Day11

객체지향 (Object Oriented Programming)

- 클래스, 객체, 인스턴스, 상속, 인터페이스, 다형성, 추상화 등의 개념을 포함

계산기

class Calculator1 {

```
static int result = 0;

static int add(int num) {

  result += num;
  return result;

}
```

}

class Calculator2 {

```
static int result = 0;

static int add(int num) {

  result += num;
  return result;

}
```

}

public class Sample {

```
public static void main(String[] args) {


  System.out.println(Calculator1.add(3));
  System.out.println(Calculator1.add(4));
```

```
  System.out.println(Calculator2.add(1));
  System.out.println(Calculator2.add(2));
}
```

}

- 한계

만약 계산기를 10 개 ,100개 로 늘려야 한다면 또, 그 100개의 계산기의 기능을 수정해야 한다면 많은 시간과 노력이 들게 된다.

위와는 달리 하나의 클래스로 만든 객체 cal1, cal2 들은

다른 개성을 갖게되어 각각의 값을 유지하게 됨

추가기능이 필요할 경우 하나의 클래스에만 적용하면

모든 객체에서 동일하게 추가기능 사용

class Calculator {

```
int result = 0;

int add(int num) {

  result += num;
  return result;
}

int sub(int num) {

  result -=num;
  return result;
}
```

}

public class Sample {

```
public static void main(String[] args) {

  Calculator cal1 = new Calculator();
  Calculator cal2 = new Calculator();
```

```
  System.out.println(cal1.add(3));
  System.out.println(cal1.add(4));

  System.out.println(cal2.add(1));
  System.out.println(cal2.add(2));
}
```

}

객체와 인스턴스

- 객체와 인스턴스는 동일
- 인스턴스는 클래스와의 관계를 설명할때 사용
- 예를 들면 위에서 cal1객체와 cal2객체는 Calculator 클래스로 찍어낸 객체이다. 라는 표현보다는 cal1객체와 cal2객체는 Calculator 클래스의 인스턴스이다. 라는 표현이 더욱 어울림

클래스와 객체

Animal Class 작성

class Animal{

```
String name; // 객체변수, 인스턴스변수, 멤버변수

public void setName(String name) {

  this.name = name;

}
```

}

public class Sample1 {

```
public static void main(String[] args) {

  Animal cat = new Animal();
  // new는 새로운 객체로 생성할때 사용하는 키워드
  // cat 인스턴스는 Animal 클래스로 만들었다
  System.out.println(cat.name);
```

```
    cat.setName("boby");
    System.out.println(cat.name);

    System.out.println();

    Animal dog = new Animal();
    System.out.println(dog.name);
    dog.setName("honey");
    System.out.println(dog.name);

  }
```

}

- 결과를 확인해보면 name 객체 변수값은 공유되지 않고 각각 따로 유지되는 것을 확인할 수 있음
- 이부분이 가장 중요한 객체지향 개념
- 객체들의 변수 값들이 독립적으로 유지되는 것이 클래스의 존재의 이유 (static을 사용하여 공유하는 방법도 존재하긴함)
-

매서드(Method)

- 다른 언어에서는 함수라고 부르기도 하나 자바에서는 모든 것이 클래스 안에 존재하기에 매서드라는 표현만 사용함
- 매서드를 사용하는 이유

→ 똑같은 내용을 반복해서 처리하는일이 대부분

여러번 반복해서 사용하는 기능을 체계적으로 한번만 구성해 놓으면 필요할때마다 호출하여 편리하게 사용 가능

int sum(inta,intb){

return a+b;

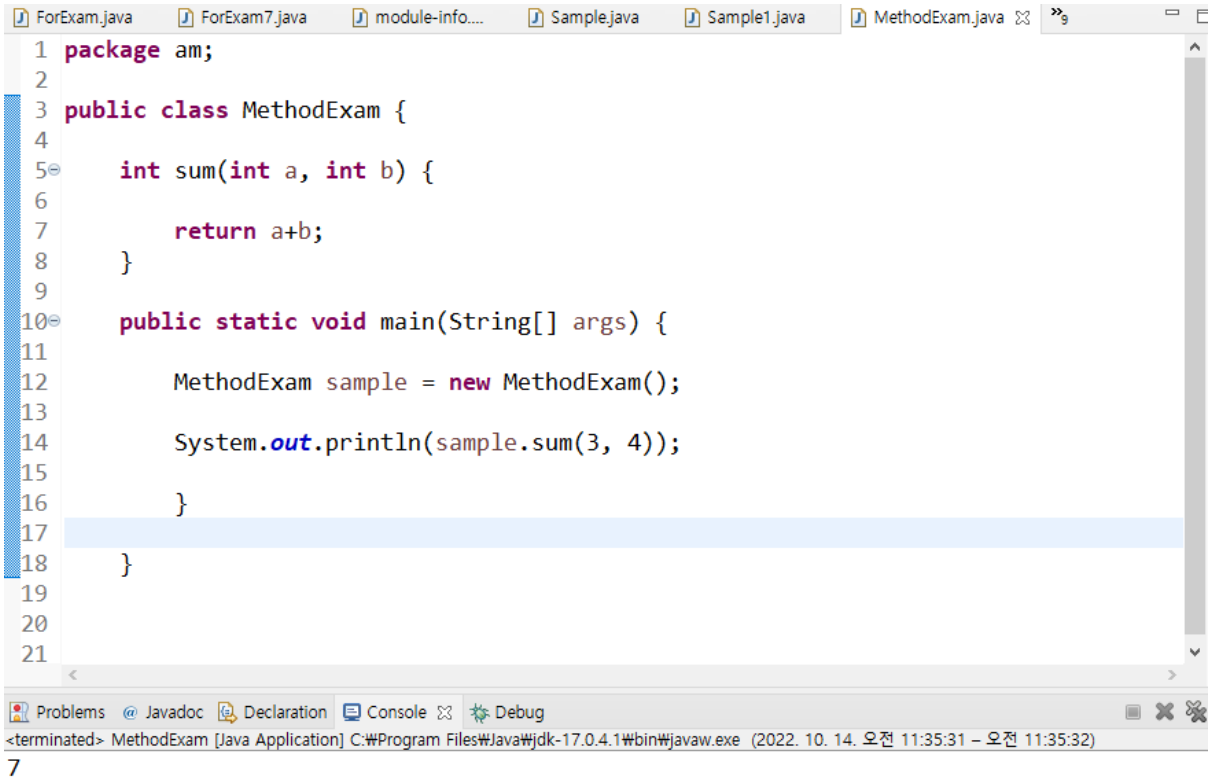
}

객체명.sum(2,3)

⇒ sum 매서드는 입력값으로 a,b를 받아 리턴값으로 두개의 값을 더한 결과를 돌려줌

public class MethodExam {

```
int sum(int a, int b) {

  return a+b;
}

public static void main(String[] args) {

  MethodExam sample = new MethodExam();

  System.out.println(sample.sum(3, 4));

  }

}
```



매개변수와 인수

- 매개변수(Parameter) : 매서드에 입력되는 전달 값을 받는 변수
- 인수(Argument) : 매서드를 호출할때 전달되는 값

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

  public static void main(String[] args) {

    MethodExam sample = new MethodExam();

    System.out.println(sample.sum(3, 4)//인수);

    }
```

- 매서드의 구조

```
int sum(int a, int b) {

    return a+b;
}
```

1) int : 리턴 자료형

2) sum : 매서드 이름

3) (int a, intb) : 매개변수

4) return a-b : 리턴값

매서드는 크게 4가지가 존재

1) 입력 출력 모두 O (가장 일반적임)

int sum(int a, int b) {

```
        return a+b;
}
```

2) 입력 출력 모두 X

```
void nomethod(){
    System.out.println(”인쇄”);
}
```

3) 입력 O 출력 X

```
void yesinput(String a){
    System.out.println(a+”님 환영합니다”);
}
```

4) 입력 X 출력 O

```
int nointput(){
    return 7;
}
```

실습

public class MethodTypeExam {

```
// 1. 입력 출력 모두 O
int inputout (int a, int b) {
  return a*b;
}

//2. 입력 출력 모두 X
void noinout() {
  System.out.println("환영합니다");
}

//3. 입력 O 출력 X

void yesinnoout(int a,String b) {
```

```
  System.out.println("나이:"+a+" 이름:"+b);
}

//4. 입력 X 출력 O

int nointyesout() {

  return 30;
}

public static void main(String[] args) {
  MethodTypeExam mt = new MethodTypeExam();

  //1. 입력 출력 모두 O
  System.out.println(mt.inputout(2, 3));

  //2. 입력 출력 모두 X
  mt.noinout();

  //3. 입력O 출력 X
  mt.yesinnoout(35, "홍길동");

  //4. 입력X 출력 O
  System.out.println(mt.nointyesout());

}
```

- return의 또 다른 용도

매서드를 강제로 빠져나가는 용도로 사용

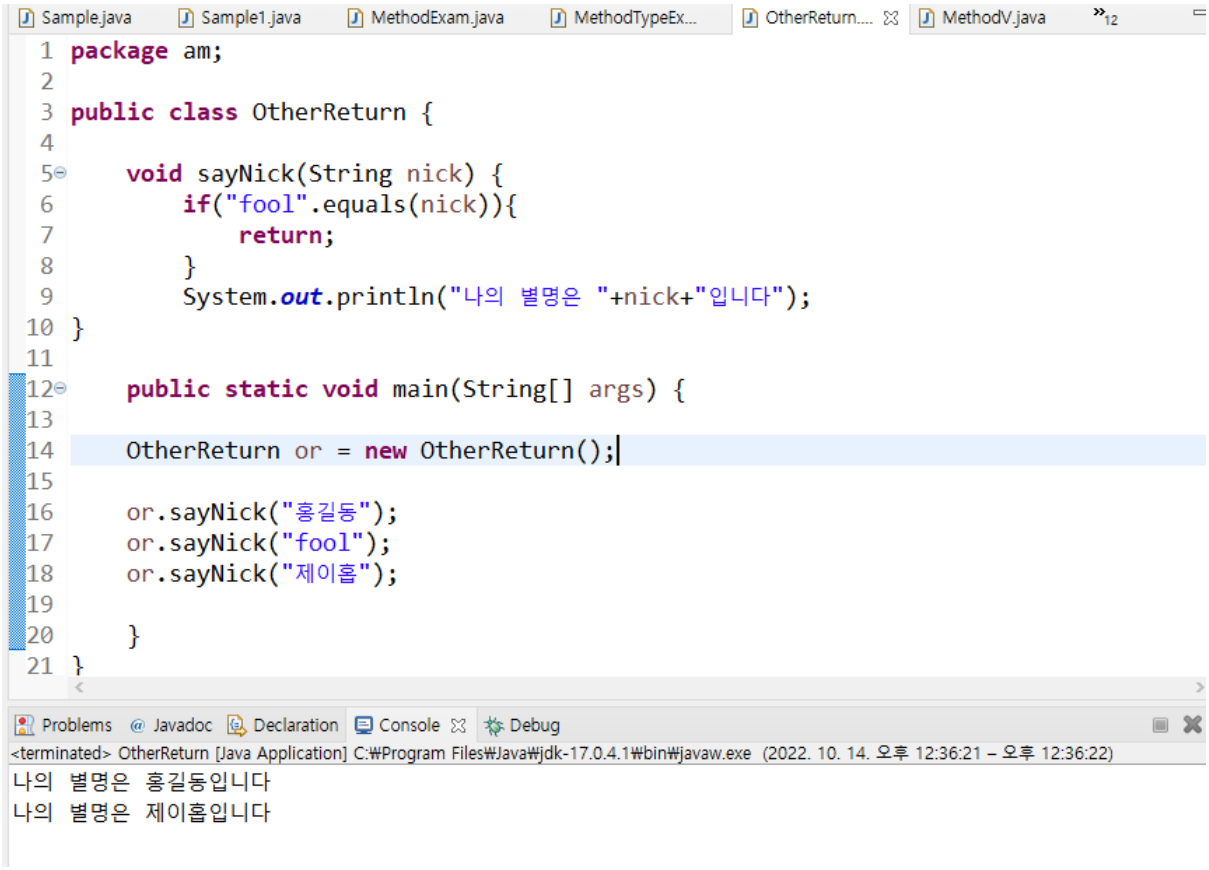
public class OtherReturn {

```
void sayNick(String nick) {
  if("fool".equals(nick)){
    return;
  }
  System.out.println("나의 별명은 "+nick+"입니다");
```

}

```
public static void main(String[] args) {

OtherReturn or = new OtherReturn();

or.sayNick("홍길동");
or.sayNick("fool");
or.sayNick("제이홉");

}
```

}

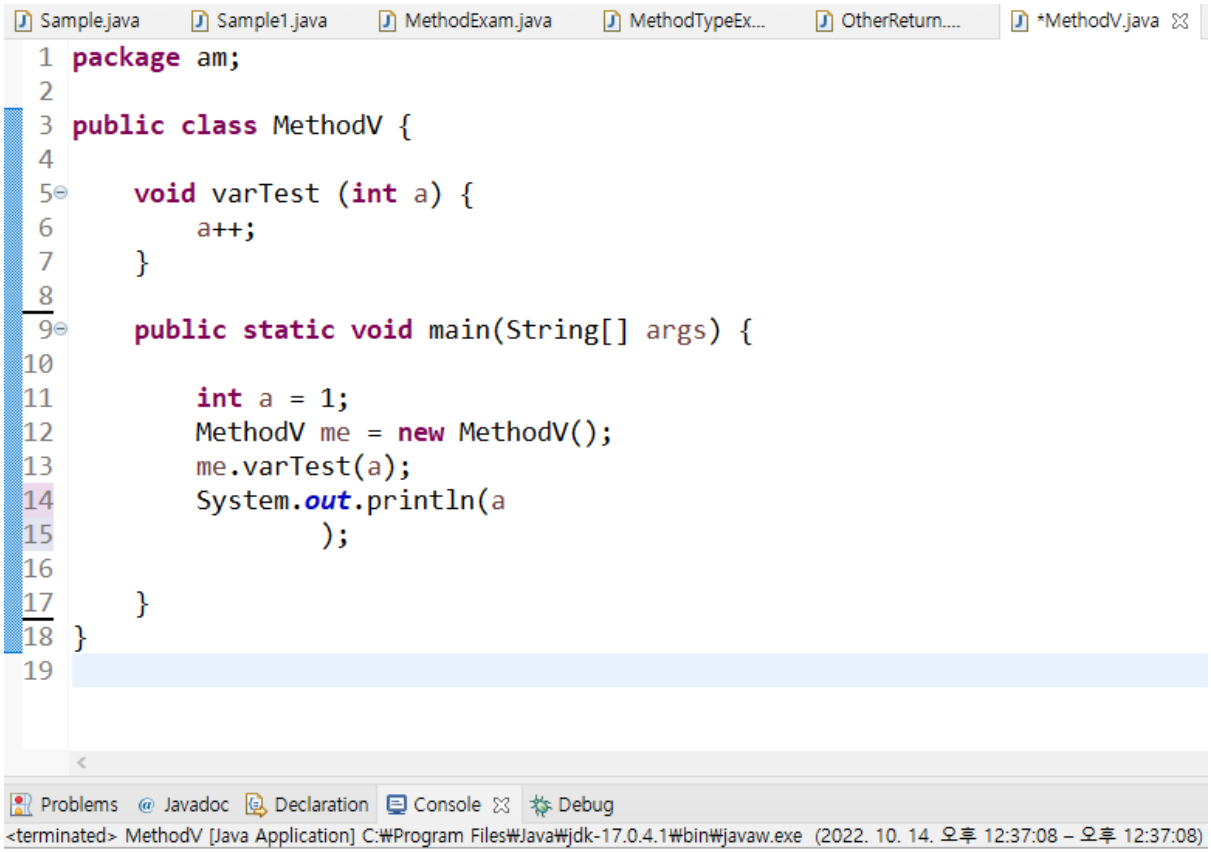


- 매서드 안에서의 변수

public class MethodV {

```
void varTest (int a) {
  a++;
}

public static void main(String[] args) {

  int a = 1;
  MethodV me = new MethodV();
  me.varTest(a);
  System.out.println(a
      );

}
```

}



return 이용하여 결과값을 가지고 나가는 방법

public class MethodV {

```
int varTest (int a) {
  a++;

  return a;
}

public static void main(String[] args) {

  int a = 1;
  MethodV me = new MethodV();
  me.varTest(a);
  System.out.println(a);
  System.out.println(me.varTest(a));
}
```

}

Sample.java | Sample1.java | MethodExam.java | MethodTypeEx... | OtherReturn.... | MethodV.java

```java
package am;

public class MethodV {

    int varTest (int a) {
        a++;

        return a;
    }

    public static void main(String[] args) {

        int a = 1;
        MethodV me = new MethodV();
        me.varTest(a);
        System.out.println(a);
        System.out.println(me.varTest(a));
    }
}

```

Problems | Javadoc | Declaration | Console | Debug

<terminated> MethodV [Java Application] C:₩Program Files₩Java₩jdk-17.0.4.1₩bin₩javaw.exe (2022. 10. 14. 오후 12:47:04 – 오후 12:47:05)

```
1
2
```