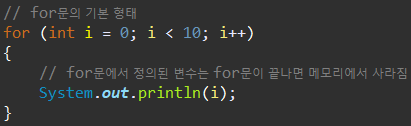
**[ 반복문 ]**

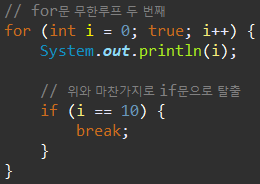
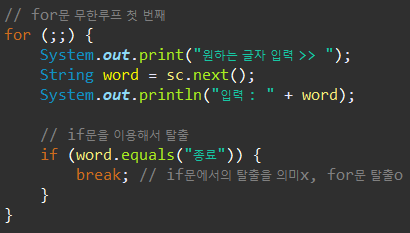
**- for**

• for문의 기본 형태



for문의 첫 번째 요소는 for문에서 사용할 변수를 초기화하거나 선언하고 정의한다. (여기서 선언된 변수는 for문 밖에서 사용할 수 없다.) 두 번째 요소는 for문이 실행될 조건을 정의한다. 세 번째 요소에는 변수의 증감식을 작성한다.

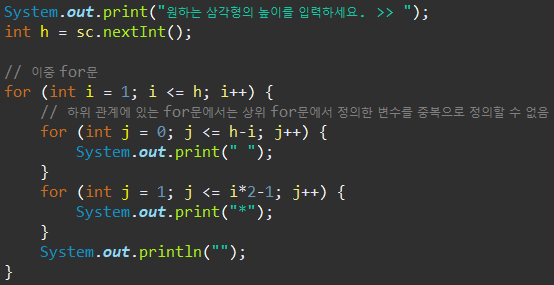
• for문 무한루프



for문의 요소를 적지 않거나, 조건식을 true라고 적으면 무한루프로 사용할 수 있다.

이 때는 if문을 사용해서 무한루프를 빠져나오는 조건을 설정해주면 된다.

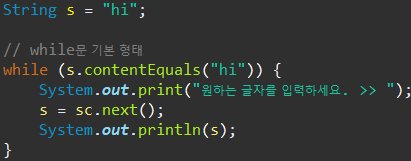
• 이중 for문



for문 안에 다중으로 for문을 쓸 수 있다. 하위 for문의 경우 상위 for문에서 정의한 변수를 사용할 수 있기 때문에, 같은 이름의 변수를 정의할 수는 없다. 반대로 상위 for문에서는 하위 for문에서 정의된 변수를 사용할 수 없다.

**- while**

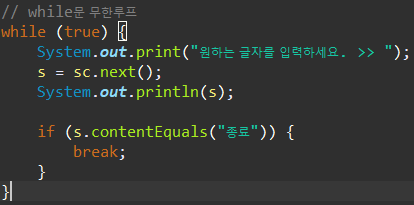
• while문의 기본 형태



while문은 조건식을 넣어주고, 조건에 맞을 때 실행된다. (조건에 맞지 않게 되는 순간 while문을 빠져나오게 된다.)

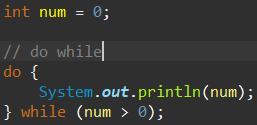
for문과 다른 점은 while문은 밖에서 변수 정의를 해주고, 그 변수를 사용해야 한다는 점이다.

• while문 무한루프



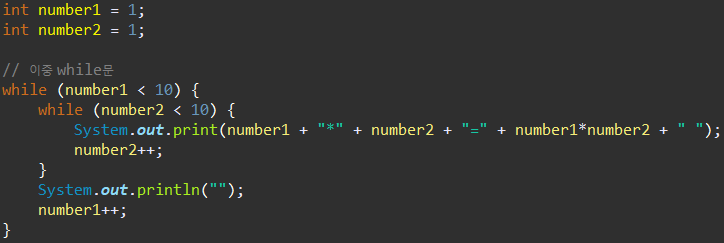
true로 조건을 넣어주면 무한루프로써 작동하게 된다. for문과 마찬가지로 if문을 통해서 루프를 빠져나올 수 있다.

• do-while문



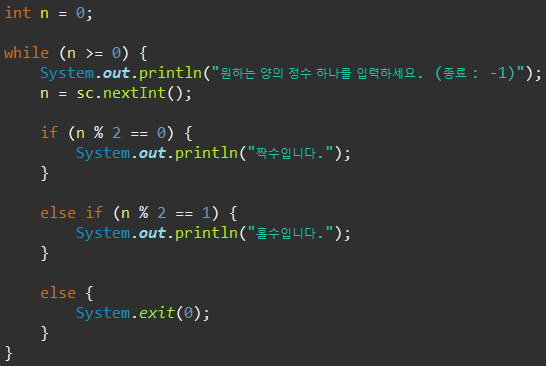
while문과 동일하지만, 차이점은 코드를 일단 한 번 실행하고나서 그 후에 조건이 맞는지를 판별한다는 것이다.

• 이중 while문



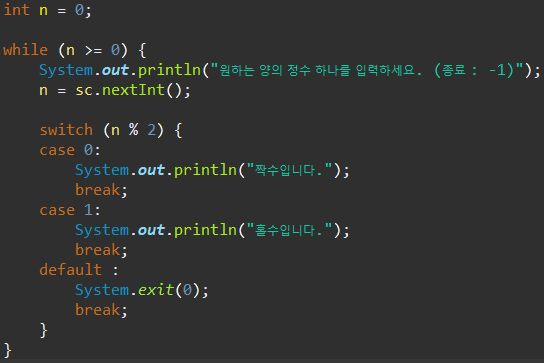
**[ 조건문 ]**

**- if**



if문은 조건식이 참인 경우에 코드가 실행된다. else의 경우에는 if, else if의 조건이 맞지 않는 모든 경우의 수를 얘기하기 때문에 따로 조건을 적지 않는다.

**- switch**



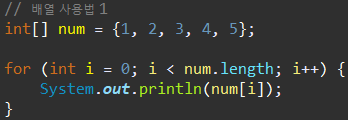
하나의 값을 가지고 여러 조건을 설정하고자 할 때 if문 보다 간편하게 표현할 수 있다. default는 if문의 else와 같은 역할을 한다. 모든 case에 맞지 않을 경우 default에 해당한다.

switch 내에서는 break가 매우 중요한 역할을 한다. case안에 break가 있어야만 switch문을 벗어날 수 있고, break가 없을 경우 계속해서 다음 case를 실행하게 된다.

switch문에 사용되는 수식은 항상 정수형 자료여야 하고, case문에 사용되는 값은 오로지 한 개의 값만 사용할 수 있다.

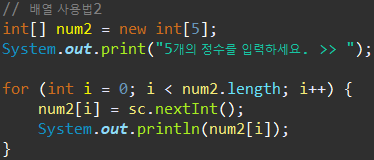
**[ 배열 ]**

**- 배열을 정의하는 방법 첫 번째**



직접 배열에 값을 넣어주는 방법이다. 배열은 for문을 이용해 값을 읽어오거나, 넣어주면 편하다.

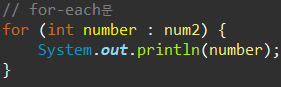
**- 배열을 정의하는 방법 두 번째**



new 연산자를 사용해서 배열의 타입과 크기만 정해주고, 추후에 필요한 값을 넣어주는 방식이다.

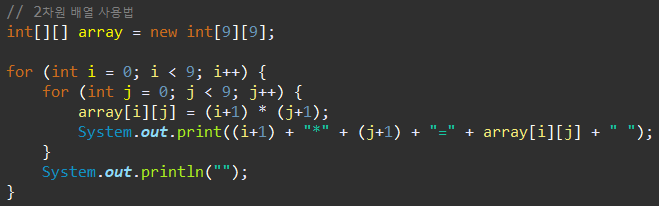
**- 배열과 for-each 반복문**

(for each라는 키워드가 따로 있는 것은 아니고, 동일한 for를 사용하지만 조건식 부분이 다르다.)



직관적으로 조건식을 나타내고, 배열을 순차적으로 반복하는 반복문이다. 간단하다는 장점이 있지만, 반복 회수나 조건을 따로 주지 못하고 꼭 한 단계씩 순차적으로 반복될 때만 사용할 수 있다는 제약이 있다.

**- 2차원 배열**



1차원 배열과 같은 방식으로 선언할 수 있고, for문을 사용할 때는 2중 for문을 이용하면 된다.