**Web개발\_Day03\_노트정리**

**제어문, Control Statement**

ㆍ제어문 : 실행의 흐름 위에 아래로 순차적으로 진행된다

* 때로는 문장을 건너뛰고, 때로는 반복해서 수행할 필요가 있다.

ㆍ조건문

* 조건에 따라 다른 실행 문장을 수행한다.
* If, if~else, if~else if~else, switch

ㆍ반복문

* 특정 문장들을 반복해서 수행
* For, while, do~while

**조건문**

ㆍ조건에 따라 다른 실행 문장을 수행한다

* 가장 기본적인 조건문법으로써, 가장 많이 사용된다.

**조건문1, if**

ㆍif (조건식) { 조건이 true일 때 수행할 문장 }

* 수행할 문장이 한 문장이라면 { } 표시 생략 가능 (추천되진 않는다.)
* 수행할 문장이 없다면 { ; } 를 적는다 (자주 있는 상황은 아니다.)
* 조건식의 결과는 true / false 의 결과값만 취급한다.

ㆍif문의 조건식

* 조건식을 두개 이상 사용하려면, “논리연산자”를 사용하여 연결 사용 가능하다.
* 조건문A && 조건문B : A와 B의 결과가 둘다 true일 때, true 리턴
* 조건문A || 조건문B : A 혹은 B 둘 중 하나라도 true일 때, true 리턴
* String변수.equals(“문자열데이터”) : 변수의 데이터가 “문자열데이터”와 동일하면, true 리턴

ㆍ블럭 { } : 중괄호 {} 를 이용해서 여러 ‘실행할 문장(들)’을 하나의 단위로 묶어서 사용한다.

* 블록 끝엔 ;를 붙이지 않는다.
* 보통 여러줄을 넣는데, 한문장을 넣거나 안넣을 수도 있다.
* 실행문이 한줄일 경우 생략 가능

**조건문1-2, if~else**

ㆍif 조건문의 연장된 기능

ㆍif (조건식) { 조건식이 true일 때 실행할 문장; }

else { 조건식이 false일 때 실행할 문장; }

ㆍ if문 두개와 if~else문의 차이점 : if문 두개는 무조건 if문의 조건식을 두번 경유하게 되는 반면에, if~else문은 if 조건식을 하나만 경유하여 무슨 실행구문을 갈지 결정한다.

* 성능 향상

**조건문1-3, if~else if~else**

ㆍif 조건식에 분기를 추가하여 경우의 수를 늘릴 수 있다.

ㆍif (조건식1) { 조건식 1이 true 일 때, 수행구문; }

else if (조건식2) { 조건식 2이 true일 때 수행구문; }

else if (조건식3)\_ { 조건식 3이 true일 때 수행구문; }

…

else { 모든 조건식이 false일 때 수행구문; }

ㆍ첫번째 조건식부터 순서대로 평가하여, 결과가 true인 조건식을 만나면 해당 블록만 수행하고 벗어난다.

**조건문1-4, if 중첩문**

ㆍif문 실행 구문에 다시 if문을 작성하여 이중 if 중첩문을 사용할 수 있다.

ㆍif (조건식) {

If (조건식) { 실행구문 }

… 실행구문;

}

* 논리연산자를 활용하여 동일한 기능을 구현할 수도 있으므로, 상황에 따라 사용해야 한다.

**조건문2, switch**

ㆍif문은 기본형 자체가 true / false 인 두가지 경우밖에 없기 때문에, 경우의 수가 많아질수록 else if를 추가해야 하므로 조건식이 많아져서 복잡해진다.

ㆍif와 달리 switch문은 단 하나의 조건식으로 많은 경우를 처리한다. **=가독성이 좋아진다.**

* 처리할 조건이 많다면 if <<< switch

ㆍ구동 순서

1. 조건식 계산한다
2. 조건에 일치하는 case로 ‘점프’한다
3. 그 case 문장들을 수행한다.
4. Break문이나 switch구문의 끝에서 전체를 빠져나간다.

ㆍswitch (변수) {

Case 1 : 실행문

Case 2 : 실행문

Case 3 : 실행문

Case 4 : 실행문

Default : 모든 조건이 false일 경우 실행문

}

* 자세한 사용법은 Day03.java 파일 참고