7_22_2022__java_Operator

Step01_DataType\MainClass09.java

~Step03_Class\MainClass03.java

method가 return 된다 → method가 끝난다.

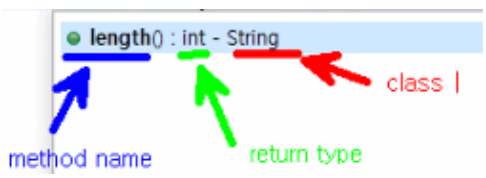
```
8    public static void main(String[] args) {
9
10       //콘솔창에 문자열 출력하기
11       System.out.println("Hello, World!!");
12   }
13 }
```

커서 하이라이트 → thread

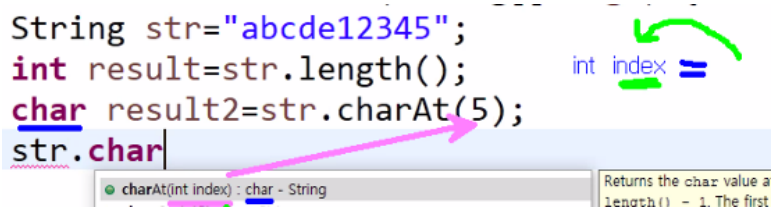
1project 1 main method

참조 데이터 타입

- 사물함 영역(heap)영역에 실체가 만들어 진다.
- 그 실체는 객체라고 부른다.
- 객체는 데이터의 저장소(field) + 기능(method)으로 이루어져 있다.
- 객체의 저장소(field)에는 java에서 다루는 다양한 data type이 들어있다.
- 그 type은 기본 data type 8 가지 혹은 참조data type이다.
- 참조값에 “.” 찍으면 해당 사물함을 찾아가 저장소나 기능을 불러온다.



  - ()괄호가 비어있는건 그 안에 어떤 인자를 전달하지 않아도 된다는 뜻.



- (ㅇㅇㅇ) 괄호 안에 들어가야 할 정보가 나온다.

- method 안에서 만드는 지역변수는 실행영역(stack)에 만들어진다.
- 지역변수는 method가 실행중에 만들어 졌다가 해당method가 종료(return)되면 사라진다.
- 키를 잃어버린 객체들을 garbage라고 한다. java에서 자동으로 수거해감 →낭비가 능

OPERATOR

- 산술연산자

  +, -, *, /, % 를 사용해 산술 연산이 가능하다

  정수끼리 연산하면 결과는 정수만 나온다.

  연산의 결과를 실수값을 얻어내고 싶으면 적어도 하나는 실수여야 한다.

```
package test.main;
public class MainClass01 {
   public static void main(String[] args) {
      int num1=5;
      int num2=2;

      int sum = 10 + 1;
      int sum2 = num1 + num2;
      int sum3 = 10 + num2;
      //정수끼리 연산하면 결과는 정수만 나온다.

      int result1 = 5 / 3;
      int result2 = num1 / num2;
      //연산의 결과로 실수 값을 얻어내고 싶으면 적어도 하나는 실수여야 한다.

      double result3 = 5 / 3.0;
      double result4 = num1 / (double)num2;

      //앞에 있는 수를 뒤에 있는 수로 나눈 나머지 값을 얻어낸다.
      int result5 = num1 % num2;
```

```
   }
}
```

- 증감 연산자

  ++, — 를 사용해 변수에 있는 숫자 값을 1씩 증가 혹은 1씩 감소 시킬 때 사용한다.

```
package test.main;
public class MainClass02 {
   public static void main(String[] args) {
      int num=0;
      num++;
      num++;
      num++; // num 이 최종적으로 3이 된다.

      int num2=0;
      num2--;
      num2--;
      num2--;   // num2 가 최종적으로 -3이 된다.

      int num3=0;
      int result1 = num3++; // result1 에 0 이 대입되고 num3 가 1 증가한다.

      int num4=0;
      int result2 = ++num4; // num4 가 1 증가하고 result2 에 1 이 대입된다.
   }
}
```

- 비교 연산자

  비교연산의 결과는 boolean type 을 리턴해준다.

  ==, !=, >, >=, <, <= 사용한다.

```
package test.main;
public class MainClass03 {
   public static void main(String[] args) {

      boolean result1 = 10 == 10; //true
      boolean result2 = 10 != 10; //false
      boolean result3 = 10 > 100; //false;
      boolean result4 = 10 >= 10; //true
      boolean result5 = 10 < 100; //true
      boolean result6 = 10 <= 10; //true

      // String type 변수 name 에 null 대입하기
      String name=null;
```

```
        // null 인지 아닌지 비교가 가능하다 (즉 java 에서 null 값은 비교가 가능)
        boolean result7 = name == null; //true
        boolean result8 = name != null; //false

    }
}
```

- 논리 연산자

  or 연산

  - “또는” 이라고 읽으면 된다.
    - 어느 하나만 true면 결과는 true. 모두다 false 일때만 결과는 false

```
boolean result1 = false || false; //false
boolean result2 = false || true; //true
boolean result3 = true || false; //true
boolean result4 = true || true; //true
```

  and 연산

  - “그리고” 라고 읽으면 된다.
    - 모두다 true 일때만 결과는 true. 어느 하나만 false면 결과는 false

```
boolean result5 = false && false; //false
boolean result6 = false && true; //false
boolean result7 = true && false; //false
boolean result8 = true && true; //true
```

  no 연산

  - “~가 아니면” 이라고 읽으면 된다.
    - boolean 값을 반전시킨다.

```
boolean result9 = !true; //false
boolean result19 = !false; //true
```

- 대입연산자

  +=, -=, *=, /=, %= 등을 사용 할 수 있다.

```
package test.main;

public class MainClass05 {
   public static void main(String[] args) {
      int num=10;
      //num = num+2;
      num += 2;
      //num = num-3;
      num -= 3;
      //num = num*4;
      num *= 4;
      //num = num/5;
      num /= 5;
      //num = num%6;
      num %= 6;
   }
}
```

- 3항 연산자

  피연산자를 세개 가지는 조건 연산자

  - 조건식? 반환값1 : 반환값2

    물음표(?) 앞의 조건식에 따라 결괏값이 true이면 반환값1 을 반환하고, 결과값이 false 이면 반환값2 를 반환한다.

```
package test.main;
public class MainClass06 {
   public static void main(String[] args) {
      boolean isWait=false;
      // isWait 이 true 면 "기다려요" 가 대입되고 false 면 "기다리지 않아요" 가 대입된다.
      String result = isWait ? "기다려요" : "기다리지 않아요";

      System.out.println(result);

      //위의 3항 연산자는 아래와 같이 if ~ else 문과 같은 로직이다.
      String result2=null;
      if(isWait) {
         result2="기다려요";
      }else {
         result2="기다리지 않아요";
      }
      System.out.println(result2);
   }
}
```

    : 의 양쪽의 data type은 같아야 한다.

클래스(CLASS)

```
public class ClassName{

}
```

객체의 설계도 역할

해당 클래스로 객체를 만들었을 때(new)어떤 field와 어떤 method를 가지게 할지를 설계할 수 있다.

data type 역할

지역번수나 필드를 만들 때 선언하는 data type 의 역할을 할 수 있다.

변수나 필드에 저장된 값의 사용설명서에 해당된다.

static field or ststic method를 포함하는 역할

필요에 따라서 객체에 field 나 method를 만들지 않고 class 자체에 만들어 놓을 수도 있다.

java 메모리 영역

- static

  class가 올라가는 영역

- stack

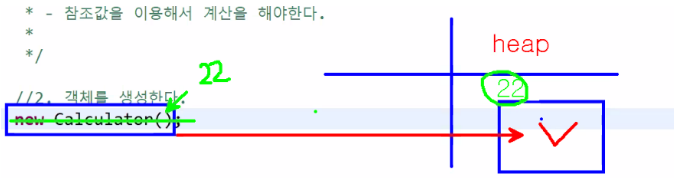
  local variable(지역변수)가 올라가는 영역

- heap

  object(객체)가 올라가는 영역

method 만들기

- 계산기를 만든다고 생각해 보면
- MainClass와 기능을 가진 Class가 필요하다
- 계산기능이 있는 Class를 Calculator라고 했을 때
- 만들기




```
public void exec() {
    System.out.println("계산해요!");
}
```

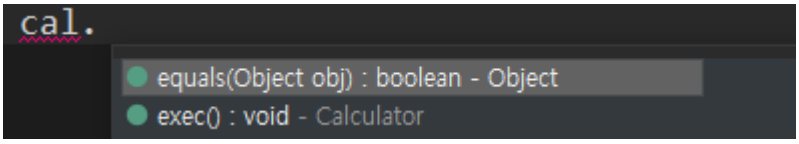
- 지역변수로 잡을 때 class명 자체를 타입으로 지을 수 있다.
- cal. 을 했을 때 Calculator Class 에 있는 기능도 사용 할 수 있다.

```
package test.main;

//1. 객체를 생성하는데 필요한 클래스 import
import test.mypac.Calculator;

public class MainClass01 {
public static void main(String[] args) {

//2. 객체를 생성한다.
  Calculator cal=new Calculator();
  cal.exec();
}
}
```



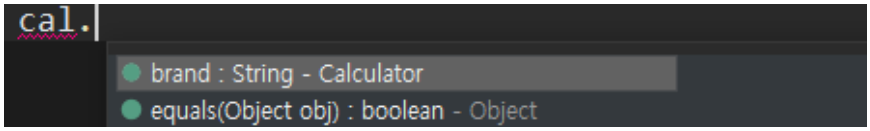
void 가 나오는 method는 동작하는 목적이 아니고 call 만 하는 method

field 만들기

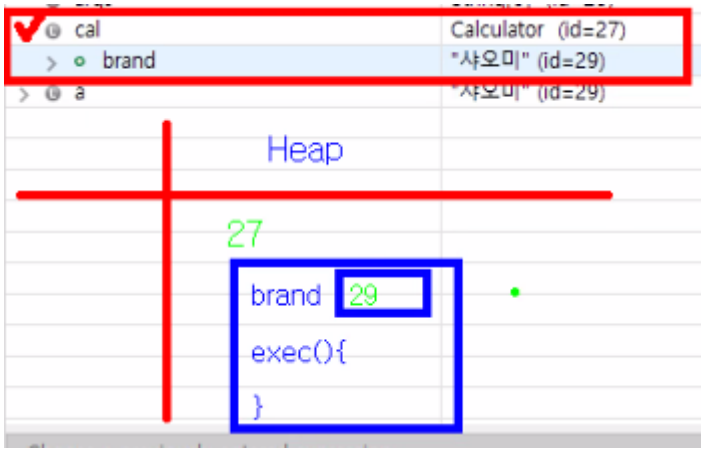
- 만들기

```
public String brand="세이코";
```

- field는 소괄호가 없다.



heap 영역 안에



▼ Calculate Class가 27번 키 안에 저장 되어있고

▼ brand 라는 필드와

- 안에 있는 정보는 29번 키에 저장 되어있다고 생각할 수 있다.

- exec라는 method가 있다

- 스택영역

---

field를 한번만 쓰고 버릴 수도 있다.

field는 =0, =null 할 필요 없이 초기값이 정해져 있다.

예제 Member 객체를 생성해서 지역변수에 담기

```
package test.mypac;

public class Member {
public int cNumber;
public String cName;
public String cAddr;
}
```

```
package test.main;

import test.mypac.Member;

public class MainClass03 {
  public static void main(String[] args) {
    System.out.println("main method가 시작되었습니다.");
    //Member 객체를 생성해서 그 참조값을 mem1 이라는 지역변수에 담아보세요.(1줄)
    Member mem1=new Member();
    mem1.cNumber=1;
    mem1.cName="두부";
    mem1.cAddr="페어펙스";


    // Member 객체를 생성해서 그 참조값을 mem2 이라는 지역변수에 담고 유키의 정보도 담아보세요
    Member mem2=new Member();
    mem2.cNumber=2;
```

```
    mem2.cName="유키";
    mem2.cAddr="하이데저트";
  }
}
```