[디지털 컨버전스]
스마트 콘텐츠와 웹 융합 응용 SW
개발자 양성과정

강사 : 이상훈
훈련생 : 김원석
5회차 수업
2021/05/14

1. 비트 연산자 AND (&)

- (다 이해했습니다.)

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

        // & 이 비트연산자 AND
        // 관계 연산자에서는 && 형태로 존재하였음
        // 10 ===> 1010
        //  8 ===> 1000 AND
        // ----------------
        //  8 ===> 1000
        System.out.printf("%d AND %d = %d\n", num1, num2, num1 & num2);

        num2 = 138;

        // 138 ===> 10001010
        //  10 ===>     1010 AND
        // ---------------------
        //  10 ===> 00001010
        System.out.printf("%d AND %d = %d\n", num1, num2, num1 & num2);
    }
}
```

출력값)

```
10 AND 8 = 8
10 AND 138 = 10
```

-And(&)연산자는 1 , 0 (참과 거짓)으로 나뉘었을경우
1과 1이 만나야지 참이되고 1과 0이 만나면 거짓이 되어
결과값이 위와 같이 도래된다.

-그래서 선생님이 말씀하신것처럼 AND 연산은 교집합에 개념이다.

2. 비트 연산자 OR (|)

- (다 이해했습니다.)

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

        // | 이 비트연산자 OR
        // 관계 연산자에서는 || 형태로 존재하였음
        // 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
    }
}
```

출력값)

```
10 OR 5 = 15
10 OR 136 = 138
```

-OR(|) 연산자는 1 , 0 (참과 거짓)으로
나뉘었을경우 1과 0이 만나도 참이된다.
그래서
결과값이 위와 같이 도래된다.

-그래서 선생님이 말씀하신것처럼 OR 연산은
합집합에 개념이다.

4. 쉬프트 연산 (<<, >>) (이해 되었습니다.)

```java
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)
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 << num1);
// 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 2^10 = 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);

// 왼쪽 쉬프트의 경우 단순히 2^n을 곱하면 되지만
// 오른쪽 쉬프트의 경우 단순히 2^n을 나누면 안된다.
```

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

// 종합적 결론:
// 쉬프트 연산은 2^n을 곱하거나 나눈다.
// 안타깝게도 쉬프트 연산은 정수형끼리밖에 안된다.
// 최근에 나온 휴대폰 전용 ARM 프로세서에서는 소수점에 대한 쉬프트 연산을 지원하기도 한다.
```

-강사님이 말씀 하신 부분 처럼 결론적으로
쉬프트 연산은 2^n을 곱하거나 나눈다.
여기서 오른쪽 쉬프트 연산같은 경우
나누고 난뒤 소수점을 버린다.(제외 시킴)

5. 인터럽트(이벤트) Interrupt (이해 되었습니다.)

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

            Thread.sleep( millis: 500);
        }
    }
}
```

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

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

// 조금 더 구체적인 예제로 몰컴을 예로 들어보자!
// 전제조건: 부모님이 안방에서 주무시고 계신다.
//           나는 자는척하다가 일어나서 컴퓨터를 키고 게임을 시작한다.
// 1. 몰컴을 하고 있었음. 펜타킬찍음
// 2. 갑자기 안방의 문이 열렸음. <<<--- 이벤트(인터럽트)
// ???? 어떻게 해야할까요 ???? ---> 빨리 모니터 끄고 여차하면 컴퓨터도 끄고 자는척 시전
// ----> 문을 열어보시고 걸리면 이제 인생온라인 로그아웃
// ----> 이게 아니라면 문을 열어보시고 잘못 들었나 ? 하고 화장실갔다가 다시 주무시러 가실것임
// ----> 부모님의 퇴거를 확인하면 다시 일어나서 컴퓨터를 키고 탈주닌자 복귀를 한다.
// 3. 이후부터 다시 여러분의 시간

// 기본적으로 이벤트라는 것은 최우선적으로 처리해야 하는 작업으로
// 어떤 작업보다도 우선순위가 높은 녀석들입니다.
// 마찬가지로 여기서 Thread.sleep()하는 작업도 일종의 이벤트(인터럽트)에 해당한다.
// 그러므로 이 작업이 시작되면 다른 모든 작업들을 제쳐두고 이것을 최우선적으로 처리하게 된다.
// 물론 조금 더 정확한 것은 CPU의 동작과 Thread의 동작 과정에 대해 설명할 때 자세히 기술하도록 하겠다.

// 결국 Thread.sleep(500)이 가장 중요한 작업이므로
// 이 작업을 완료하기 전까지는 어떠한 작업도 수행하지 않는다는 뜻이다.
// 그래서 0.5초동안은 무조건적인 대기를 하게 된다.
```

Interrupted 하드웨어 개발자들이 주로 사용하는 단어.
2.보통 '자바'나 'GUI' '어플리케이션'의 개발자들은 이벤트라고 표현
3. 이벤트의 문법의 경우 어떤 작업보다 최우선적으로 처리해야 하는 작업이라는 뜻으로 이 작업을 완료하기 전까지는 어떠한 작업도 수행하지 않음.

22. 배열 없이 중복 회피하기 예제 복습 **(**이해되지 않는 부분이 있습니다**.)**

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

        //...
        int testBit = 0;
        int randNum;

        //...

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

1.BIN은 1로 고정값
2.testBit는 0
3.여기서 for문을 사용하여 i는 0 ~ 9의 숫자가 나오게 만듬
4.randNum = 0 ~ 9의 랜덤값

```
        while ((testBit & (BIN << randNum)) != 0) {
            System.out.println("중복이 이렇게나 많이 발생합니다: " + randNum);
            randNum = (int)(Math.random() * 10);
        }

        System.out.printf("randNum = %d\n", randNum);

        //...
        testBit |= (BIN << randNum);
    }

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

    Thread.sleep( millis: 500);
  }
}
```

while은 소괄호 안은 풀이하자면
(0 & (1 x 2^n)) != 0)인데
이 부분에서 0이 아니면 중괄호로 들어가서
또 숫자를 빼내는것은 알겠는데, 그 원리가 좀
헷갈린다. ← 이부분 다시 설명 부탁 드립니다.

그리고
testBit != (BIN << randNum);
이부분도 != 같지 않으냐란 표식을 사용하여
작성했는데 이부분도 설명 부탁 드립니다.