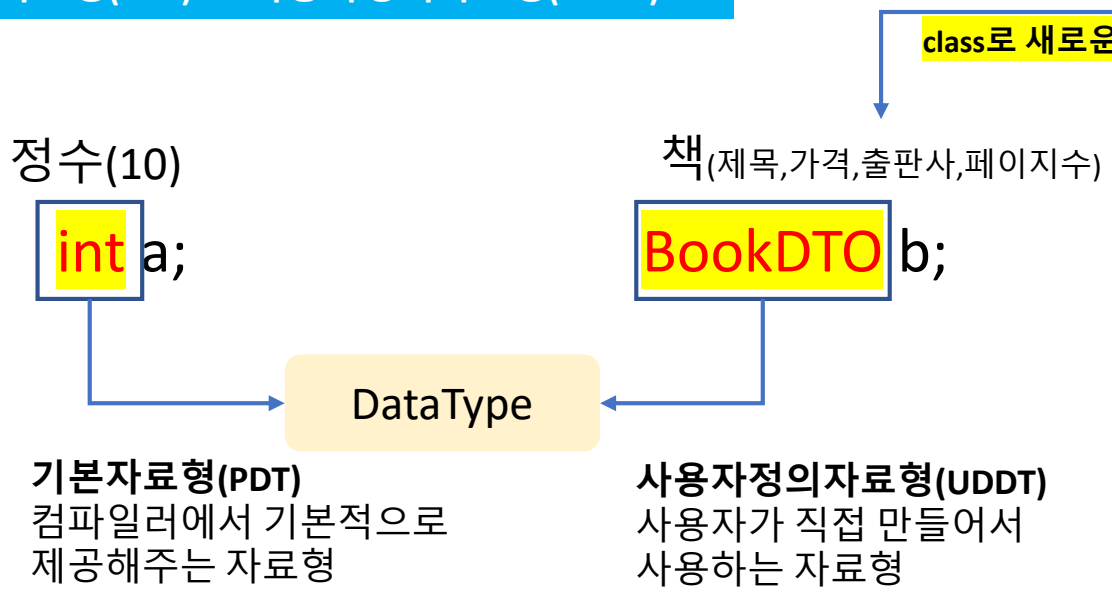
1. 기본자료형(PDT) VS 사용자정의자료형(UDDT)

이것만 알면 !

1. **기본자료형(PDT) VS 사용자정의자료형(UDDT)**
   - DataType을 확실히 이해하자
2. **class, object, instance 상호관계**
   - 객체생성과정(new 연산자, 생성자 메서드, this)
3. **잘 설계된 클래스**
   - DTO(VO), DAO, Utility



기본자료형(PDT) VS 사용자정의자료형(UDDT)



class로 새로운 자료형을 만든다.

```
public class BookDTO {
   public String title;
   public int price;
   public String company;
   public int page;

}
```

```
public BookDTO(){
   super();
}
```

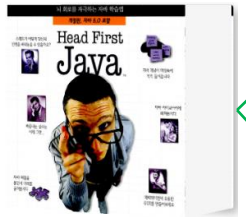
default constructor
(기본생성자)

2주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

2) 객체가 **메모리에 생성되는 과정**을 이해하기

1. 객체생성과정

객체생성과정 BookVO b=new BookVO(); // new 연산자와 생성자 메서드 호출



Object(책)

1. 상태정보(멤버변수) : attribute, property, member
: **제목**, **출판사**, ~~저자~~, **가격**, **페이지수**, ~~ISBN~~, ~~이미지~~, ~~두께~~, ~~무게~~, ~~재질~~.....

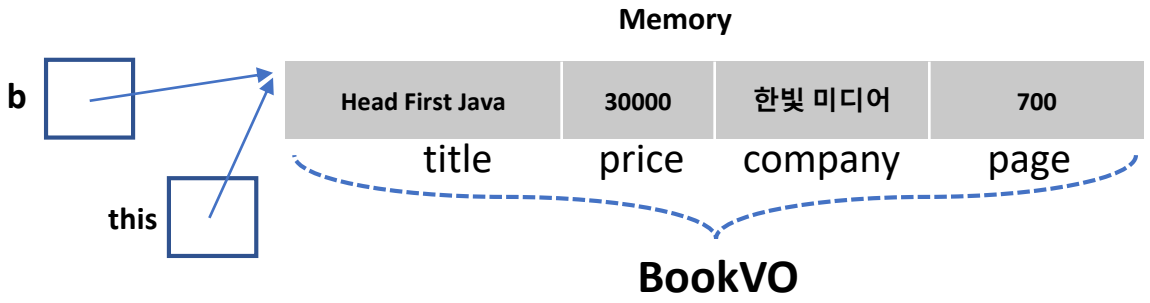
Modeling 필요한 속성만 뽑아내는 과정

제목, **가격**, **출판사**, **페이지수 -> Head First Java, 30000, 한빛 미디어, 700**

2. 행위정보(멤버메서드) : 동작(method), 기능(function)

```
public class BookVO {
  public String title;
  public int price;
  public String company;
  public int page;
}
```

객체생성



Point

b.

title
price
company
page

.(dot)연산자
접근,참조연산자

public member만 접근가능

객체생성 후 접근 방법 .(dot) 연산자

```
b.title = "Head First Java";
b.price=30000;
b.company="한빛 미디어";
b.page=700;
```

객체의 상태정보를 직접 접근하면
잘 못된 데이터가 저장될 수 있다.
설계를 잘 해야 된다.

정보은닉 필요
Information hiding(private)

2. 생성자 메서드(Constructor)

이것만 알면 !

1. 객체를 생성할 때 사용되는 메서드
2. 객체 생성 후 객체의 초기화를 하는 역할 수행
3. **특징**
   - **클래스이름과 동일한 메서드**
   - **메서드의 return type이 없다(void 아님)**
   - **public 접근 권한을 가진다.(단, private 생성자도 있음)**
   - **생성자가 없을 때는 기본 생성자가 만들어 진다.**

생성자 메서드 중복정의(Overloading)

BookVO b=new BookVO();

BookVO b=new BookVO("자바",20000,"길벗",790);

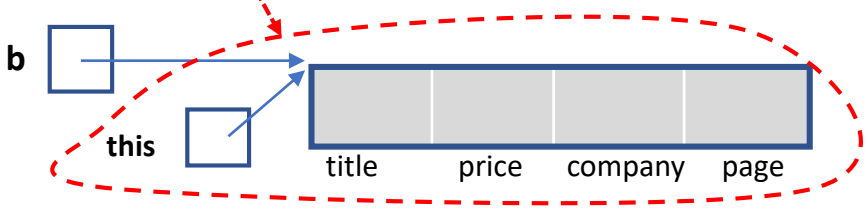
생성자 메서드를 활용하여 객체를 적절하게 초기화 하라. 중복정의(Overloading)

```
public class BookVO {
  private String title;
  private int price;
  private String company;
  private int page;

  public BookVO(){
     super();
  }
}
```

BookVO b=new BookVO();

BookVO() 호출

default constructor : 초기화 작업 없음
(기본생성자)




```
public class BookVO {
  private String title;
  private int price;
  private String company;
  private int page;

  public BookVO(String title, int price, String company, int page){
     this.title=title;
     this.price=price;
     this.company=company;
     this.page=page;
  }
}
```

BookVO b=new BookDTO("자바",20000,"길벗",790);

초기값

이동

초기화

→ overloading constructor : 초기화를 위해서(중복 정의된 생성자)
→ 생성자를 중복정의 하면 기본생성자는 지동으로 만들어지지 않는다.

3. private 생성자 메서드(Constructor)

→ 객체생성에 관여하는 생성자 메서드가 private 접근제어를 가지면 객체를 생성할 수 없다는 뜻이 된다.
→ 그러므로 객체를 생성하지 않고도 사용가능 해야 된다.(모든 클래스의 멤버가 static 멤버가 되어야 한다.)

→ non-static 멤버인 경우(인스턴스 메서드)

객체생성 후 접근 가능

```
Erica inf=new Erica();  // 생성자가 public인 경우
inf.tpc();
```

→ static 멤버인 경우(클래스 메서드)

객체생성 없이 접근가능(**클래스 이름으로 접근**)

```
Erica.java();
```

클래스를 사용하는 시점에서 static 멤버는 먼저 자동으로 메모리에 로딩이 된다. 그 이후에 호출이 된다

생성자가 private이므로 객체 생성 불가
Erica inf=new Erica(); // X

※ Java API 중에서도 생성자가 private인 클래스도 많이 있다.
System, Math…. System sys=new System(); X
→ 자주 사용하는 객체나, 동작은 static 멤버로 만든다.

```
public class Erica {
  private Erica(){
  }
```

모든 멤버가 static멤버이면 인위적으로 private생성자로 만들어 객체생성을 막을 수도 있다.

```
  // 인스턴스 메서드
  public void tpc(){
    System.out.println("TPC강의 너무 재미있다.");
  }
  // 클래스 메서드
  public static  void java(){
    System.out.println("Java강의 너무 재미있다.");
  }
}
```

어떤 객체에 생성자 메서드가 private이면 모든 멤버는 static가 붙은 멤버가 되어야 한다.

※ static 멤버 접근방법
→ 클래스이름.클래스메서드(static 메서드)

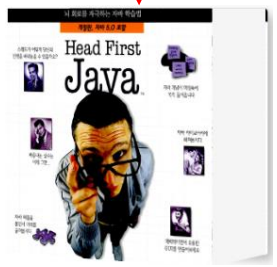
2주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

3) class, object, instance의 상호관계

1. class, object, instance 상호관계

객체생성과정 BookDTO b=new BookDTO(); // new 연산자와 생성자 메서드 호출

현실세계의 객체(책) | class | object | instance



Modeling

```
public class BookDTO {
  public String title;
  public int price;
  public String company;
  public int page;
}
```

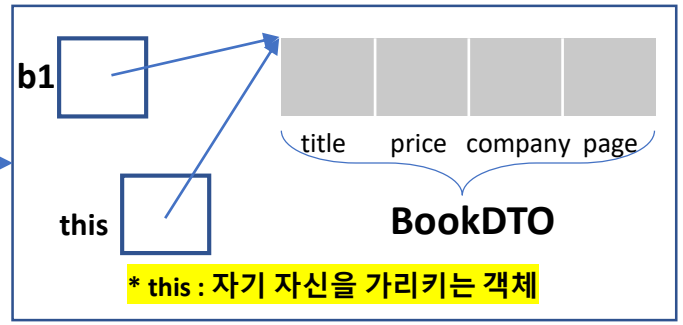
```
public BookDTO(){
    super();
}
```

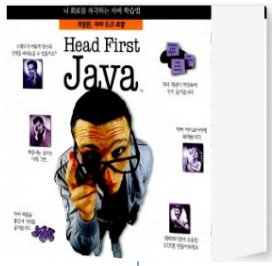
BookDTO **b1**;
BookDTO **b2**;
BookDTO **b3**;

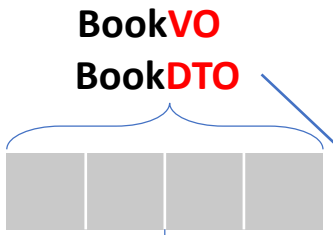
b1=new **BookDTO()**;

실체

Memory(Heap Area)







BookVO
BookDTO

책(Object)

Point

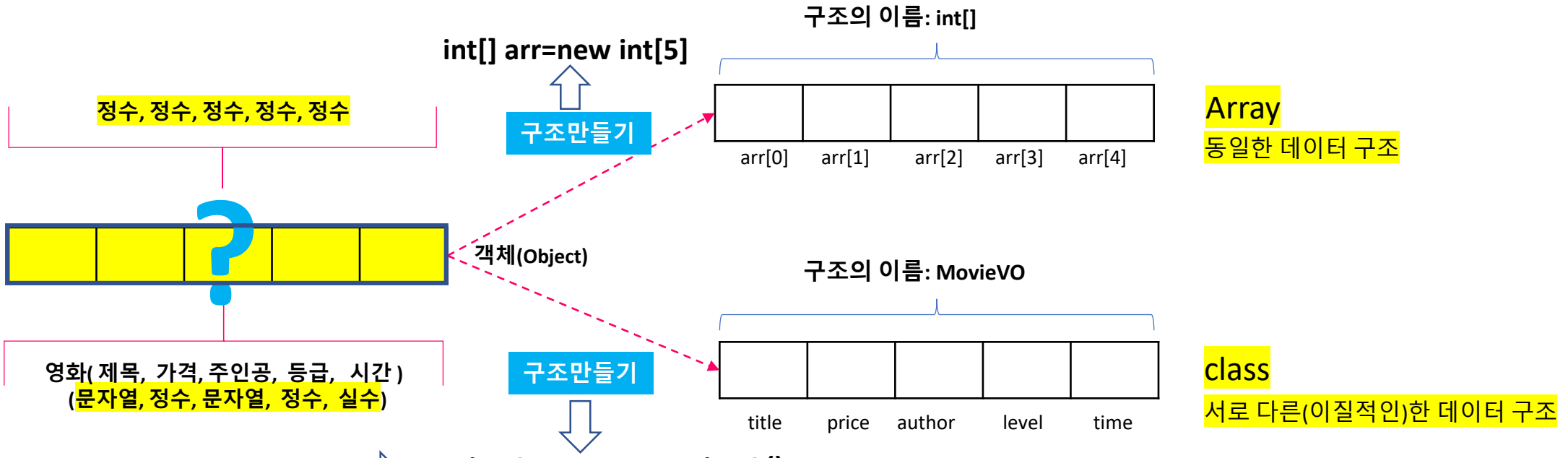
default constructor (기본생성자)

1. 생성자 메서드에서 객체를 **생성**한다.
2. 객체가 생성된 후 **초기화** 역할을 수행한다.

BookDTO : class 로 만든 새로운 자료형 = 바구니 = 이동

* **DTO : Data Transfer Object**
* **VO : Value Object**

2. 배열 vs 클래스의 관계



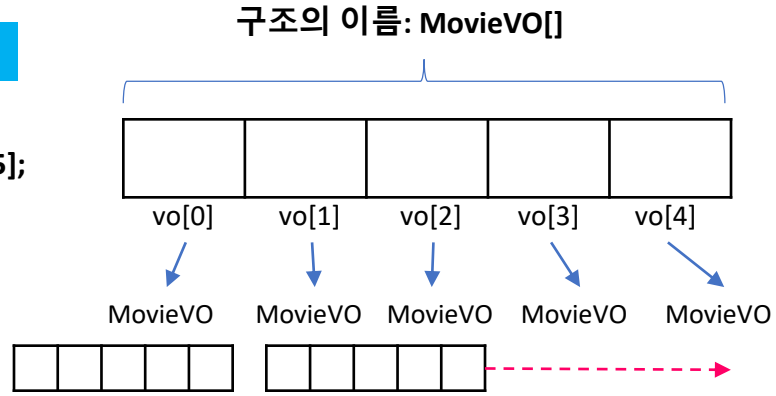

```
public class MovieVO{
    private String title;
    private int price;
    private String author;
    private int level;
    private float time;
}
```

⇨ **MovieVO vo=new MovieVO();**

객체배열(Object array)

// 객체배열

```
MovieVO[] vo=new MovieVO[5];
vo[0]=new MovieVO();
vo[1]=new MovieVO();
vo[2]=new MovieVO();
vo[3]=new MovieVO();
vo[4]=new MovieVO();
```



2주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

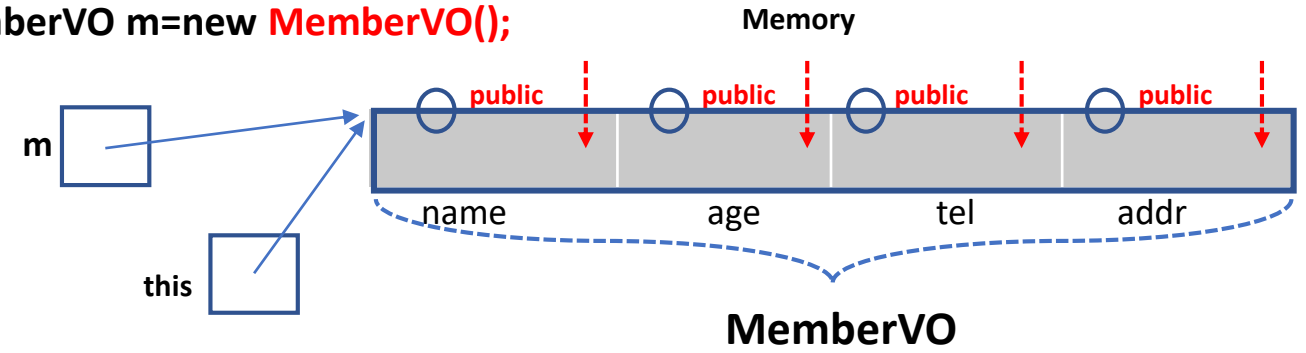
4) 잘 설계된 클래스 이란 무엇인(Model : DTO,VO)

1. 정보은닉(private)

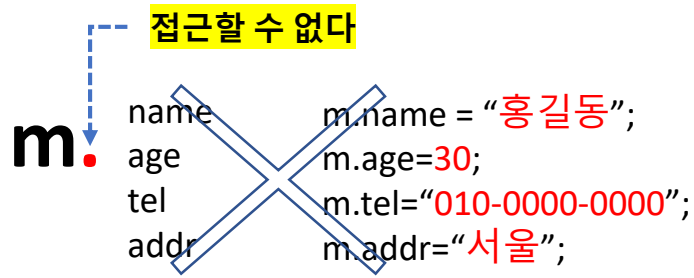
→ 정보은닉(private) : 다른 객체(class)로부터 접근을 막는 것(private)

```
public class MemberVO {
  private String name;
  private int age;
  private String tel;
  private String addr ;

  public MemberVO(){

  }
}
```

MemberVO m=new MemberVO();



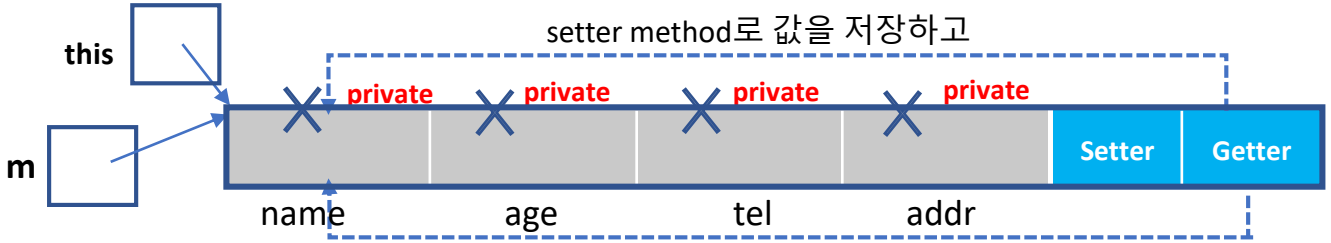
Point



private 멤버변수를 접근할 때
setter, getter method를 활용하라!

setter, getter method

```
public void setName(String name){
   this.name=name;
}
public String getName(){
  return name;
}
```



m.setName("자바");
m.getName();

Getter method로 값을 얻어온다.

2. 잘 설계 된 DTO, VO 클래스 * DTO(Data Transfer Object), VO(Value Object)

```
public class MemberVO {
 private String name;
 private int age;
 public MemberVO() {  }
 public MemberVO(String name, int age) {
     this.name = name;
     this.age = age;
 }
 public String getName() {
     return name;
 }
 public void setName(String name) {
     this.name = name;
 }
 public int getAge() {
     return age;
 }
 public void setAge(int age) {
     this.age = age;
 }
 @Override
 public String toString() {
 return "MemberVO [name=" + name + ", age=" + age + "]";
 }
}
```

1 → private 으로 객체의 상태를 보호한다.
정보은닉(information hiding)

2 → 디폴트 생성자를 명시적으로 만든다.
오버로딩 생성자를 만들어 적절하게 초기화 한다.
- 객체를 생성하는 작업은 생성자 내부에서 JVM이 자동으로 처리한다.

3 → Private으로 만들어진 멤버변수를 접근하기 위해서 setter, getter method를 만든다.

- DI(Dependency Injection : 종속객체 주입)
  - setter method의 역할

객체가 가지고 있는 값 전체를 출력하기위한 toString() method를 재정의 한다.

4

5) 메서드의 오버로딩(overload)이란?

1. 메서드 오버로딩(Method Overloading)

메서드 오버로딩(Method Overloading)

같은 이름의 메소드를 여러 개 가지면서 매개변수의 유형과 개수가 다르도록 하는 기술
-> 메서드의 **signature** 가 다르면 된다(**signature** : 매개변수의 타입, 개수)

```
public class Test{
  public static void main(String[] args){
    OverLoad ov=new OverLoad();
    ov.hap(34.6f, 46);
    ov.hap(34, 62);
    ov.hap(74.6f, 26.6f);
  }
}
```

ov.hap(34.6f, 46);

ov.hap_float_int(34.6f, 46);

컴파일러가 만들어내는 메서드 이름

hap_int_int(int a, int b)

hap_float_int(float a, int b)

hap_float_float(float a, float b)

```
public class OverLoad{
  public void hap(int a, int b){
     System.out.println(a+b);
   }

  public void hap(float a, int b){
     System.out.println(a+b);
   }

  public void hap(float a, float b){
     System.out.println(a+b);
  }
}
```

오버로딩 (Overloading) : 정적 바인딩(컴파일 시점에서 호출될 메서드가 이미 결정되어 있는 바인딩)→속도와는 관계 없다.

2주차

본강의 - 1 (40~50분)	본강의 - 2 (40~50분)	본강의 - 3 (40~50분)	질의응답 (5개 내외)

6)Q&A 질의응답

질의응답(Q&A)

Q. 클래스를 2가지 측면에서 다시 한번 설명해주세요?

- DataType 측면 : 새로운 자료형을 만드는(설계하는) 도구 = **모델링**도구
- OOP(객체지향)측면 : 객체의 상태정보와 행위정보를 추출하여 **캡슐화** 하는 도구

Q. API란 무엇인가요?

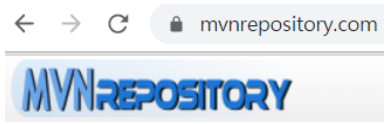
- **API(Application Programming Interface**, 응용 프로그램 프로그래밍 인터페이스)는 응용 프로그램에서 사용할 수 있도록, 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 **인터페이스**

Q. 개발자가 만드는 Model(class)의 종류는 어떤 게 있나요?

- **DTO**(Data Transfer Object) : 데이터 구조, 데이터를 담는 역할, **이동**하기위해서 데이터를 담는다.
  VO(Value Object) : 객체를 담아서 **하나의 값**(덩어리, 바구니)으로 취급한다는 의미로
- **DAO**(Data Access Object) : 데이터를 **처리**하는 역할(비즈니스 로직), 데이터베이스와 CRUD하는 역할
- **Utility**(Helper Object) : **도움**을 주는 기능을 제공하는 역할(날짜, 시간, 통화, 인코딩 등)

Q. 우리가 앞으로 사용하게 될 클래스들의 종류가 어떤 게 있나요?

- Java에서 제공해주는 class 들 : - **String, System, Integer, ArrayList, Map** 등
- 개발자가 만들어 사용하는 class 들 : - **DTO, DAO, Utility**
- 다른 회사(사람)에서 만들어서 제공해주는 class 들 : - **Gson, Jsoup, POI, iText** 등



3. 상속을 이해하라(inheritance)

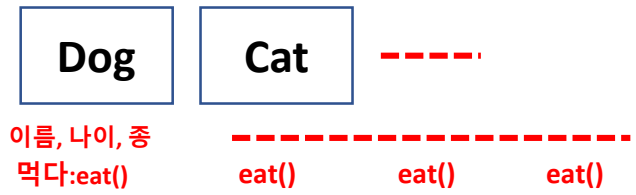
- 1) **상속**, 수평적 구조와 수직적 구조를 이해하기
- 2) **재정의(override)**란 무엇인가?
- 3) **상속관계**에서 객체생성 방법
- 4) **객체 형 변환**(Object Casting)이란?
- 5) **다형성**(Message Polymorphism)이란?
- 6) Q&A 질의응답

3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

1) 상속, 수평적 구조와 수직적 구조를 이해하기

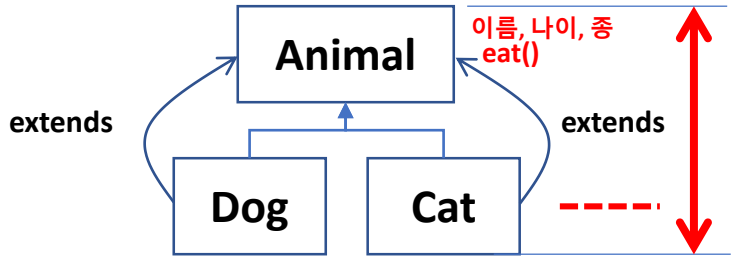
1. 상속→클래스의 설계(행위적인 측면)

수평적 설계



→코드의 중복이 발생
→새로운 요구사항에 대한 코드의 수정이 불가피하다.
→관리하기가 어렵다.(부모와 자식의 관계를 생각해보라)

수직적 설계(계층화, 상속구조)



추상화
보편화,
일반화,
개념화
super class(상위, 부모)

세분화,
상세화,
구체화,
구상화
sub class(하위, 자식)

→수평적 설계의 단점을 극복할 수 있다.
→확장을 쉽게 할 수 있다.
→코드가 복잡해 진다.(이점이 많다)

2. 상속 개념

```
public class Animal extends Object {
  public String name;
  public int age;
  public String part;
  public void eat(){
    System.out.println("?");
  }
  public Animal(){
    super();
  }
}
```

new Object();

이름, 나이, 종 공통

이름, 나이, 종 공통

```
public class Dog extends Animal {
  public void eat(){
    System.out.println("개 처럼 먹다.");
  }
  public Dog(){
    super();
  }
}
```

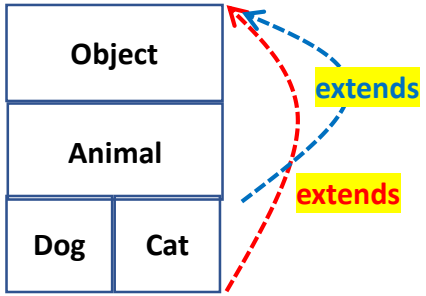
new Ainmal();

→상속에서 부모와 자식에 연결되는 방법
super() : 자신의 생성자에서 부모의 생성자를 호출

```
public class Cat  extends Animal {
  public void eat(){
    System.out.println("고양이 처럼 먹다.");
  }
  public Cat(){
    super();
  }
}
```

new Ainmal();

상속 memory



3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

2) 재정의(override)란 무엇인가?

2. 재정의(override)란 무엇인가?

1. Override(재정의) → 상속관계에서 상속받은 하위 클래스가 상위 클래스의 **동작을 수정하는** 것

```
public class Animal {
   public void eat() {
     System.out.println("?"); // 포괄적, 추상적
   }
}
```

Override (재정의)

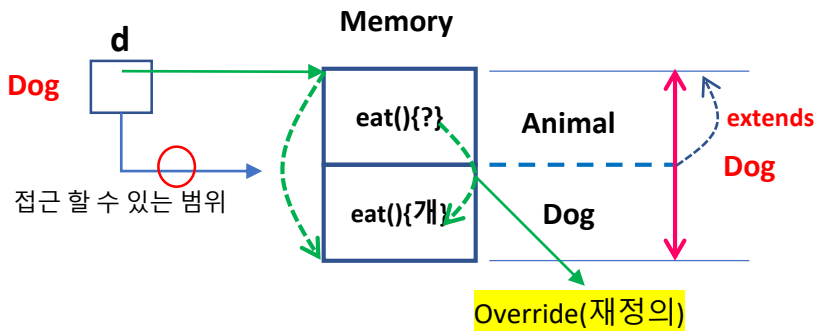
```
public class Dog extends Animal{
  public void eat(){
      System.out.println("개처럼 먹다.");
  }
}
```

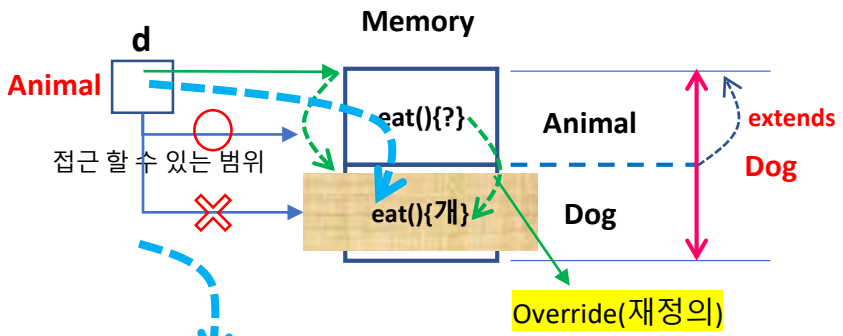
```
public class Cat extends Animal{
  public void eat(){
      System.out.println("고양이 처럼 먹다.");
  }
  public void night() {
     System.out.println("밤에 눈에서 빛이 난다.");
  }
}
```

Dog d=new Dog();
d.eat();

Override(재정의=보모 메서드 무시)
메모리에 부모와 자식 메서드가 공존하지만
결국에는 자식 메서드가 실행된다.

Animal d=new Dog();
d.eat();





→ Override(재정의) : 동적 바인딩(호출될 메서드가 실행시점에서 결정되는 바인딩)
프로그램의 속도가 떨어지는 원인이 되지만 이점이 더 많기때문에 사용 한다.

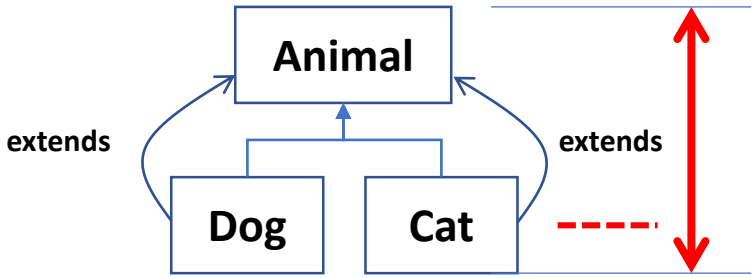
Point Override를 통해 하위 클래스를 접근 할 수 있다.

3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

3) 상속관계에서 객체생성 방법

1. 상속관계에서 객체생성 방법

상위클래스를(부모를) 활용하라!



Animal d=new Dog(); 간접

Dog d=new Dog(); 직접

부모 클래스를 이용하지 않는 방식(직접이용)

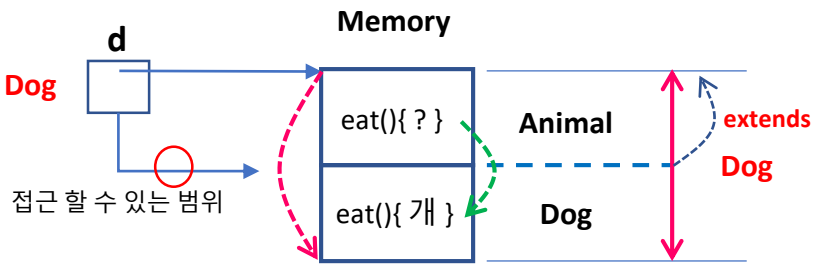
```
Dog d=new Dog();
Cat c=new Cat();
```

부모 클래스를 이용하는 방식(하위 클래스의 동작 방식을 모를 때, 간접이용)

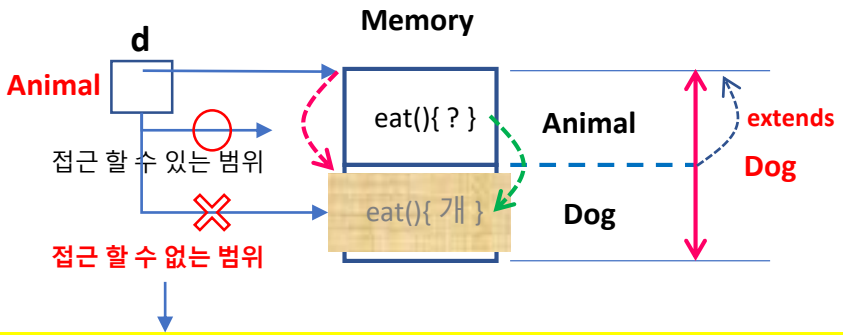
→ .class(실행) 파일만 있고 .java(소스)파일이 없는 경우

```
Animal d=new Dog();
Animal c=new Cat();
```

Dog d=new Dog();



Animal d=new Dog();



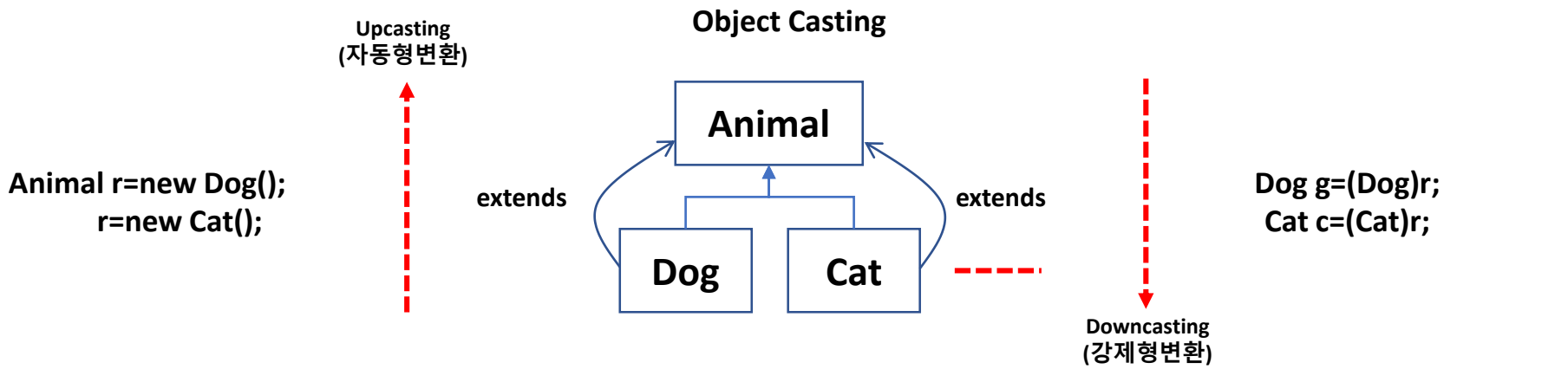
→ 하위 클래스를 접근 할 수 없다. 가능 하게 하는 방법 ? Override(재정의)

3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

4) 객체 형 변환(Object Casting)이란?

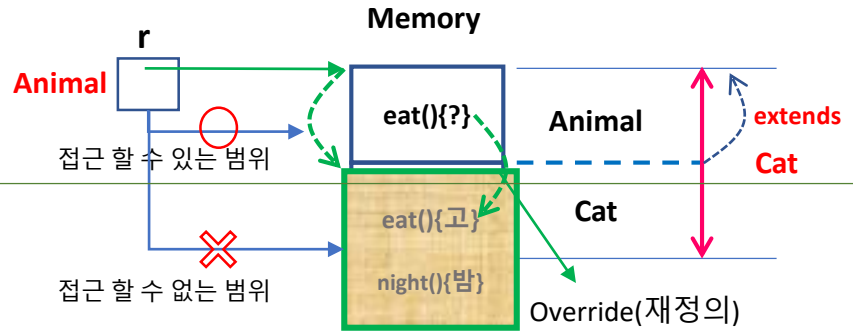
1. Object Casting(객체 형 변환)

- **상속관계**에 있는 클래스들 간의 데이터 형(DataType)을 바꾸는 것



Animal r=new Cat(); // upcasting
r.eat();

고양이처럼 먹다.



Animal r=new Cat();
// r.night();
Cat c=(Cat)r; // downcasting
c.night();

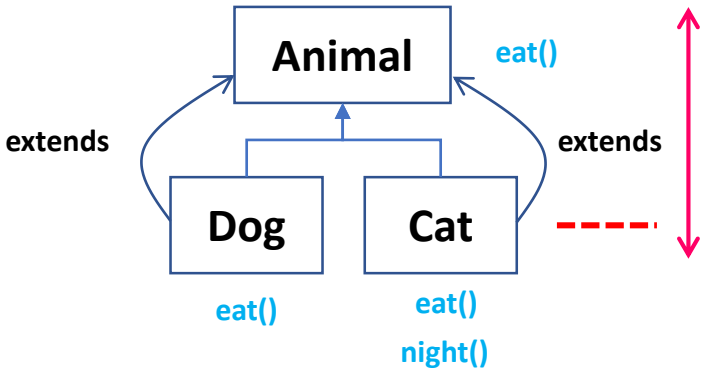
밤에 눈에서 빛이 난다.

3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

5) 다형성(Message Polymorphism)이란?

1. message polymorphism(다형성)

- 상속관계에 있는 클래스에서 상위클래스가 동일한 메시지로 하위클래스들을 서로 다르게 동작시키는 객체지향 원리(개념)



message polymorphism(다형성)

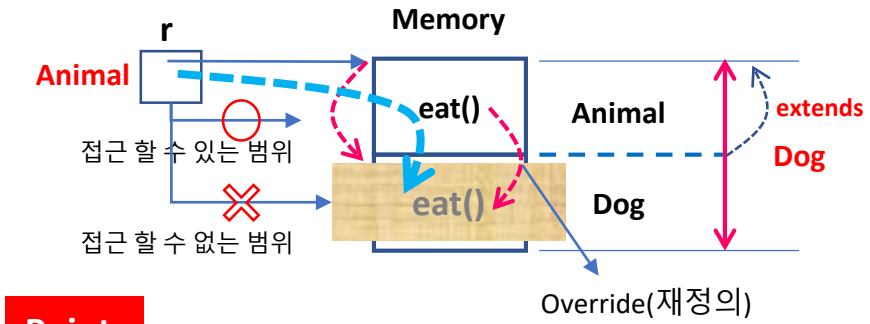
상위클래스인 Animal이 하위클래스인 Dog와 Cat에게 동일하게 먹어라(eat)라고 메시지를 보냈을 때 하위 클래스인 Dog와 Cat의 eat()메서드가 **서로 다르게 동작되는 객체지향 원리**

Point 개처럼 먹다. 고양이처럼 먹다. → 동작은 하지만 결과는 다르다.

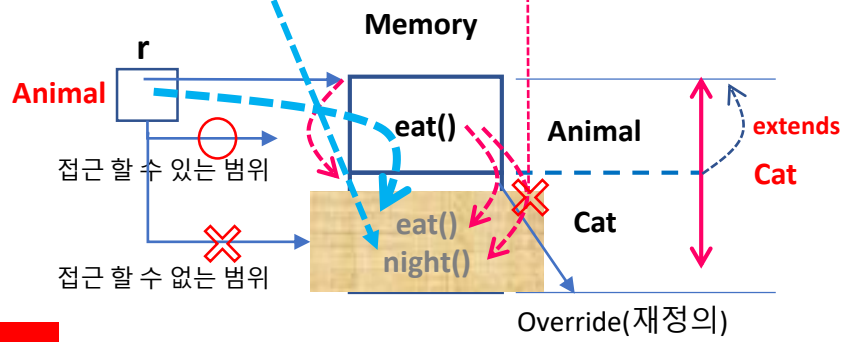
```
Animal r=new Dog();
r.eat();
r=new Cat();
r.eat();
```

하위클래스의 동작방식을 알 수 없어도 상위클래스를 통해 하위클래스를 구동 시킬 수 있다.

```
Animal r=new Cat();
r.night();  night() method는 재정의가 되지 않음(접근불가)

Cat c=(Cat)r; // downcastiing
c.night();
```



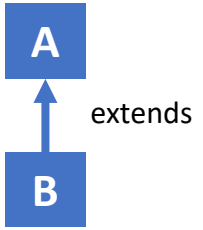
Point Override를 통해 하위 클래스를 접근 할 수 있다.



Point Downcasting으로 하위클래스의 night()를 접근할 수 있다.

1. 다형성 이론의 전제조건(부모 클래스를 잘 활용하라)

→ 상속관계가 되어야 한다.
→ 객체생성을 upcasting으로 할 것(상위클래스가 하위클래스에게 메세지를 보내야 하므로)
(upcasting이 되면 downcasting을 할 수 있다.) → A a=new B();



→ 하위클래스가 반드시 재정의(override)해야 한다.(다형성이 보장되기 위해서는)
→ 동적 바인딩을 통해 실현된다.
(동적 바인딩 : 실행시점에서 사용될 메서드가 결정되는 바인딩, 프로그램의 속도를 떨어뜨리는 원인이 된다.)

2. 다형성활용 방법 부모 클래스를 잘 활용하라

1. 다형성인수(데이터 이동)

```
Dog d=new Dog();
 display(d);
Cat c=new Cat();
 display(c);
```

d c

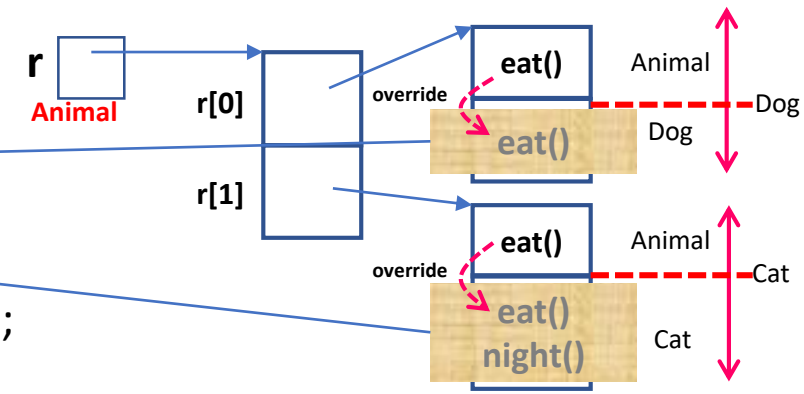
```
public static void display(Animal r ){
   r.eat();
}
```

다형성인수
(Animal의 모든 하위클래스를 받을 수 있다.)

2. 다형성 배열(서로 다른 객체를 담을 수 있다)

```
Animal[] r=new Animal[2];
r[0]=new Dog();
r[1]=new Cat();

r[0].eat();

r[1].eat();
((Cat)r[1]).night();
```

다형성배열
서로 다른 하위클래스를 담을 수 있다.



3주차 | 본강의 - 1 (40~50분) | 본강의 - 2 (40~50분) | 본강의 - 3 (40~50분) | 질의응답 (5개 내외)

6) Q&A 질의응답

질의응답(Q&A)

Q. 상속을 알아야 되는 이유가 무엇인가요 ?

- 자바 모든 API가 상속 관계로 되어있기 때문에 상속구조를 알아야 API를 잘 활용 할 수가 있다.
- 상속관계에서 상위클래스, 즉 부모 클래스를 알면 자식, 즉 하위 클래스들을 이해하기가 쉽다.

Q. 상속 체이닝(Inheritance Chaining)이란 무엇인가 ?

- 상속 체이닝은 가장 **상위 클래스**부터 객체가 생성된 뒤 **연쇄적(Chaining)**으로 **하위 클래스**로 객체가 생성되는 방법을 말한다. 내가 존재하기 위해선 반드시 부모님이 먼저 태어나야 한다. 이 과정은 super()라는 키워드로 이루어진다.

Q. Message polymorphism(다형성)이란?

- 다형성이란 상속 관계에 있는 클래스에서 부모 클래스가 동일한 메세지로 자식 클래스들을 서로 다르게 동작시키는 원리이다. 같은 메서드를 호출해도 출력되는 결과 값이 달라진다. 객체지향의 핵심 이론이다.

Q. 동적바인딩과 정적바인딩이란 무엇인가 ?

- 동적 바인딩은 프로그램이 실행될 때(실행 시점) 호출될 메서드가 정해지는 것(override)
- 정적 바인딩은 프로그램 시작 전(컴파일 시점)에 미리 호출될 메서드가 정해지는 것(overloading)