3장 과제 – 산술과 비트 연산 계산기

- **입력되는 중위 표현의 논리연산을**

  1) 후위 표기를 바꾸어 출력하고

  2) 계산 결과를 출력하는 프로그램을 작성하시오.

  - 아래의 연산자만 대상으로 함
    - 단항연산자 : - (마이너스), ~(bitwise not)
    - 이항 산술 연산자 : * / % + -
    - 이항 비트 연산자 : <<, >>, &, ^, |
    - 괄호연산자
    - 연산자 우선 순위는 Java 기준과 동일하게 적용
  - 피연산자들과 연산자들 사이에 항상 공백이 있는 것으로 가정
  - 피연산자는 정수 만 허용
  - 계산 결과가 Java의 결과와 동일해야 함

- **강의지원시스템에 4/25(일) 23:00까지 제출**
  - .java 파일들만 하나의 압축파일로 만들어 제출.
  - 파일이름은 본인 학번으로.

3장 과제 – 계산기

실행 예제

식을 입력하세요 (종료는 quit) : 1 + 2 * 4 * (5 – 3 + 1) + (4 | 2 ^ 7 & 3)
--- postfix notation은 : 1 2 4 * 5 3 – 1 + * + 4 2 7 3 & ^ | +
---- 결과는 30

식을 입력하세요 (종료는 quit) : 11 % 3 * 2 ^ (3 * (1 + 7) << 3)
--- postfix notation은 : 11 3 % 2 * 3 1 7 + * 3 << ^
---- 결과는 196

식을 입력하세요 (종료는 quit) : 13 >> 2 & 1 + - - 9
--- postfix notation은 : 13 2 >> 1 9 m m + &
---- 결과는 2

계산식을 입력하세요 (종료는 quit) : 3 * 5 << ^ ~3
--- postfix notatio은 : 3 5 * << 3 ~ ^
[오류] 이해할 수 없는 수식

계산식을 입력하세요 (종료는 quit) : 3*7<<~-2
--- postfix notation은 : 3 7 * 2 minus ~ <<
---- 결과는 42

계산식을 입력하세요 (종료는 quit) :quit
**** 종료합니다 ****

– 또는 ~와 같은 단항연산자는
후위연산식으로 변환한 뒤, 계산할 때는
stack에서 피연산자 하나만 pop하여
단항연산을 수행하는 것이 좋은 구조.
마이너스와 빼기가 혼동될 수 있으므로
구분할 수 있는 기호로 변경

"<< ^" 가 연산자 연속이라서
오류가 나는 것이 아닌
(ex. "31 * – – (((–2 * 3) + …"
와 같은 표현이 있을 수 있으므로),
후위표기 생성 후 계산과정에서
"<<" 의 피연산자가 부족해서
오류로 인식되는 것이
바람직한 처리 방식임

3장 과제 – 비교와 논리, 비트 연산 계산기

- **구현 아이디어**
  - 중위표현과 후위표현은 연결리스트로 구현된 Queue<Token>에 저장
    - class Token 은 isOperator가 true이면 operator에 연산자 문자열을, false이면 operand에 int형 값을 저장
    - 추후 Tree 계산기가 과제로 나갈 예정이므로, 좋은 클래스 설계가 도움이 됨
  - 피연산자와 연산자간 공백이 있어도 없어도 되게 구현하면 추가 점수
  - 계산식 parsin에서 split, StringTokenizer 수준의 메소드 사용은 허용
    - 단, 정규식이나 match()등의 2학년이 이해할 수 없을 것으로 보이는 메소드 사용시 개별 발표 요구 또는 감점 예정
  - 단항연산자인 –, ~ 연산자도 연산자로 취급하여 구현 할 것
    - -1이 들어오면, “-1”의 정수로 저장하면 일부 감점
    - 마이너스와 빼기를 구분하기 위해 마이너스를 “minus”등의 기호로 변경 가능