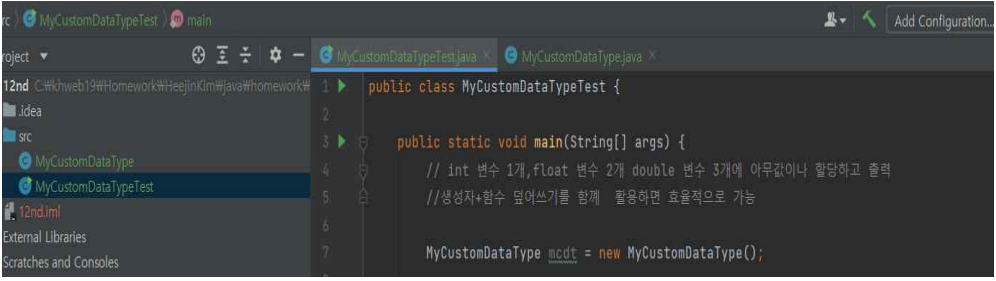
생성자



main class에 MyCustomDataType의 데이터 타입의 mcdt라는 객체를 만들어줌

```java
public class MyCustomDataTypeTest {

    public static void main(String[] args) {
        // int 변수 1개,float 변수 2개 double 변수 3개에 아무값이나 할당하고 출력
        //생성자+함수 덮어쓰기를 함께  활용하면 효율적으로 가능

        MyCustomDataType mcdt = new MyCustomDataType( intNum: 1, floatNum: 2, doubleNum: 3);//생성자 호출
        MyCustomDataType mcdt2 = new MyCustomDataType();//이건 아무것도 안넣었을 때 디폴트 값을 나옴
        //메서드의 이름이 같은데 서로 입력이 다르면 다른 것으로 인식함.
        MyCustomDataType mcdt3 = new MyCustomDataType( intNum: 1, floatNum: 2f, doubleNum: 3.0);//입력값만 적어넣어줌



        mcdt.allocRandom();//이제 이것을 만들어줌 랜덤값
        System.out.println(mcdt);

        MyCustomDataType mcdt4 = new MyCustomDataType( intNum: 2,MyCustomDataType.INT_PROC);
        mcdt4.allocIntRandom();
        System.out.println(mcdt4);

        MyCustomDataType mcdt5 = new MyCustomDataType( intNum: 3,MyCustomDataType.FLOAT_PROC);
        mcdt5.allocFloatRandom();
        System.out.println(mcdt5);
```

생성자 만드는 방법

1. public을 적는다.
2. 클래스 이름과 동일한 이름을 작성한다.
3. 매서드 작성하듯이 입력으로 사용할 정보들을 입력하도록 한다.
4. 중괄호 내부에 이 매서드(생성자)가 구동할 작업을 작성한다.

*** 생성자는 리턴 타입이 없다!

```
MyCustomDataType mcdt = new MyCustomDataType( intNum: 1, floatNum: 2, doubleNum: 3);//생성자 호출
MyCustomDataType mcdt2 = new MyCustomDataType();//이건 아무것도 안넣었을 때 디폴트 값을 나옴
//메서드의 이름이 같은데 서로 입력이 다르면 다른 것으로 인식함.
MyCustomDataType mcdt3 = new MyCustomDataType( intNum: 1, floatNum: 2f, doubleNum: 3.0);//입력값만 적어넣어줌


mcdt.allocRandom();//이제 이것을 만들어줌 랜덤값
System.out.println(mcdt);
```

1.MyCustomDataType의 데이터 타입의 mcdt라는 객체를 만들어주고
2.()안에 (int,int,int)인 생성자를 작성
3.mcdt객체 안에 있는 allocRandom()인 메소드를 호출
4.출력에 mcdt만 적으면 오버로드(toString)으로 나온 부분 출력 가능

```java
public MyCustomDataType(int intNum,int floatNum,int doubleNum){
    //생성자가 int int int 가 나온다

    intArr = new int[intNum]; //생성자를 만드는 부분 intNum에서 받아온 수를 객체의 intArr에 넣는다.
    floatArr = new float[floatNum];
    doubleArr = new double[doubleNum];

    System.out.println("나는 (int, int, int) 생성자!");
    setRange();
}
```

```java
public MyCustomDataType(){ //이건 디폴트
    System.out.println("나는 기본 생성자!");
}
```

```java
public  MyCustomDataType(int intNum,float floatNum,double doubleNum){
    System.out.println("나는 int, float,double 생성자");
}
```

정수형 int 타입 2개, 실수형 float 타입 3개, 실수형 double 타입3개 인 랜덤값을 도출하기 위해서는

1. 랜덤값의 범위를 지정한다.

--- setRange의 메소드안에 int타입과 float 타입의 범위를 지정

--- int타입과 float 타입의 범위는
각각 메소드로 정의

2.랜덤값을 할당하는 메소드를 만든다

--- int 타입, float 타입, double 타입을
따로따로 메소드로 만들어서

---랜덤값을 할당하는 전체 메소드 안에 집어넣는다

```
    }
public void setRange() {
    setIntRange();
    setRealRange();
      }
public void setIntRange(){        //int범위를 지정해줌
    intRange=INT_MAX-INT_MIN+1;
    }

public void setRealRange(){      //float범위를 지정해줌
        // 0.0 ~ 1.0 미만 - Math.random()
        // 4.75 + 4.75 -> 9.5
        // 0.0 ~ 9.5 미만 + 0.1
        // 0.1 ~ 9.6 미만 (9.5xx)
        // 그래서 최종적으로 ===> (-475 ~ 475) / 100
        // floatRange = 951
        floatRange = FLOAT_MAX * BIAS - FLOAT_MIN * BIAS + 1;
    }
```

4.75~ -4.75의 범위를 만들어주기 위해서
random는 0.0~0.999
따라서 -475~+475의 수를 /100으로 해서 딱맞게
값을 나오도록함

BIAS를 100으로 설정하여 만들어줌

```java
public void allocRandom(){

    allocIntRandom();
    allocFloatRandom();
    allocDoubleRandom();

}
public void allocIntRandom() {//int값을 랜덤으로 할당해줌
        for (int i = 0; i < intArr.length; i++) {
            intArr[i] = (int) (Math.random() * intRange + INT_MIN);
        }
    }
public void allocFloatRandom () {
     for (int i = 0; i < floatArr.length; i++) {
            // (0.0 ~ 950) + (-4.75f)//먼저 0.0~950을 백분위로 만들어주고 -4.7f를한다
            //float tmp = ((int) (Math.random() * floatRange)) / BIAS + FLOAT_MIN;
            //float로 만들어주고 받을때는 int로 만들어줌 ---이 전체를 bias로 나눔
            //이 식은 지금 계산이 이상함 그래서 bias처리는 따로 해줌

            float tmp = (int) (Math.random() * floatRange);

            tmp /= BIAS; //
            floatArr[i] = tmp + FLOAT_MIN;
        }
    }
public void allocDoubleRandom(){

            for (int i = 0; i < doubleArr.length; i++) {
```

2.랜덤값 생성 메소드

입력값이 들어온다고 하더라도 갯수는int라 데이터타입이 같다고 생각하여 오류를 발생시킴
입력값이 하나이기 때문에 정수,double처리할껀지 알 수 가 없어서 decision이라는 것을 통해서 알 수 있도록함

1. decision을 만들어서 switch로 정수,float,double을 구분시켜줌
2. Decision을 지정해줌

```
static final int INT_PROC = 1;
static final int FLOAT_PROC = 2;
static final int DOUBLE_PROC = 3;
```

3. 범위 메소드를 호출해줌

```java
public MyCustomDataType(int intNum,final int DECISION) {
    System.out.println("나는 (int,int)생성자");
    // this.intNum = intNum; 원래 여기서 해줘야하는데 같이 처리하려면 switch문에 같이 넣어준다
    System.out.println("DECISION:" + DECISION);
    decisionAlloc(intNum, DECISION);//여기서 DECISION을 넣어줌

}
//이 함수 안에서는 int arrNum으로 쓸것이다
public void decisionAlloc(int arrNum,final int DECISION) { //이 메소드는 (배열갯수,decision값)
    switch (DECISION) { //PROC에 쓴 값이 1,2,3이라 그 값을 도출
        case INT_PROC:
            intArr = new int[arrNum];
            setIntRange();
            break;

        case FLOAT_PROC:
            floatArr = new float[arrNum];
            setRealRange();
            break;
        case DOUBLE_PROC:
            doubleArr = new double[arrNum];
            setRealRange();
            break;

        default:
            System.out.println("올바른 값을 입력하세요" );
            break;
    }
```

이 메소드에서는 intNum으로 몇 개를 생성할것인지 받고 Decision을 입력해준다.

```
e( intNum: 2,MyCustomDataType.INT_PROC);

e( intNum: 3,MyCustomDataType.FLOAT_PROC);

e( intNum: 3,MyCustomDataType.DOUBLE_PROC);
```