6강

JAVA_PROGRAMMING

Method

❖ 특정한 기능을 정의해 놓은 명령의 집합

- 복잡한 동작을 기능별로 정의하여 편리하게 사용 및 관리가 가능하다.

```
접근제한자 [지정예약어] 반환자료형 메서드명 (매개변수들) [throws 예외클래스들]
{
        내용;
        내용;
        …
        내용;
        return //필요 시 사용
}
        ( [..]부분은 필요 시 사용 )
```

```
public static void main(String []args) throws IOException
{
        System.out.println("Hello JAVA!!");
}
```

Method 정의 및 설명

- ❖ 메서드 명(add)
  - 사용자가 임의로 정하는 명칭
  - 명명법 고려 및 기능을 명시
- ❖ 매개변수 (int a, int b)
  - 메서드가 동작하기 위해 필요한 값을 받는 변수
  - 매개변수에 따라 **오버로딩** 가능
- ❖ 리턴 (return sum;)
  - 메서드의 종료
  - 메서드의 결과 값 반환
  - 반환 값과 반환 자료형은 일치
  - ex) sum은 int형이다.
- ❖ 예외클래스 (throws IOException)
  - 예외 처리가 필요할 경우 예외를 처리하는 클래스를 선언
  - ex) 입출력 예외 전가

```
public class test{
    public static int add (int a, int b){
        int sum = a + b;
        return sum;
    }
    public static void main(String []args){
        System.out.println("1 + 2 = "+add(1,2));
    }
}
```

예제

❖ 다음 예제에서 비효율적인 동작은 무엇인가?

```
public class test{
    public static void main(String []args){
        System.out.println("JAVA Programing!");
        System.out.println("1 + 2 = "+(1+2));
        System.out.println("JAVA Programing!");
        System.out.println("2 × 4 = "+(2*4));
        System.out.println("JAVA Programing!");
        System.out.println("10 ÷ 5 = "+(10/5));
    }
}
```

- 동일한 작업을 계속 코드로 작성할 경우 상당히 비효율 적이다.
- 이런 경우에 메서드로 정의하여 필요할 때 호출만 하여 사용한다면?

예제

❖ 다음과 같이 method를 정의

- 필요한 부분에서 호출만 하여 사용할 수 있으므로 보다 효율적이다.
- 기능 수정이 필요할 경우 메서드만 수정하면 된다.

```
public class test{
    public static void javaPrint(){
        System.out.println("JAVA Programing!");
    }
    public static void main(String []args){
        javaPrint();
        System.out.println("1 + 2 = "+(1+2));
        javaPrint();
        System.out.println("2 × 4 = "+(2*4));
        javaPrint();
        System.out.println("10 ÷ 5 = "+(10/5));
    }
}
```

Method

❖ 메서드(Method)
- 메서드는 작업을 수행하기 위한 명령들의 집합이다.
- 어떤 값을 전달 받아 기능을 동작하고 결과를 돌려준다.
- 전달받는 입력 값과 돌려주는 결과 값은 없을 수도 있다.

❖ 메서드의 장점
- 복잡한 코드를 쉽게 줄일 수 있다.
- 관리가 용이하다.

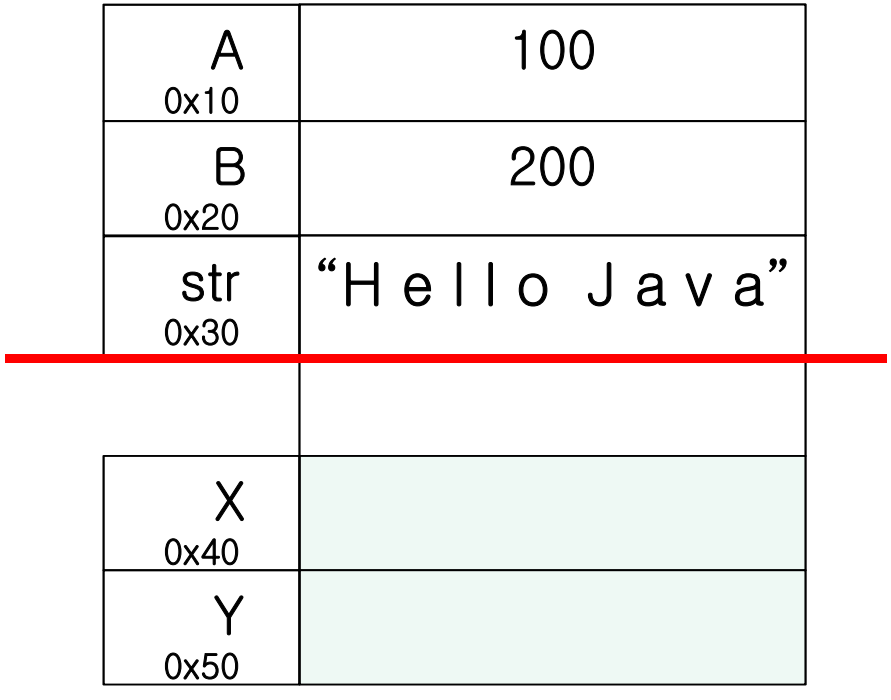
❖ 작성 시 고려
- 반복되는 특정 기능을 메서드로 정의한다.
- 하나의 메서드는 가능한 하나의 기능만 정의한다.

호출의 형태

❖ Call by name (이름에 의한 호출)
- 인자의 전달 없이 이름만으로 호출하는 형태
- ex) javaPrint();

❖ Call by value (값에 의한 호출)
- 데이터 값을 복사하여 넘겨주며 호출하는 형태
- ex) add(a,b);

❖ Call by reference (참조에 의한 호출)
- 참조변수(주소)를 전달하여 원본 데이터를 참조하여 호출하는 형태
- ex) System.out.println(str);



Method overloading

- 메서드 오버로딩(중복 정의)
  - 메서드의 이름은 같지만 구분 하여 호출 할 수 있는 기능
  - 동일한 기능의 메서드에 자료형만 다르게 해야 할 경우 유용
- 조건
  - 메서드가 호출될 때 전달 받는 매개변수로 구분
  - 개수, 순서 또는 자료형이 다르면 메서드의 이름이 같아도 구분이 가능

✓ 다음 코드에서 에러는?

```
void method(){}
void method( int  a ){}
void method( char  b ){}
void method( int  c ){}
void method( int  a, int  b ){}
```

함수 만들기 예제

❖ 두 수를 입력 받아 큰 수 출력 메서드 만들기

- ex) 첫 번째 수 : 10
  두 번째 수 : 20
  10 과 20 중 큰 수는 : 20

❖ 두 수를 입력 받아 두 수 사이의 합을 출력하는 메서드 만들기

- ex) 첫 번째 수 : 1
  두 번째 수 : 10
  1~10 사이의 합 : 55

Quiz

❖ 메서드 만들기
- 입력 받은 수가 홀수인지 짝수인지 반환하는 메서드
- 수를 입력 받아 1부터 입력 받은 수 까지 홀수의 합을 반환하는 메서드
- 문자를 입력 받아 대문자 <-> 소문자로 변환하는 메서드

❖ 메서드 오버로딩 활용하기
- 정수(int) 두 개를 입력 받아 더한 값을 반환하는 메서드
- 실수(float)두 개를 입력 받아 더한 값을 반환하는 메서드
- 정수와 실수 두 개를 입력 받아 더한 값을 반환하는 메서드
- 실수와 정수 두 개를 입력 받아 더한 값을 반환하는 메서드

❖ 사칙연산 계산기를 기능별로 메서드를 만들기
- 사칙연산 기능을 하나씩 나누어서 메서드로 정의
- 정의한 메서드를 이용하여 계산기 프로그램 작성하기

Quiz

❖ 메서드 만들기 추가 실습

- 인자로 N을 전달하면 N에 대한 절대값을 반환하는 함수
- cm값을 inch 값으로 반환하는 함수(1 Inch == 2.54cm)
- 삼각형의 넓이를 구하는 함수(밑변 * 높이 / 2)
- 파일의 용량(byte)을 매개변수로 입력 받아 bit단위로 반환하는 함수
  파일의 용량을 입력할 때 단위도 입력한다.(G, M, K)
  ex) byteToBit(32,'G');
  byteToBit(64,'M');
- 인자로 N을 전달하면 N에 대한 팩토리얼을 반환하는 함수(재귀)
- 인자로 N을 전달하면 거꾸로 만든 수를 반환하는 함수(재귀)