2023.4.14.

Ⅰ.Ⅱ.Ⅲ.Ⅳ.Ⅴ.Ⅵ.Ⅶ.

<<요약>>

==========

Ⅰ. 자바 기본 문법

1. 기본 연산자
   : +, -, *, /(몫), %(나머지)
2. 복합 대입 연산자
   : num += 10; (num = num+10;)
3. 관계 연산자 :true 또는 false 결과만 나옴
   : >, <, >=, <=, ==, !=(같지 않으면 참)
     a=10; b=20;
     a > b(a가 b보다 큰지) -> false
4. 논리 연산자 : true & false (여러 식을 묶어 한번에 계산)
   : &&(and - 모든 값이 참일때만 참), ||(or - 하나라도 참이면 참), !(not - 결과 반전)
5. 증감 연산자 (결과적으로 자기 자신을 1 증가 또는 감소)
   : ++num, num++, --num. num--
     num = num+1
6. 비트 연산자
   << : 피연산자의 비트열을 왼쪽으로 이동시키며 이동에 따른 빈공간은 0으로 채움.
   >> : 피연산자의 비트열을 오른쪽으로 이동시키며, 이동에 따른 빈공간은
        음수의 경우엔 1, 양수의 경우엔 0으로 채움.
   >>> : 피연산자의 비트열을 오른쪽으로 이동시키며, 이동에 따른 빈공간은 0으로 채움.

int num = 2 << 1
를 놓고 봤을때 2의 비트 열을 왼쪽으로 1만큼 이동하고 이에 해당하는 정수값을 변수 num에 저장하는 것이다.
즉 2의 비트열인 00000010에 1을 왼쪽으로 1만큼 이동하는 것이며 00000100이 되고 정수 4가 되게 된다.
마찬가지로 2를 넣을때는 00001000 으로 8이 3만큼 이동할때는 00010000이 되어 16이 되며,
2의 배수 곱만큼 값이 변화하게 된다.

(2x2 = 4, 2x4 = 8, 2x8 = 16....)

***2진수 정수의 특성상 왼쪽으로 한칸씩 비트열을 움직이면 2의 배수 곱으로 오른쪽으로 한 칸 씩 이동시키면 2의 배수 나눗셈이 되게 된다.
해당 내용이 중요한 이유는 곱셈 및 나눗셈 처리시에 CPU 부담을 줄일 수 있다는 것에 있다.

7. 삼항 연산자(조건 연산자)

변수 = (조건식) ? 참인경우 실행 // : 거짓인 경우 실행 ;

Ex01.java Ex02.java

```
1  package day03;
2
3  public class Ex02 {
4
5      public static void main(String[] args) {
6          /*
7           삼항 연산자(조건 연산자)
8            변수 = (조건식) ? 참인경우 실행 : 거짓인 경우 실행 ;
9           */
10
11         int su =8;
12
13         String s= (su%2==0)?"짝수" :"홀수";
14         System.out.println(su +" : "+s);
15
16         s= (su%3==0)?"3의 배수" :"3의 배수 아님";
17         System.out.println(s);
18
```

Problems Javadoc Declaration Search Console

<terminated> Ex02 (1) [Java Application] D:₩JAVA₩eclipse₩plugins₩org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v2

```
8 : 짝수
3의 배수 아님
```

**연산자 우선 순위

연산자	연산 방향	우선순위
증감(++, --), 부호(+, -), 논리(!)	←	높음
산술(*, /, %)	→	
산술(+, -)	→	
비교(<, >), (<=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리(¦)	→	
논리(&&)	→	
논리(¦¦)	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=)	←	낮음

** 비트 연산자

비트 연산자	설명
&	대응되는 비트가 모두 1이면 1을 반환함. (비트 AND 연산)
|	대응되는 비트 중에서 하나라도 1이면 1을 반환함. (비트 OR 연산)
^	대응되는 비트가 서로 다르면 1을 반환함. (비트 XOR 연산)
~	비트를 1이면 0으로, 0이면 1로 반전시킴. (비트 NOT 연산, 1의 보수)
<<	명시된 수만큼 비트들을 전부 왼쪽으로 이동시킴. (left shift 연산)
>>	부호를 유지하면서 지정한 수만큼 비트를 전부 오른쪽으로 이동시킴. (right shift 연산)
>>>	지정한 수만큼 비트를 전부 오른쪽으로 이동시키며, 새로운 비트는 전부 0이 됨.

Ⅱ. if문

제어문 : 프로그램의 흐름을 제어하는 것

```
if(조건식)
    종속문장 (종속문장 하나일 때만 사용가능)
if(조건식){
    종속문장
}
```

**리팩토링

Ⅲ.

Ⅳ. Ⅴ. Ⅵ. Ⅶ.