5일차
디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용SW개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 - namkyo Kim
siary11@naver.com

```java
public class BitAndTest {
    public static void main(String[] args) {
        int num1 = 10, num2 = 8;



        // & 이 비트 연산자 AND
        // 관계 연산자에서는 && 형태로 존재했음

        //ex        10101011001100
        //          01110111011100 AND
        //-----------------------------------
        //          00100011001100
        //          2^12 2^8 2^7 2^4 2^3
        //              4504??                  노가다로 해보고 요령 조금 알아서 파악한듯!

        // 10 ===> 1010
        // 8 ===> 1000 AND
        // ---------------
        // 8 ===> 1000
        System.out.printf("%d And %d = %d\n ",num1,num2,num1 & num2);

        num2 = 138;

        // 138 ===> 100001010
        // 10 ===>      1010 AND
        //-----------------------
        // 10 ===> 00001010

        System.out.printf("%d And %d = %d\n ",num1,num2,num1 & num2);
    }
}
```

```java
public class BitOrTest {
    public static void main(String[] args) {
        int num1 = 10, num2 = 5;



        // | 이 비트 연산자 OR
        // 관계 연산자에서는 || 형태로 존재했음
        // 10 ===> 1010
        // 5 ===>  0101 OR
        // ----------------
        // 15 ===> 1111



        System.out.printf("%d OR %d = %d\n ",num1,num2,num1 | num2);

        num2 = 136;

        // 10 ===>   00001010
        // 136 ===>  10001000 OR
        // ----------------
        // 138 ===> 10001010

        System.out.printf("%d OR %d = %d\n ",num1,num2,num1 | num2);

        //*************** OR 연산은 합집합 개념, AND 연산은 교집합 개념 ***************
        //                  이거 보고 무슨느낌인지 팍 와닿았음 이런개념으로 생각하기
    }
}
```

```java
public class BitShiftTest {
    public static void main(String[] args) {
        int num1 = 2, num2 = 5, num3 = 10;

        // 2^1 x 2^5 = 2^6(64)
        System.out.printf("%d << %d = %d\n",num1,num2, num1 << num2);
        // 5 x 2^5 = 160
        System.out.printf("%d << %d = %d\n",num2,num2, num2 << num2);
        // 10 x 2^5 = 320
        System.out.printf("%d << %d = %d\n",num3,num2, num3 << num2);

        // 2^1 x 2^2 = 2^3(8)
        System.out.printf("%d << %d = %d\n",num1,num1, num1 << num3);
        // 5 x 2^2 = 20
        System.out.printf("%d << %d = %d\n",num2,num1, num2 << num1);
        // 10 x 2^2 = 40
        System.out.printf("%d << %d = %d\n",num3,num1, num3 << num1);

        // 2^1 x 2^16 = 2^11(2048)
        System.out.printf("%d << %d = %d\n",num1,num3, num1 << num3);
        // 5 x 2^10 = 5120
        System.out.printf("%d << %d = %d\n",num2,num3, num2 << num3);
        // 10 x 2^10 = 10240
        System.out.printf("%d << %d = %d\n",num3,num3, num3 << num3);



        // 이유 :
        // 0101 ----> 5
        // 0001 ----> 1

        // 종합적인 결론:
        // 쉬프트연산은 2^n은 곱하거나 나눈다.
        // 쉬프트 연산은 정수형끼리밖에 안된다.
        //-------------------------------------궁금점-----------------------------------------------
        // 정수 끼리면 대표적으로 int 가 해당되는건가 싶다..   쓰는걸 많이 보진 못했지만 long, short 도 정수 인데
        //-----------------------------------------------------------------------------------------


        // 최근에 나온 휴대폰 전용 ARM 프로세서에서는 소수점에 대한 쉬프트연산을 지원하기도한다.
    }
}
```

질문입니다.

Git ▶ Run TODO Problems Debug Terminal Build

```java
public class InterruptComment {
    public static void main(String[] args) throws InterruptedException {
        for(int i = 0;; i++){
            if(i % 2 == 0){
                System.out.println("짝수");
            }else {
                System.out.println("홀수");
            }

            Thread.sleep(500);
        }
    }
}

// Interrupt: 인터럽트란 무엇인가?
// 사실 인터럽터라는 용어는 하드웨어 개발자들이 주로 사용하는 단어다.
// 보통 자바나 GUI 개발자들 혹은 애플리케이션 개발자들은 이벤트라고 표현한다.
// 격국 이벤트와 인터럽트는 동의어란 뜻이다.
// 그렇다면 인터럽트라고 부르지말고 이벤트라고 불러보자
// 이벤트는 뭘까>?
```

```java
// 연인끼리 100 일 이벤트를 챙긴다고 해보자
// 100일은 사실 어쩌면 존재할 수도 있고 어쩌면 존재하지 않을수도 있다.

// 몰컴을 예로 들기
// 전제조건 : 부모님이 안방에서 주무시고 계시고
//           나는 자는척하다 일어나서 컴퓨터를 켠 후 게임 시작
// 1. 몰래 컴터를 하고있었고 게임중이였음
// 2. 갑자기 안방의 문이 열렸음. <<< --- 이벤트(인터럽트)
// 빨리 자는척 해야함
// ---> 문을 열어 보시고 걸리면 끝
// ---> 이게 아니라면 문을 열어 보시고 잘못 들었나 ? 하고 화장실 갔다가 다시 주무시러 방에감
// ---> 부모님이 방에 다시 들어가시면 컴퓨터를 키고 다시 게임을 복귀
// 3. 이후부터 다시 우리시간

// 기본적으로 이벤트라는 것은 최우선적으로 처리해야 하는 작업으로 *
// 어떤 작업보다 우선순위가 높은 녀석들입니다. *
// 마찬가지로 여기서 Thread.sleep()하는 작업도 일종의 이벤트(인터럽트)에 해당.
// 그러므로 이 작업이 시작되면 다른 모든 작업을 제쳐두고 이것을 최우선적으로 처리
// 물론 조금 더 정확한 것은 CPU의 동작관 Thread의 동작 과정에 대해 설명할 때 자세히 기술하도록함

// 결국 Thread.sleep(500)이 가장 중요한 작업이므로
// 이 작업을 완료하기 전까지는 어떠한 작업도 수행하지 않는다는 뜻
// 그래서 0.5초동안은 무조건적인 대기!
```

```java
public class Quiz {
    public static void main(String[] args) {
        // 2^1 + 2^3 + 2^5 + 2^6 을 2진수로 표현해보자!
        // 1101010

        // 10^6     10^5    10^4    10^3    10^2    10^1    10^0
        //  1        0       0       3       0       0       0


        // 2^0 + 2^3 + 2^6 + 2^7 + 2^8 + 2^9 을 2진수로 표현해보자!
        // 1111001001

        //  2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
        //   1   1   1   1   0   0   1   0   0   1



    }
}
```

```java
public class NonDuplicateWithoutArrayTest {
    // 0 ~ 9까지의 숫자가 중복되지 않게 출력되게 만들어보자!(배열 없이)
    public static void main(String[] args) throws InterruptedException {
        final int BIN = 1;

        // 2진 비트 연산자 AND와 OR 연산자를 활용함
        // 또한 쉬프트 연산자를 함께 활용해서 각각의 비트를 채우는 형식으로 코드 구현

        int testBit = 0;
        int randNum;

        // 2진수에 대한 이해가 필요함( 이 문제를 다루기 위해)
        // 십진수 10을 이진수로 변환해보자
        // 변환절차
        // 1. 10에서 가장 근접하면서 10보다 작은 2^n을 찾는다 = 8
        // 2. 찾은 숫자 10 에서 8을 뺀 값인 2를 적는다.
        // 3. 값이 0이 나올때 까지 이 절차를 반복한다.
        // 4. 0이 된이후 뺏던 값들의 2^n에 해당하는 n 값들을 열거한다 ===> 3, 1
        // 5. 구한 숫자들이 각각 이진수의 자리수에 해당한다.
        // 6.       2^3     2^2     2^1     2^0
        //           1       0       1       0
        // 7. 검산 ===> (2^3 x 1) + ( 2^2 x 0 ) + (2^1 x 1) + (2^0 x 0 ) = 10

        for (int i = 0; i < 10; i++) {
            // 0 ~ 9 나오게 * 10 설정
            randNum = (int) (Math.random() * 10);

            // 나온 randNum 에 대한 중복판정을 어떻게 할 것 인가? bit 연산
            // 2^9  2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
            //   1    1   1   1   1   1   1   1  1   1

            // BIN = 2로 시작하면 2^1 에서 시작하는 것이라
            // 맨 마지막값이 무조건 2^1 x 2^9이 되서 1024가 더해지므로 논리적 오류 발생
```

```
        // 맨 마지막값이 무조건 2^1 x 2^9이 되서 1024가 더해지므로 논리적 오류 발생
        // 그러므로 BIN = 1 로 만들어서 2^0 x 2^9 으로 512가 되게 처리해야 정상적인 결과값 출력

        // 쉬프트 연산의 결과 ( 비트를 왼쪽으로 이동시킨다)
        // 1 << 2 ===> 2^0 x 2^2 = 4    (비트를 왼쪽으로 2칸 이동시킴)
        // 1 << 4 ===> 2^0 x 2^4 = 16   (비트를 왼쪽으로 4칸 이동시킴)
        // 1 << 8 ===> 2^0 x 2^8 = 256  (비트를 왼쪽으로 8칸 이동시킴)
        // 1 >> 9 ===> 2^0 x 2^9 = 512  (비트를 왼쪽으로 9칸 이동시킴)

        //       10000   (1이라는 숫자를 왼쪽으로 2칸 이동시키면?)
        //     1000000   ( 10^2 이 곱해진다 )
        //      10000    (1이라는 숫자를 왼쪽으로 4칸 이동시키면 ?)
        //  100000000    (10^4 이 곱해진다)

        // 십진수기 때문에 위치에 이동할 때마다 10 씩 곱해진다.

        // 이 부번은 중복이 있는지 없는지 검사하는 루틴
        // testBit는 int형이니까 전체가 4바이트(32바이트)
        // ex) 4, 5, 4, 1

        while ((testBit & (BIN << randNum)) != 0) {
            randNum = (int) (Math.random() * 10);
        }
        System.out.printf("ranNum = %d\n", randNum);

        // 실제 자리수 셋팅은 여기서한다
        // A += B ===> A = A + B
        // 위아래는 다른 의미 스타일만 같다.
        // A |= B ===> A = A | B
        testBit |= (BIN << randNum);

    }
    System.out.println("testBit의 최종값은 1023이다. 진짜? " + testBit);
```

```
//21 -----> 16(2^4) + 4(2^2) + 1(2^0)
//            10101
// 1, 3, 5 번째 비트지만
// 실제 표현할때는 0번 비트 , 2번 비트, 4번비트로 표현해주도록 한다.

// 73 ---3 64(2^6) + 8(2^3) + 1(2^0)
//                  1001001
// 0번 비트 , 3번 비트, 6번 비트로 표현됨

// 2^9  2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
//  1    0   0   0   0   0   0   0   0   0  2^9(512)
//  0    1   0   0   0   0   0   0   0   0  2^8(256)
//  0    0   1   0   0   0   0   0   0   0  2^7(128)
//  0    0   0   1   0   0   0   0   0   0  2^6(64)
//  0    0   0   0   1   0   0   0   0   0  2^5(32)
//  0    0   0   0   0   1   0   0   0   0  2^4(16)
//  0    0   0   0   0   0   1   0   0   0  2^3(8)
//  0    0   0   0   0   0   0   1   0   0  2^2(4)
//  0    0   0   0   0   0   0   0   1   0  2^1(2)
//  0    0   0   0   0   0   0   0   0   1  2^0(1)

// 관계연산자 AND와 비트연산자 AND는 서로 동작방식이 약간의 차이가 있다.
// 십진수 10과 십진수 5의 AND 연산은 아래와 같이 이루어진다.

// 1010 ------ 10
// 0101 ------ 5  AND
//------------------
//0000--------- 0

// 1010 ------ 10
// 0101 ------ 5  OR
```

```
6
7   // 비트 연산자 AND는 각 비트의 자리수가 1(참)인 녀석들 끼리만 1(참)이 된다.
8   // 하나라도 0(거짓)이 있으면 해당 자리수는 0이 된다.
9
0   // 비트 연산자 OR는 각 비트의 자리수중 하나라도 1(참)이 있으면 1(참)이 된다.
1   // 양쪽 모두 0(거짓)을 가지고 있는 경우에만 0(거짓)이 된다.
2
3   // Q : bit OR 연산은 덧셈인가요?
4   // A :
5
6   // 1010 - 10
7   // 0111 - 7 OR
8   //--------------
9   //1111- 15==== 8 + 4 + 2 + 1
0   //10000 (2^4) - 1 = 1111(2) = 15
1
2   // 2^0 + 2^1 + 2^2 + ..... + 2^n = 2^(n+1) - 1
3
4   // 2진수 10101000 을 10진수로 바꿔보자!
5   // 2^7 + 2^5 + 2^3 = 8 + 32 + 128 = 168
6
7   // 2진수 11111100 을 10 진수로 바꿔보자
8   // 11111111 = 2^0 + 2^1 + ..... 2^7 => 2^8 - 1 = 256 - 1 = 255
9   // 255 -3 = 252
```

```java
public class HomeWork {
    public static void main(String[] args) throws InterruptedException {
        /*  앞서서 22번의 경우엔 난수의 범위가 0 ~ 9 였다.
            이번에는 2개의 범위를 가지는 난수 2개를 중복없이 제어해보자!
            하나는 5 ~ 10의 범위를 가지고 다른 하나는 7 ~ 10의 범위를 가진다.
            22번 예제를 응용하여 풀 수 있는 문제고 다소 난이도가 높은 문제다.
            ex) 5, 6, 7, 8, 9, 10이 모두 출력되어야 하고 또한 7, 8, 9, 10이 출력되어야 한다.
              여기서 중복이 발생하면 안됨

         */

        final int BIN1 = 1;
        final int BIN2 = 1;


        int randNum1;
        int randNum2;
        int Bit1= 0, Bit2 = 0;


        for(int i = 1; i < 10;i++){

            // 5 ~ 10 까지 설정
            // (Math.random() * (10 - 5 + 1)+5)
            //                 (최대값 - 최소값 + 1)+5
            randNum1 = (int) (Math.random() * (10 - 5 + 1)+5);

            while ((Bit1 &  (BIN1 << randNum1)) != 0){
                randNum1 = (int) (Math.random() * (10 - 5 + 1)+5);
            }
            System.out.printf("ranNum1 = %d\n", randNum1);
```

문제 풀려다가 실패하고 질문남겨요 ㅠ 맨아래까지 에요!

```
35              Bit1 |= (BIN1 << randNum1);
36
37          }
38          System.out.println(Bit1);
39
40          Thread.sleep(500);
41
42      }
43  }
44
45  // 일단 하나만 범위 정해서 풀어보려고 했는데 5~10 범위는 정해서 했습니다
46  // Bit1변수에 0 으로 초기화 해놓고 ((Bit1 &  (BIN1 << randNum1)) != 0) 이렇게 하면 되는건지 몰겠네요 ,..
47  // AND 랑 OR 쉬프트연산은 어느정도 이해하겠는데
48  //  결과값이 ranNum1 것들 5개 중복없는 애들만 나오고 마지막 Bite1 은 왜안나오는거죠?
49  // 1개라도 구하고싶어서 이것저것 찾아보고 복습도 하는겸 풀었는데 지금 런 했는데 중지되지않고 무한룹도 돌고있어요
50  //
```

무한루프 중이에요 .. 뭘잘못적은건지 ..