Chapter 4. 객체지향 프로그래밍 문법

28장. 패키지

▷ 패키지는 폴더의 다른 이름이다. 없으면 오류가 발생한다.

▷ .java 파일의 모임이다.

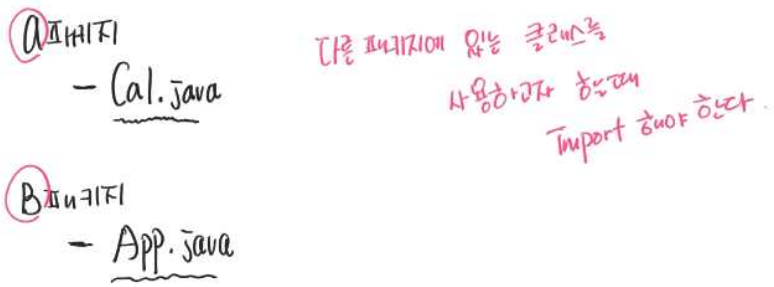
```
package ch04;
public class PackageEx01{
   public static void main(String[] args){
   }
}
```

→ PackageEx01.java 파일은 ch04 폴더 밑에 있다. 라는 의미

29장. import

▷ import : 수입하다. 가져오다 라는 영문 뜻.

▷ 특정 패키지의 java 파일을 다른 패키지에서 사용하고자 할 때 ...




```
package ch04.a;
public class Cal{
   void add(){
   // 동일한 패키지에서만 접근이 가능함.
      System.out.println("더하기 메서드");
   }
   void minus(){
      System.out.println("빼기 메서드");
   }
}
package ch04.b;
import ch04.a.Cal;   // import 는 파일명까지 적어줘야 한다.
public class PackageEx01{
   public static void main(String[] args){
      Cal c = new Cal();
      Cal.add();
      // 패키지명을 적고 Ctrl+Space를 누르면 자동으로 패키지가 추가된다.
```

```
    // 그렇지 않으면 패키지의 전체 경로를 적어 준다.
   // 그러나 Cal.add()를 사용할 수 없다... → 접근제어의 문제가 발생하기 때문
  }
}
```

30장. 접근제어자

▷ 다른 패키지, 혹은 같은 패키지에 있지만 다른 클래스에 있는 멤버나 메서드의 접근 방법

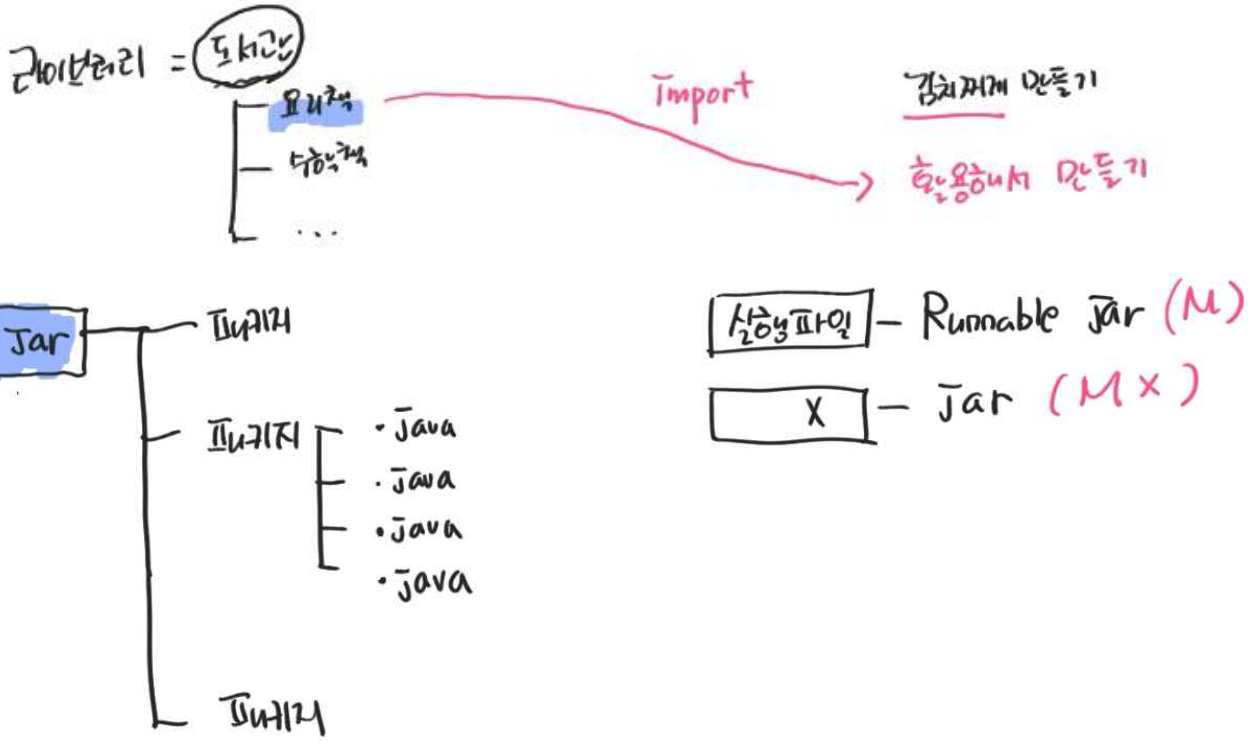
```
package ch04.a;
public class Call{
  void add(){  // default 접근 제어자(아무것도 적지 않은 경우)
  // 동일한 패키지에서만 접근이 가능함.
    System.out.println("더하기 메서드");
  }
  public void minus(){
    System.out.println("빼기 메서드");
  }
  private void multi(){  // 현재 클래스 내부에서만 접근 가능
    System.out.println("곱하기 메서드");
  }
  public void devide(){
    System.out.println("나누기 메서드");
    multi(); // 같은 클래스 내부에 있으니 찾을 수 있다.
  }
}
public class App2{
  public static void main(String[] args){
    Call c = new Call();
    c.add(); // 동일한 패키지에 있으니 찾을 수 있다.
    c.minus();  // 모든 패키지에서 접근가능하니 찾을 수 있다.
    //c.multil  // 찾을 수 없다. 같은 클래스가 아니기 때문
  }
}

package ch04.b;
import ch04.a.Cal;  // import 는 파일명까지 적어줘야 한다.
public class App{
  // JVM이 main메서드를 찾으려면 public 이 필요(공개)
```

```
    // JVM이 main 메서드를 찾으려면 static 이 필요(메모리에 올린다는 뜻)
    // main 메서드만 return 타입을 허용하지 않는다.
    // String[] args 매개변수
    public static void main(String[] args){
        Cal c = new Cal();
        c.minus();
    }
}
```

① default : 동일한 패키지 내에서만 접근 가능
② public : 모든 패키지에서 접근 가능
③ private : 동일한 클래스에서만 접근 가능
④ 클래스 자체에 public 이 없으면 import 자체가 안된다.

31장. 라이브러리 만들기

▷ 라이브러리의 개념

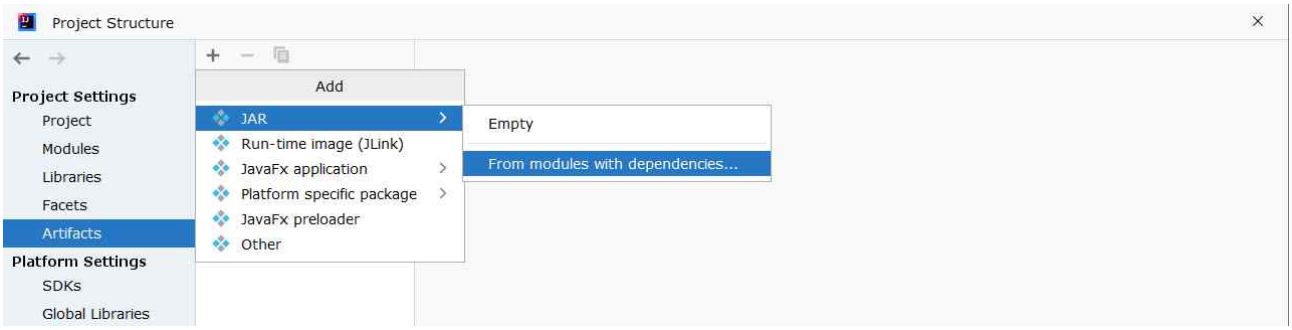


▶ 프로젝트 생성해서 Export 하기

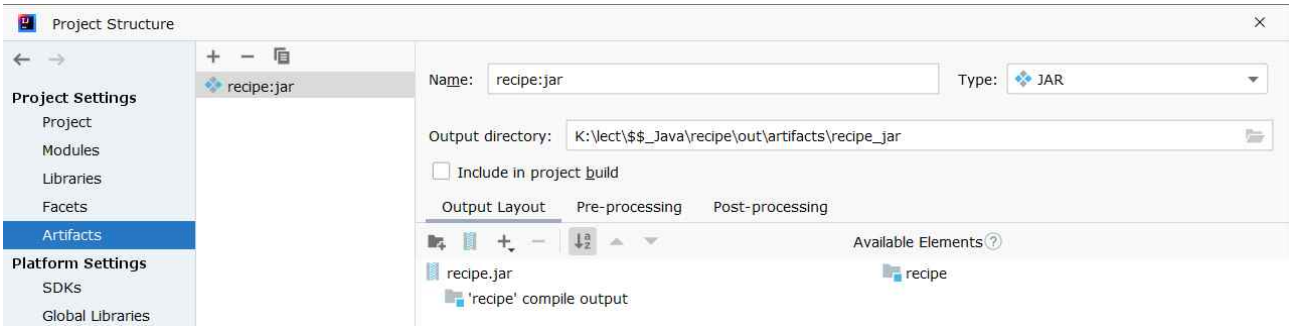
① recipe 프로젝트 생성 → recipe 패키지 생성

```
package recipe;
public class MySoup{
   public void 김치찌개(){
      System.out.println("김치찌개가 만들어졌습니다.");
   }
}
```

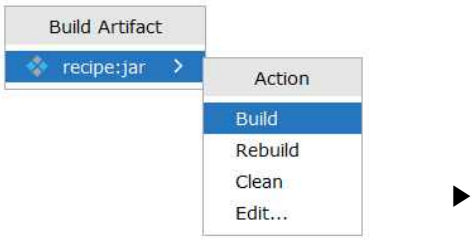
② File → Project Structure → Artifacts → JAR → From modules with dependencies...

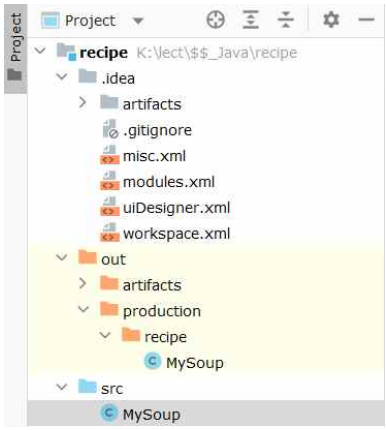


③ jar 파일 생성 완료



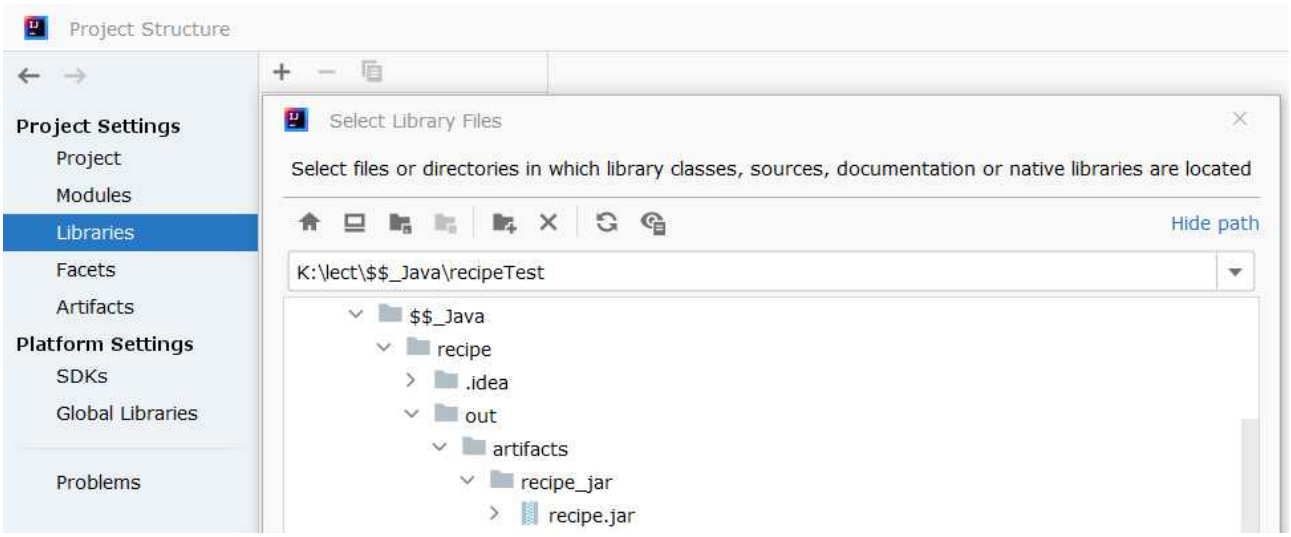
④ Build → Build Project 하고 다시 Build → Build Artifact

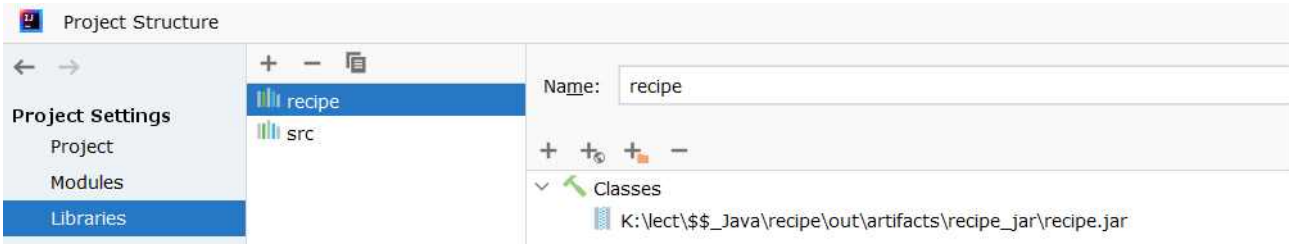




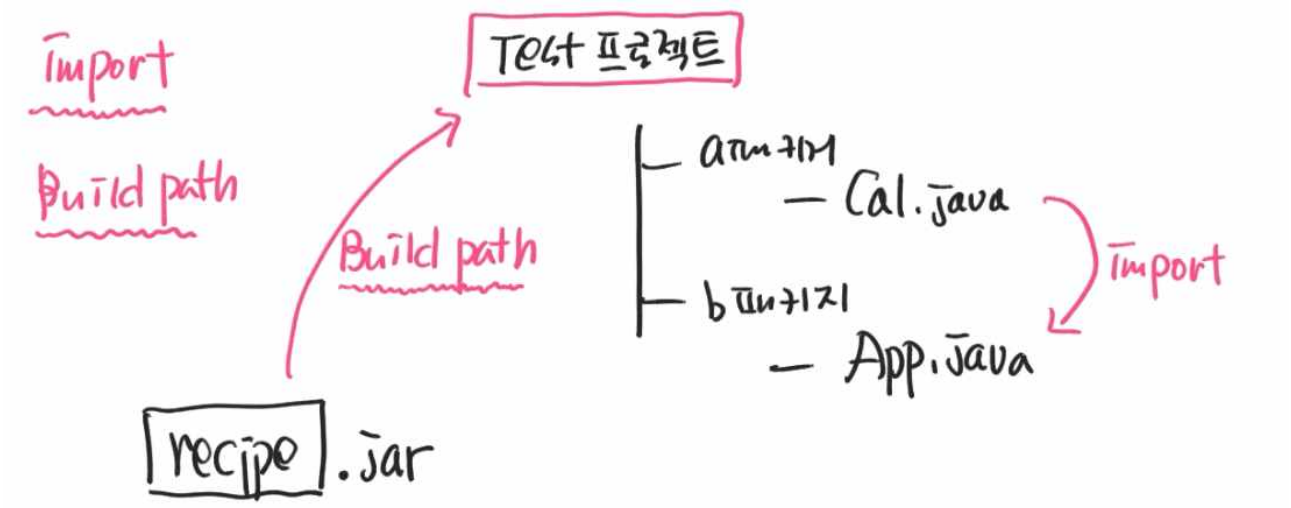
▶ **프로젝트 생성해서 import 하기**

① File → Project Structure → Libraries → JAVA → Select Library Files





② Referenced Libraries 에 해당 jar 파일이 보임.



```
package ch04;
import recipe.MySoup;
public class LibEx01{
   public static void main(String[] args){
      MySoup m = new MySoup();          // Ctrl+Space 누르면 자동완성 됨.
      m.김치찌개();
   }
}
```

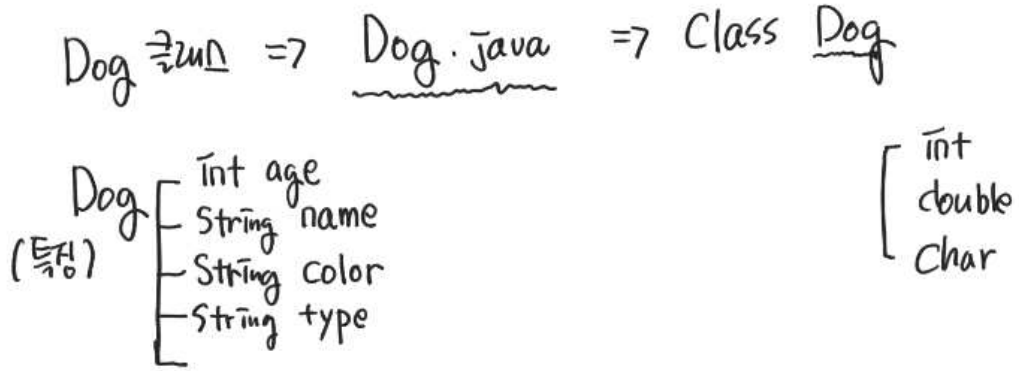
※ JDK 설치 – JDK(javac, java), JRE(라이브러리), JVM(자바 가상머신)

```
package ch04;
import java.util.Random;
public class LibEx01{
   public static void main(String[] args){
      Random r = new Random();
      System.out.println(r.nextInt());    // Ctrl+Space 누르면 자동완성 됨.
                  // static 이 존재하면 Random. 하면 나온다. 없으니 안나옴.
                  // 정수 랜덤파일 생성하기
   }
}
```

32장. 클래스 만들기(final)

▷ 클래스 = Class = 설계도

→ 클래스는 여러 가지 특성(상태)을 갖고 있다.




```
package ch04;
public class Dog{
  int age = 20;  //변경 가능
  String name = "토토"; //변경 가능
  String color = "하얀색"; //변경 가능
  final String type = "말티즈"; //변경 불가능 → final을 붙이면 상수가 된다.
}
```

```
package ch04;
public class DogApp{
  public static void main(String[] args){
    Dog d1 = new Dog();
    System.out.println(d1.name);
    System.out.println(d1.age);
    System.out.println(d1.color);
    System.out.println(d1.type);

    // 1년이 지났다. 강아지가 염색을 노락생, 이름도 레르코
    d1.age = d1.age + 1;
    d1.color = "노란색";
    d1.name = "레르코";
    // d1.type = "푸들";   // 변경 불가능 Why? final 이니까
    System.out.println(d1.name);
    System.out.println(d1.age);
    System.out.println(d1.color);
    System.out.println(d1.type);
  }
}
```

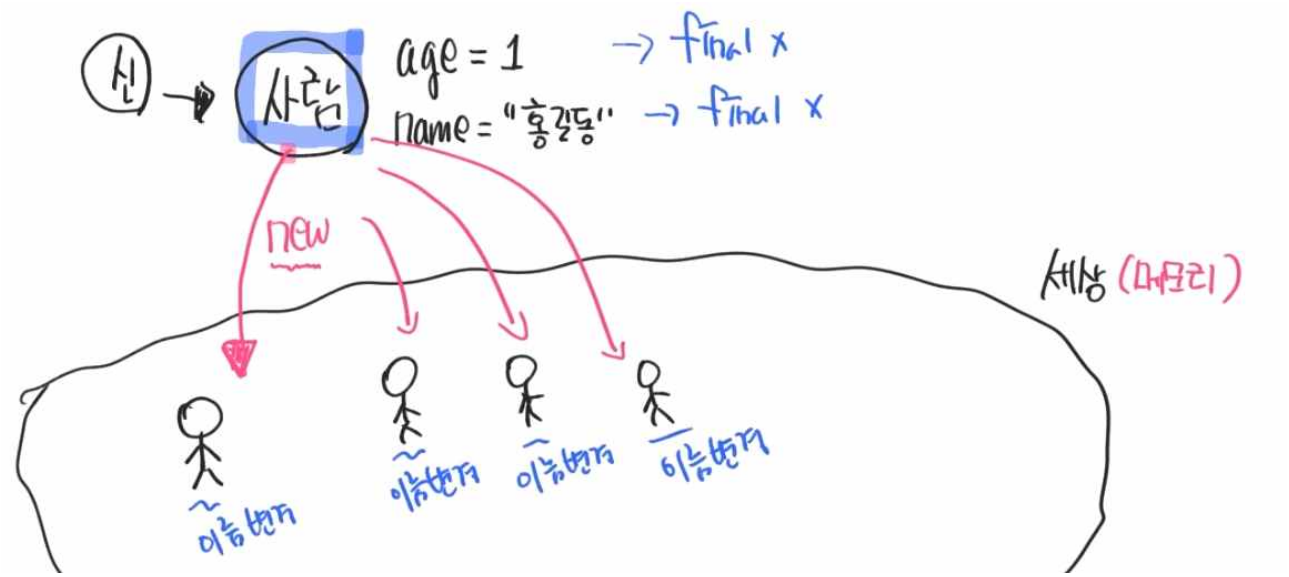
33장. 클래스 만들기(생성자)

▷ 클래스 생성시 초기화를 하지 않으면 null 값이 들어간다.

▷ Default Constructor(인자를 갖지 않는 생성자)는 기본적으로 생성됨.

```
package ch04;
// 값을 초기화하지 않는 이유는 new 할 때마다 다른 상태를 갖는 고양이를 만들기 위해
public class Cat{
  String name;
  String color;
  // default constructor
  public Cat(){
  }
  public Cat(String n, String c){
    System.out.println("고양이가 탄생함");
    // String n, String c 는 지역변수 = Cat() 이라는 생성자(메서드)가 가진 지역변수
    name = n;
    color = c;
    System.out.println("n : " + n);
    System.out.println("c : " + c);
  }
}
```

```
package ch04;
public class CatApp{
  public static void main(String[] args){
    // new → 메모리에 할당(heap) → String name, String color
    // Cat() → 생성자 호출 → 상태를 초기화하기 위해서
    // c1 → heap 공간을 가르키는 주소
    // Cat 커스텀 타입(개발자가 만드는 타입)
    Cat c1 = new Cat("집사1", "하얀색");
    System.out.println(c1.name);
    System.out.println(c1.color);

    Cat c2 = new Cat("집사2", "노란색");
    System.out.println(c2.name);
    System.out.println(c2.color);
  }
}
```



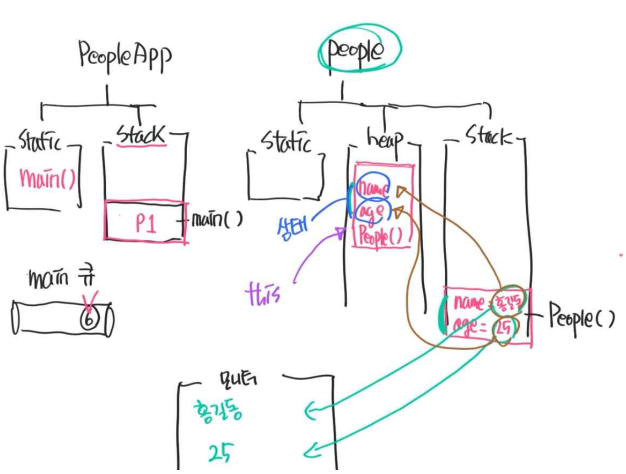
→ 이러한 다양성을 갖기 위해 생성자를 사용해서 다른 값을 갖는 사람을 만듦

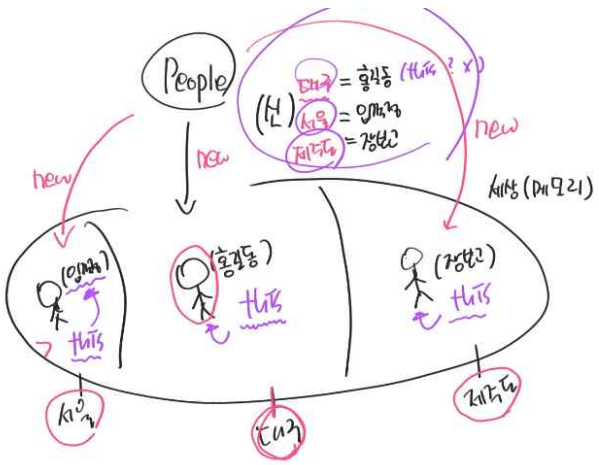
34장. 클래스 만들기(this)

▷ this 는 자기자신의 Heap 공간을 의미한다.

```
package ch04;
public class People{
   String name;
   int age;

   public People(){
   }

   public People(String name, ing age){
      System.out.println("사람이 태어남");
      this.name = name;
      this.age = age;
      System.out.println("People 메서드 스택 name : " + name);
      System.out.println("People 메서드 스택 age : " + age);
   }
}
```

```
package ch04;
public class PeopleApp{
   public static void main(String[] args){
      People p1 = new People("홍길동", "25");
      System.out.println(p1.name);
      System.out.println(p1.age);
   }
}
```





▶ Chapter 4 연습문제

▷ 다음 클래스 구성도를 보고 Food 클래스를 생성하시오.


```
package ch04;
public class Food{
   String name;
   int price;

   public Food(String name, int price){
      this.name = name;
      this.price = price;
   }
   public void myPrint(){
      System.out.println(name + "가격은 " + price + "입니다.");
   }
}

public class ExampleEx01{
   public static void main(String[] args){
      Food f1 = new Food("치킨", 2000);
      Food f2 = new Food("피자", 5000);
   }
}
```