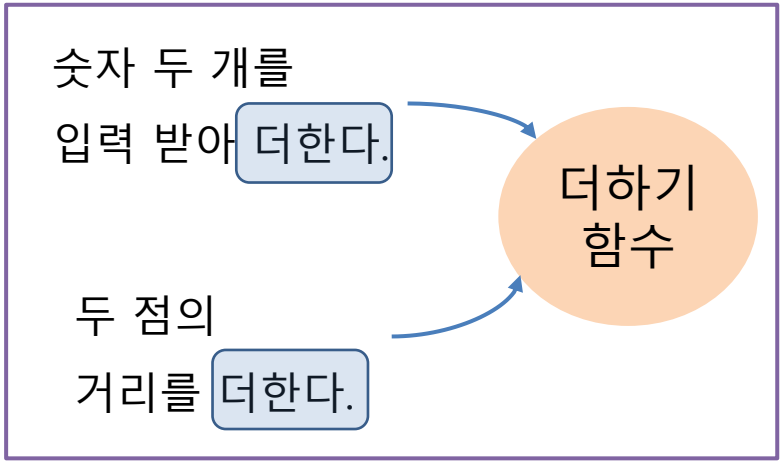
4장. 매서드(함수)

Method(Function)

메서드(멤버 함수)

- **메서드란?**
  - 객체의 기능을 제공하기 위해 클래스 내부에 구현되는 함수
- **함수란?**
  - 하나의 기능을 수행하는 일련의 코드
  - 중복되는 기능은 함수로 구현하여 함수를 호출하여 사용함



- **함수의 장점**
  - 기능을 나누어 코드를 효율적으로 구현
    예) 사칙연산 계산기 – 덧셈, 뺄셈, 곱셈, 나눗셈

※ main()함수 안에서 한꺼번에 구현하면 복잡함
add(), subtract(), times(), divide()

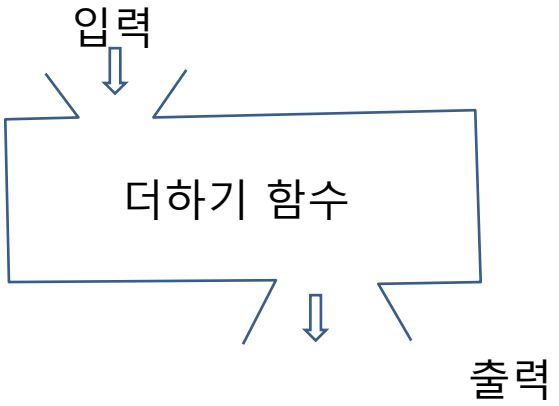
메서드(멤버 함수)

- **메서드(함수) 정의하기**
  - 함수의 이름, 매개변수, 반환값을 선언하고 코드를 구현함
    - 반환형이 없는 경우 void로 쓴다.
    - 매개변수가 없을 수도 있다.

- **함수의 유형**

1. 반환값이 없는 경우

```
void 함수이름(){
    . . .
}
```



2. 매개변수가 있는 경우

```
void 함수이름(매개변수){
    . . .
}
```

메서드(멤버 함수)

1. 반환값이 없는 메서드(함수) – 예제

메서드(함수)를 사용하는 것을 '함수를 호출한다'라고 한다.

main()이 있는 파일에서는 static을 사용하여 new 객체를 생성하지 않고도 실행할 수 있다. **(static을 사용해야 하는 이유)**

```
public static void main(String[] args) {
    sayHello();
    sayHello2("민수");
    sayHello2("은서");

}

public static void sayHello() {
    System.out.println("hello~");
}

public static void sayHello2(String name) {
    System.out.println("hello~ " + name);
}
```

메서드 호출

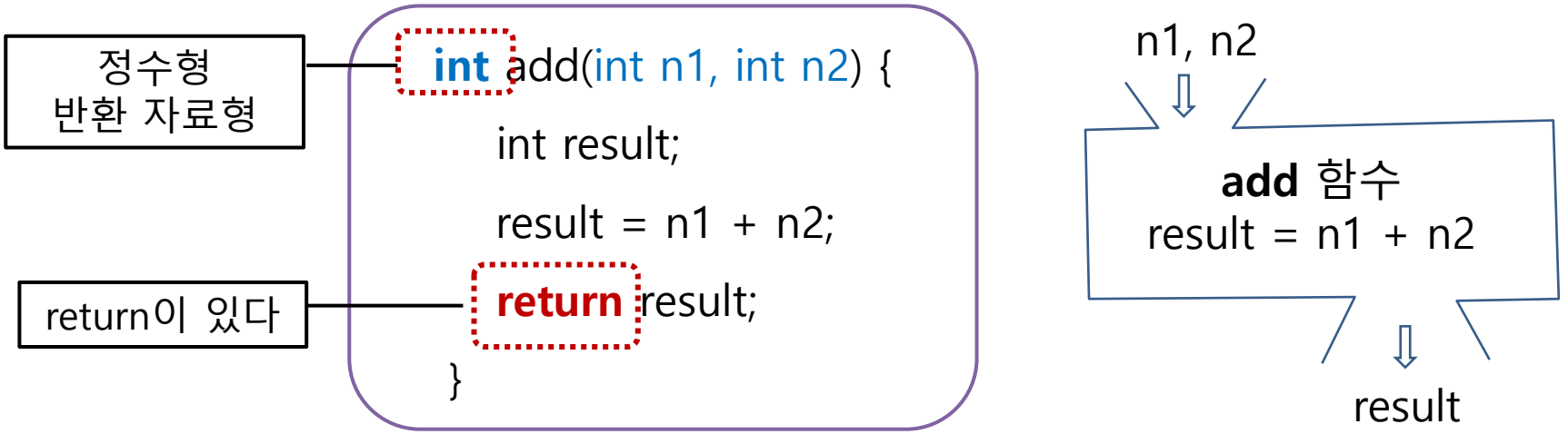
반환 자료형이 없다

메서드 이름

메서드(함수)의 유형

- 반환값이 있는 메서드(함수) – 예제 1

  반환값이 있는 경우 'return' 예약어를 사용해야 한다.



메소드(함수)의 사용

- 반환값이 있는 매서드(함수) – 예제1

```
public class OneUp {
    // 1을 더하는 함수 만들기
    public static int oneUp(int x) {
        x = x + 1;
        return x;
    }

    public static void main(String[] args) {

        int num = oneUp(10);    //oneUp() 호출

        System.out.println("반환값 : " + num);
    }
}
```

oneUp() 정의

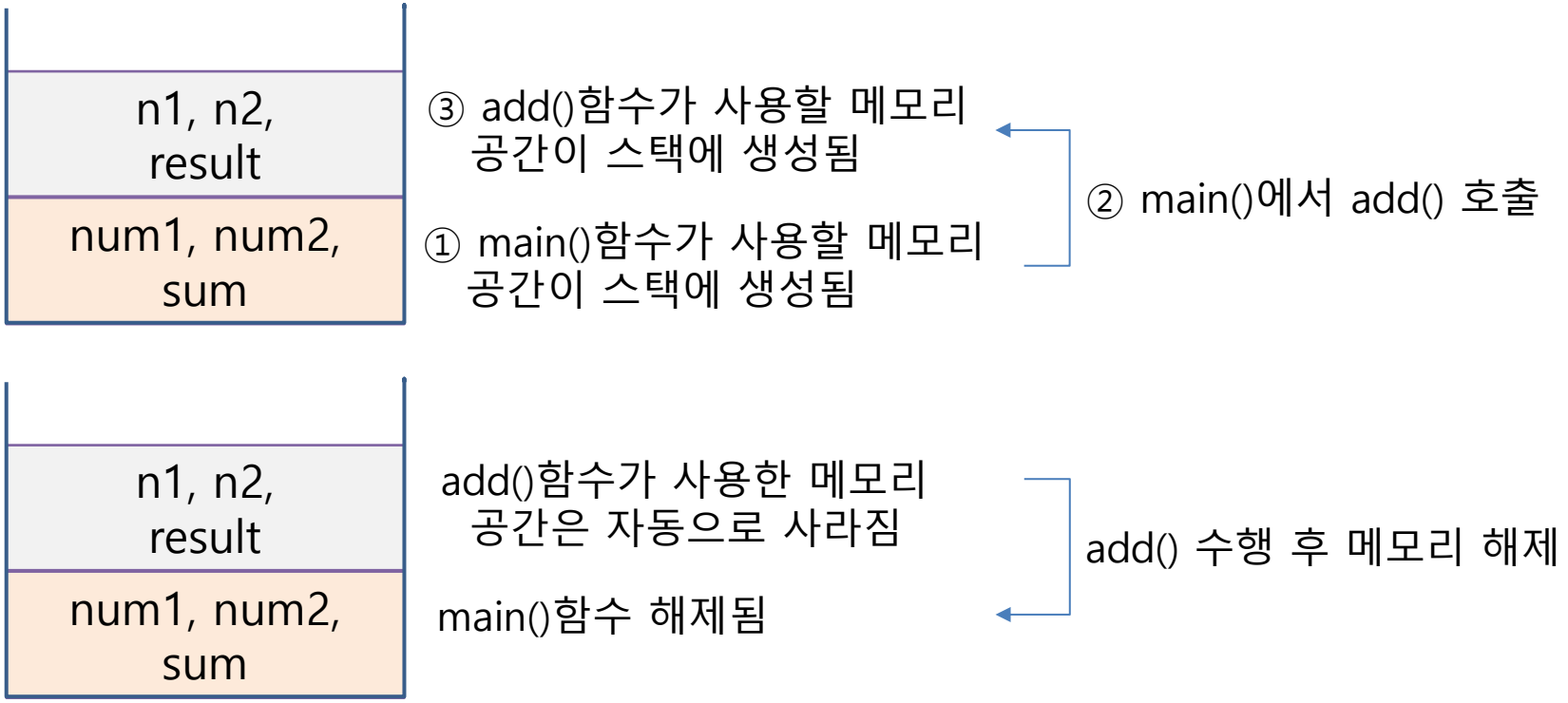
메소드(함수)의 사용

- 반환값이 있는 메서드(함수) – 예제2

```
public class Add {

    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;

        int sum = add(num1, num2);    //add() 호출

        System.out.println("합계 : " + sum);
    }

    //두 수를 더하는 메서드(함수) 정의
    public static int add(int n1, int n2) {
        return n1 + n2;
    }
}
```

함수 호출과 스택 메모리

- **함수 호출과 스택 메모리**
  - 함수가 호출될 때 사용하는 메모리 – **스택(stack),** 지역변수가 위치함
  - 함수의 수행이 끝나면 함수에 할당했던 메모리 공간이 해제됨.



변수의 유효 범위

- **변수의 유효 범위**
  - **지역 변수의 유효 범위**
    지역 변수는 함수나 메서드 내부에 선언하기 때문에 함수 밖에서는 사용할 수 없다. 지역변수가 생성되는 메모리를 **스택(stack)**이라 한다.
  - **멤버 변수의 유효 범위**
    멤버 변수는 인스턴스 변수라고도 한다. 클래스의 어느 메서드에서나 사용할 수 있다. 클래스가 생성될때 **힙(heap) 메모리**에 생성된다.
  - **static 변수의 유효 범위**
    사용자가 프로그램을 실행하면 메모리에 프로그램이 상주한다.
    new로 생성되지 않고 처음부터 **데이터 영역 메모리**에 생성된다.
    이 영역에는 상수, 문자열, static 변수가 생성된다.

변수의 범위(스코프)

- **지역변수의 범위**

  함수나 제어문에서 사용되며 호출 후에 메모리 공간에서 소멸(해제)한다.

```
public class OneUp {

    public static int oneUp(int x) {
        x = x + 1;
        return x;
    }

    public static void main(String[] args) {
        int num = oneUp(1);
        System.out.println("num= " + num);

        //System.out.println(x);
        //oneUp의 변수(지역변수) x는 호출된 후 소멸된다.
    }
}
```

변수의 범위(스코프)

- **정적변수의 범위**
  - 정적 변수는 **static** 키워드가 붙은 변수이며, 값이 공유된다.
  - 프로그램이 실행되면 메모리에 적재되고, 프로그램이 종료되면 메모리 공간에서 소멸한다.

```
public class OneUp2 {

    static int x = 1;

    public static int oneUp() {
        x = x + 1;
        return x;
    }

    public static void main(String[] args) {
        System.out.println(oneUp());  //2
        System.out.println(oneUp());  //3

        System.out.println(x);
        //x가 정적변수이므로 프로그램이 종료되어야 소멸된다.
    }
}
```

메서드(함수)로 배열을 전달

- 메서드의 매개변수로 배열을 전달

```
public class ArrayTest {

    public static int add(int[] score) {   // 배열이 매개변수
        int sum = 0;
        for(int i=0; i<score.length; i++) {
            sum += score[i];
        }
        return sum;
    }

    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4};
        int result = add(numbers);
        double avg = (double)result / numbers.length;

        System.out.println("합계 : " + result);
        System.out.println("평균 : " + avg);
    }
}
```

Math 클래스의 내장 매서드

- **static(정적) 매서드**

Module java.base
Package java.lang

Class Math

java.lang.Object
    java.lang.Math

```
public final class Math
extends Object
```

The class Math contains methods for performing

Fields

Modifier and Type	Field	Description
static double	E	The double value that is closer th
static double	PI	The double value that is closer th

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static double	abs(double a)	Returns the absolute
static float	abs(float a)	Returns the absolute
static int	abs(int a)	Returns the absolute
static long	abs(long a)	Returns the absolute

static 메서드이므로 클래스이름으로 직접 접근(new 객체 생성하지 않음)
사용 예) Math.abs(-4)

Math 클래스

- **Math 클래스의 주요 메서드 - 실습**

```java
int v1 = Math.abs(-10);       //절대값
System.out.println("v1 = " + v1);

long v2 = Math.round(5.6);    //반올림
System.out.println("v2 = " + v2);

double v3 = Math.floor(5.9);
System.out.println("v3 = " + v3); //버림

int max = Math.max(10, 20);
System.out.println("max = " + max); //최대값

double rand = Math.random();
System.out.println("rand = " + rand); //난수 값(0.0 <= rand <1.0)

int dice = (int)(Math.random()*6) + 1;
System.out.println("주사위 눈 : " + dice);
```

```
v1 = 10
v2 = 6
v3 = 5.0
max = 20
rand = 0.5067546025032655
주사위 눈 : 3
```

Math 클래스

- **Math.random() 사용하기**

```
System.out.println("=== 주사위 10번 던지기 ===");
for(int i=1; i < 11; i++) {
    int dice = (int) ((Math.random()*6)+1);
    System.out.print(dice + " ");
}
System.out.println();

System.out.println("=== 문자열 랜덤하게 뽑아내기 ===");
String[] word = {"나", "너", "우리", "세계", "우주"};
//System.out.println(word[0]);
//System.out.println(word[2]);
//word[rnd]  - 인덱스 변수 필요
int rnd = (int)(Math.random()*word.length);
System.out.println(word[rnd]);
```

```
=== 주사위 10번 던지기 ===
6 4 2 6 4 2 5 1 4 4
=== 문자열 랜덤하게 뽑아내기 ===
우주
```

Math 클래스

- **Lotto 복권 프로그램**

10 23 23 41 28 26

중복 번호 제거

이중 for(이차원배열)

6행 6열

```
// 로또 번호 생성하기
int[] lotto = new int[6];
int i = 0;
for(i=0; i<lotto.length; i++) {
    //로또 랜덤 번호 저장
    lotto[i] = (int) (Math.random()*45) + 1;
    //중복 검사(6행 6열)
    /*
     *
     **
     ***
     ****
     *****
     ******
    */
    for(int j=0; j<i; j++) {
        if(lotto[i] == lotto[j]) {
            i--;
            break;
        }
    }
}
System.out.println("로또 번호 생성");
//로또 번호 출력
for(i=0; i<lotto.length; i++) {
    System.out.print(lotto[i] + " ");
}
```

Math 클래스

영어 타자 게임 만들기

```
영어타자 게임, 준비되면 엔터

문제1
moon
moon
통과!
문제2
tree
tree
통과!
문제3
moon
```

```
오타! 다시 도전!
문제9
cow
cow
통과!
문제10
mountain
mountain
통과!
게임 소요 시간35초입니다.
```

Math 클래스

게임 방법

- 게임이 시작되면 영어 단어가 화면에 표시된다.
- 사용자는 최대한 빠르고 정확하게 입력해야 한다.
- 바르게 입력했으면 다음 문제로 넘어가고 "통과"를 출력한다.
- 오타가 있으면 같은 단어가 한 번 더 나온다.
- 타자 게임 시간을 측정한다.

Math 클래스

```
public static void main(String[] args) {
    String[] words = {"river", "mountain", "sky", "earth", "moon",
            "tree", "flower", "cow", "pig", "horse"};
    int n = 1; //문제 번호
    long start, end;
    Scanner scan = new Scanner(System.in);

    System.out.println("영어타자 게임, 준비되면 엔터");
    scan.nextLine();
    start = System.currentTimeMillis();   //게임시작 시간측정
    while(n < 11) {
        int rand = (int)(Math.random()*words.length);
        System.out.println("문제" + n);
        String question = words[rand];
        System.out.println(question);  //화면에 문제 표시

        String answer = scan.nextLine();
        if(answer.equals(question)) {  //대답이 질문과 같으면
            System.out.println("통과!");
            n++;  //통과하면 문제번호 1 증가
        }else {
            System.out.println("오타! 다시 도전!");
        }
    }
    end = System.currentTimeMillis();  //게임종료 시간측정
    System.out.println("게임 소요 시간은 " + (end-start)/1000 + "초입니다.");
    scan.close();
}
```