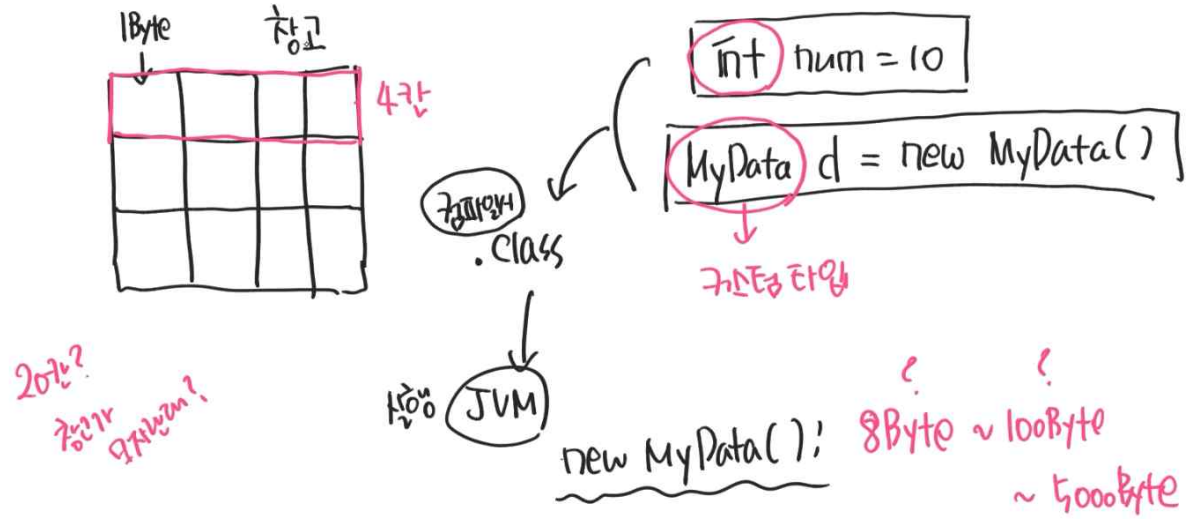
Chapter 2. 자바메모리 구조

12장. 일반변수와 레퍼런스 변수

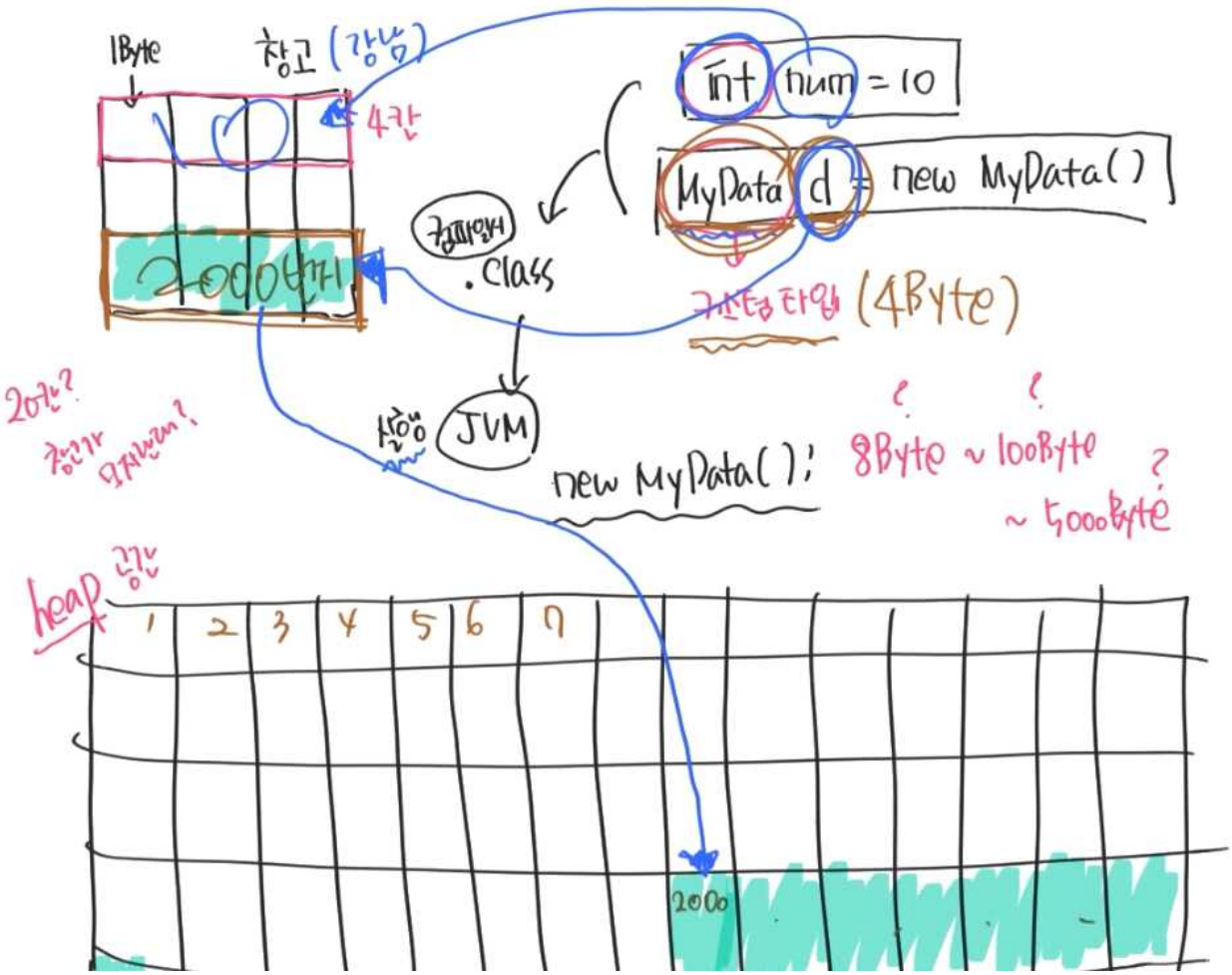
▶ package : ch02

```
package ch02;
// new가 되어서 힙에 할당 될 때 사이즈를 알 수 있다.(프로그램이 실행되었을 때 = Runtime)
class MyData{
   int id = 1; // 4Byte
   int price = 1000; // 4Byte
}
// 레퍼런스 변수, 일반 변수
public class VarRefEx01{
   public static void main(String[] args){
      // 타입 변수; int num; 변수를 선언한다.
      // 타입 변수 = 값; int num = 10; 변수를 초기화(메모리 할당)한다.
      int num = 10; // 일반변수 (크기가 정해져 있는 것 = 컴파일 시점)
      MyData d = new MyData(); // 레퍼런스 변수(크기가 정해져 있지 않는 것 = 컴파일 시점)
      System.out.println(num);
      System.out.println(d.id);
      System.out.println(d.price);
   }
}
```



▷ 커스텀 타입 즉, 사용자가 만든 데이터 타입은 JVM이 실제 얼마만큼의 메모리를 사용하는지 모른다. 그래서 실행되는 시점에서 메모리가 할당된다.

▷ 그래서 저기 변두리 땅값 싼데다가 창고를 무지하게 크게 지어놓는다. 이것이 heap 공간이다.



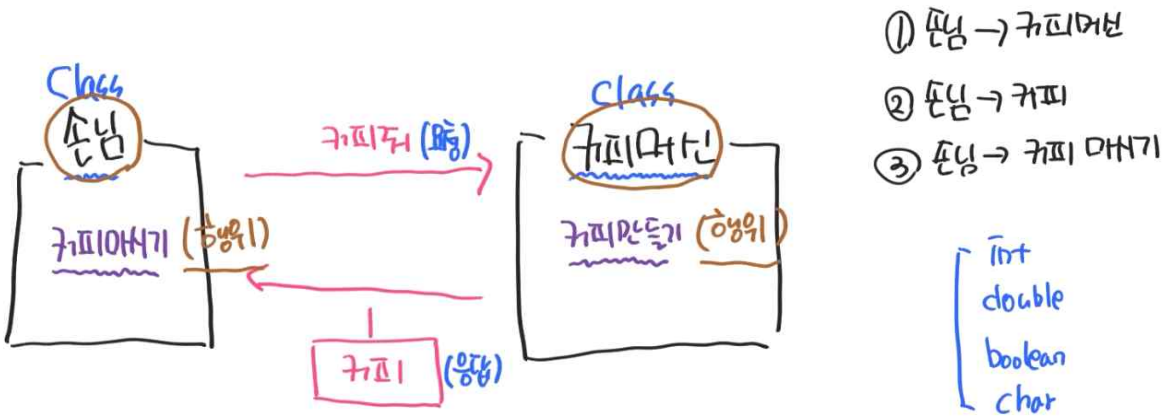
▷ 일반변수는 값을 갖고 있고, 레퍼런스 변수는 주소를 갖고 있으며 주소를 통해 실제 값을 찾아간다.

▷ 일반변수 : int, double, boolean, char

▷ 레퍼런스 변수 : class 자료형

13장. 메서드(Method)

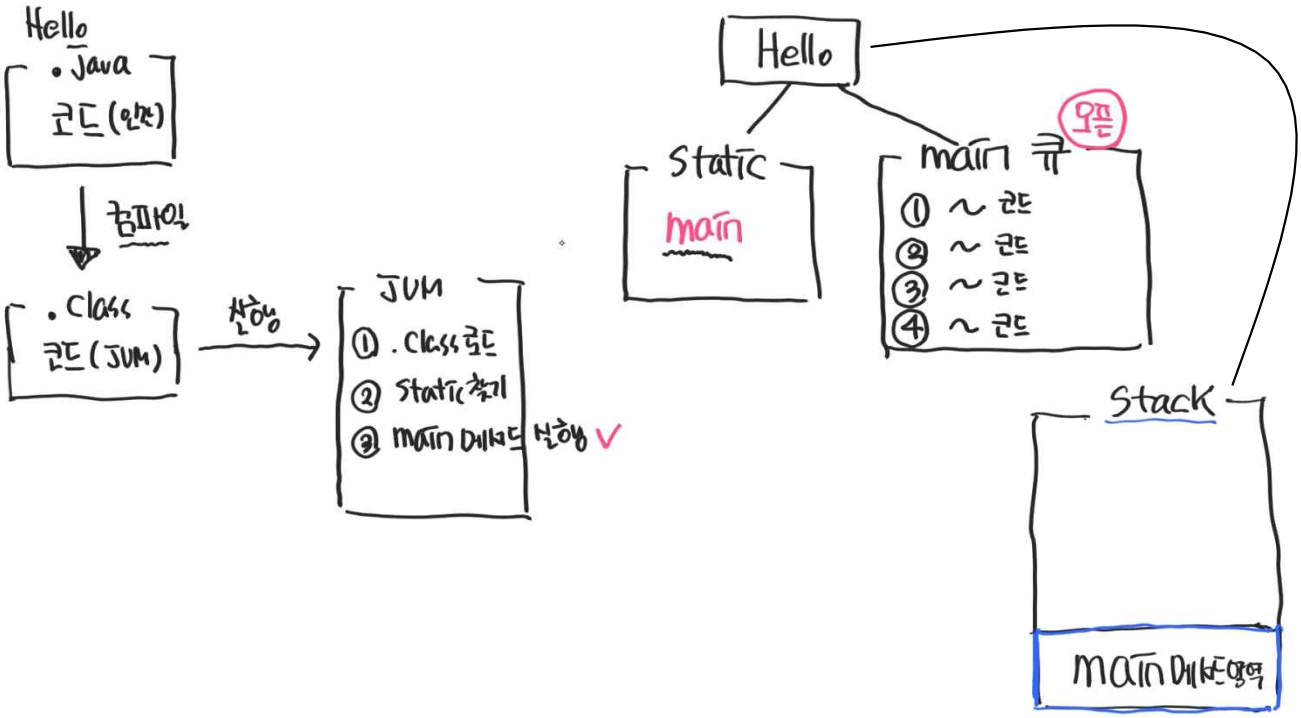
▶ 메서드는 특정 클래스가 갖고 있는 행위를 하는 의미한다.




```
package ch02;
class 손님{
  // 손님의 행위
  void 커피마시기(){
    System.out.println("손님이 커피를 마셔요");
  }
}

public class MethodEx01{
  // 메서드 = MethodEx01 클래스의 행위
  static void start(){
    System.out.println("만나서 반갑습니다."); // String 타입
    System.out.println("start 메서드 종료");
  }
  // start를 인지하는 방법은 2가지(1. static 붙이기, 2. new 해서 heap에 올리기)
  public static void main(String[] args){
    MethodEx01.start();
    손님.s = new 손님();
    s.커피마시기();
  }
}
```

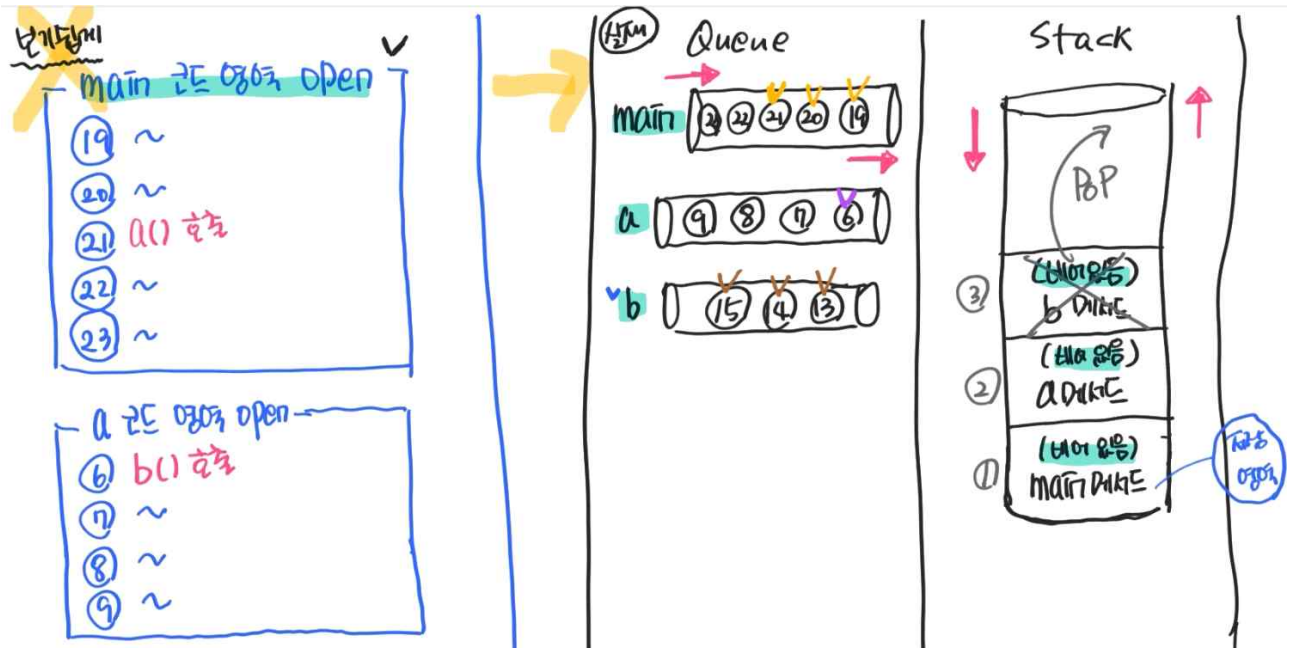
14장. 메서드 Stack 메모리-1 ★★★★★

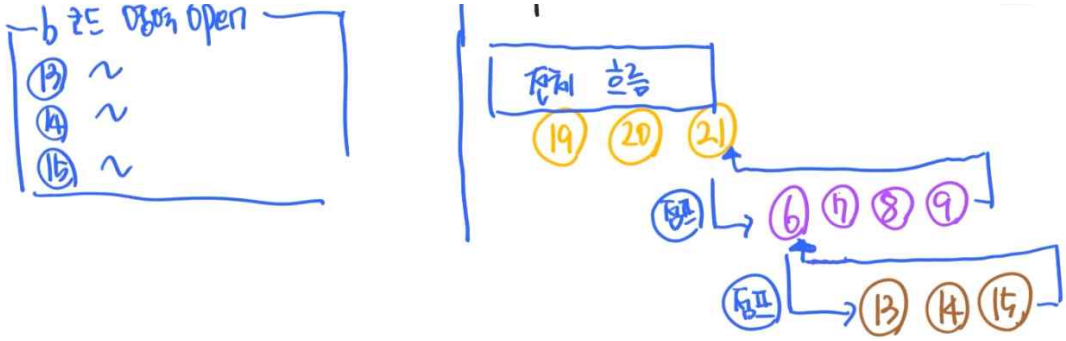
```
package ch02;
public class StackEx01{
   static void a(){
      StackEx01.b();
      System.out.println("a2”);
      System.out.println("a3”);
      System.out.println("a4”);
   }

   static void b(){
      System.out.println("b1”);
      System.out.println("b2”);
      System.out.println("b3”);
   }

   public static void main(String[] args){
      System.out.println("m1”);
      System.out.println("m2”);
      StackEx01.a();
      System.out.println("m4”);
      System.out.println("m5”);
   }
}
```



15장. 메서드 Stack 메모리-2

▷ 스택과 큐의 자료구조 확인




b 라는 영역 open
13 ~
14 ~
15 ~
작업 흐름
19
20
21
탭I
6
7
8
9
탭II
13
14
15

16장. 지역변수와 전역변수

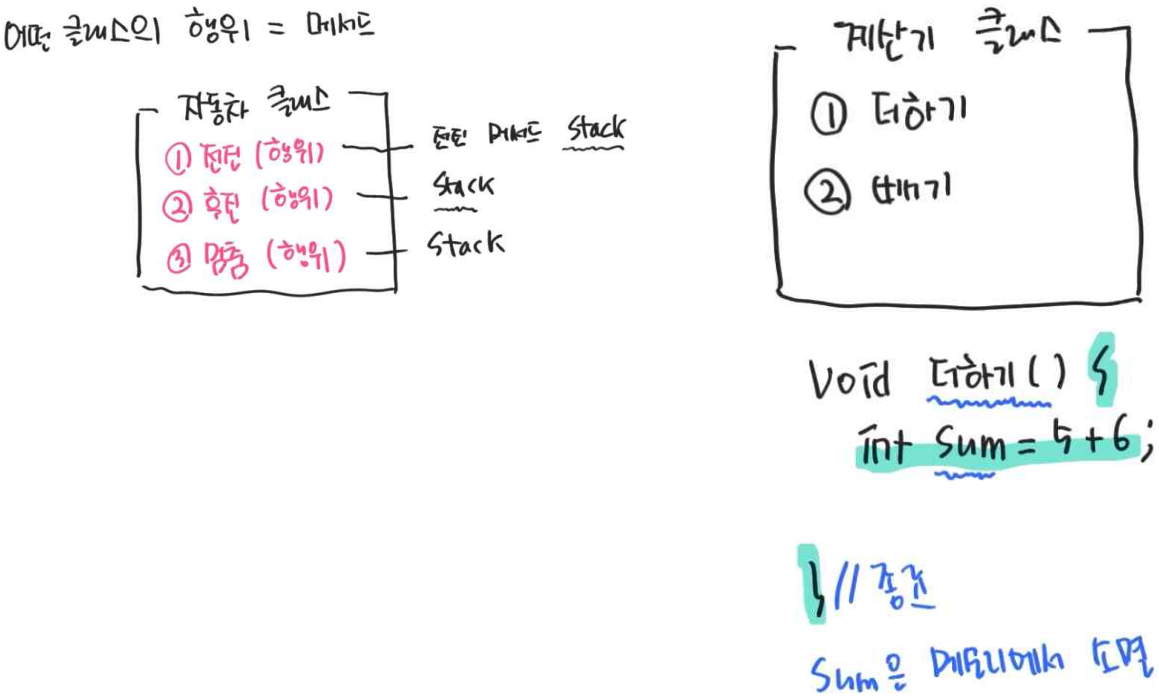
▷ 지역변수 : Stack (생명주기 : 짧다.)

▷ 전역변수 : heap, Static (생명주기 : 길다)

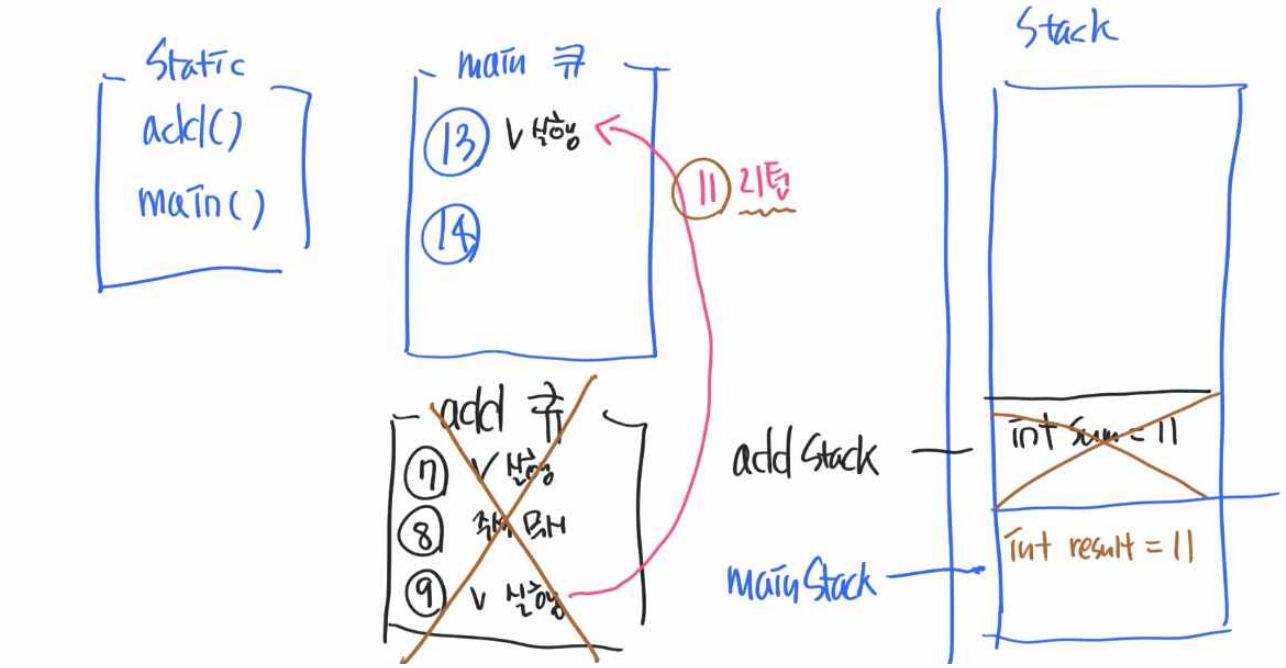
```
package ch02;
public class StackEx02{
  static int sum = 20; // 전역변수 → main 메서드 시작되기 전에 할당
  in value = 50;  // 얘를 메모리에 띄우는 방법은 class StackEx02를 new 해야 한다.
                  // 전역변수(heap에 할당) → 더이상 참조하지 않을 때 자동으로 사라진다.
  static void a(){
    int n1 = 1;  // a 메서드의 스택영역에 저장됨(지역변수)
    System.out.println(n1); a 메서드의 스택에서 끌고 옴.
    //static int n2 = 2;  // a 함수가 호출될 때 이 라인이 실행되는데 그 때 n2가 static memory에
                          // 할당되어야 하는데 시간적으로 그럴 수가 없다. 불가능 임.
  }

  public static void main(String[] args){
    // a 메서드가 실행될 때 Stack 공간에 n1이 할당되고 a 메서드가 종료되면 메모리서 사라진다.
    a(); // a 함수는 호출이 가능하다.
    System.out.println(n1); // n1에 직접 접근할 수 있는 방법이 없다.
    // n1은 a의 스택 메모리에 있기 때문에 main에서 접근 불가
    StackEx02 s = new StackEx02;
    // new 하면 StackEx02가 갖고 있는 Static을 제외한 나머지 것들을 heap 영역으로 끌고 온다.
    System.out.println(s.value);
    System.out.println("test1 : 여기서 value가 메모리에서 사라진다.");
  }
}
```

17장. 메서드의 리턴

▷ 메서드 란? 어떤 클래스에 행위



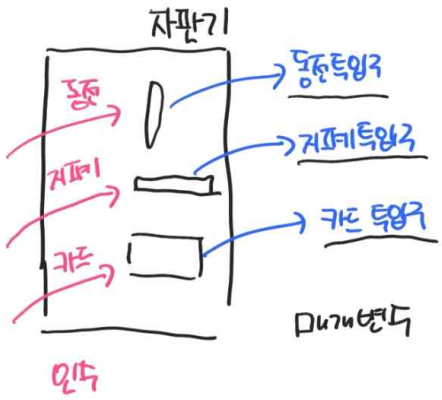

```
package ch02;
public class MethodEx02{
  // void는 리턴(돌려주지)하지 않겠다.
  //static void add(){ // 호출이 되면 중괄호 내부 실행 → 이때 stack 공간이 메모리에 만들어 진다.
  static int add(){
    int sum = 5+6;
    //System.out.println(sum);
    return sum;
  }

  public static void main(String[] args){
    int result = add(); // add() 메서드 호출
    System.out.println(result);
  }
}
```



※ 자판기는 반드시 리턴을 해 주지만 TV 켜는 동작 같은 경우 사용자에게 리턴을 안 해 줄수도 있다. 자기 자신이 뭔가를 처리하고 끝내야 할 경우 그렇다. 그럴때는 void 로 함수를 선언한다.

18장. 메서드의 매개변수와 인수

▷ 매서의 특정 동작을 실행시키기 위해 전달 받아야 하는 변수를 매개변수라 하고, 그 값을 인수라고 한다.




```
package ch02;
public class MethodEx03{
   static void 자판기_음료_돌려주기(int coin){
      System.out.println("동전을 투입하였습니다.");
   }
   static void 자판기_음료_돌려주기_2(double paper){
      System.out.println("지폐를 투입하였습니다.");
   }
   static void 자판기_음료_돌려주기_2(String card){
      System.out.println("카드를 투입하였습니다.");
   }
   static void add(int n1, int n2){
      System.out.println("더하기 메서드가 호출되였습니다.");
      int sum = n1 + n2;
      System.out.println(sum);
   }

   public static void main(String[] args){
      자판기_음료_돌려주기(10000);
      자판기_음료_돌려주기_2(1000.5);
      자판기_음료_돌려주기_2("삼성카드");
      add(20, 6);
   }
}
```

▶ Chapter 2 연습문제

▶ 아래의 코드를 보고 파란색 밑줄 친 부분에 생기는 오류에 대한 원인에 대해서 서술하시오.

```
Class A{
   int num = 10;
   static void start(){
      System.out.println(num);
   }
}
```