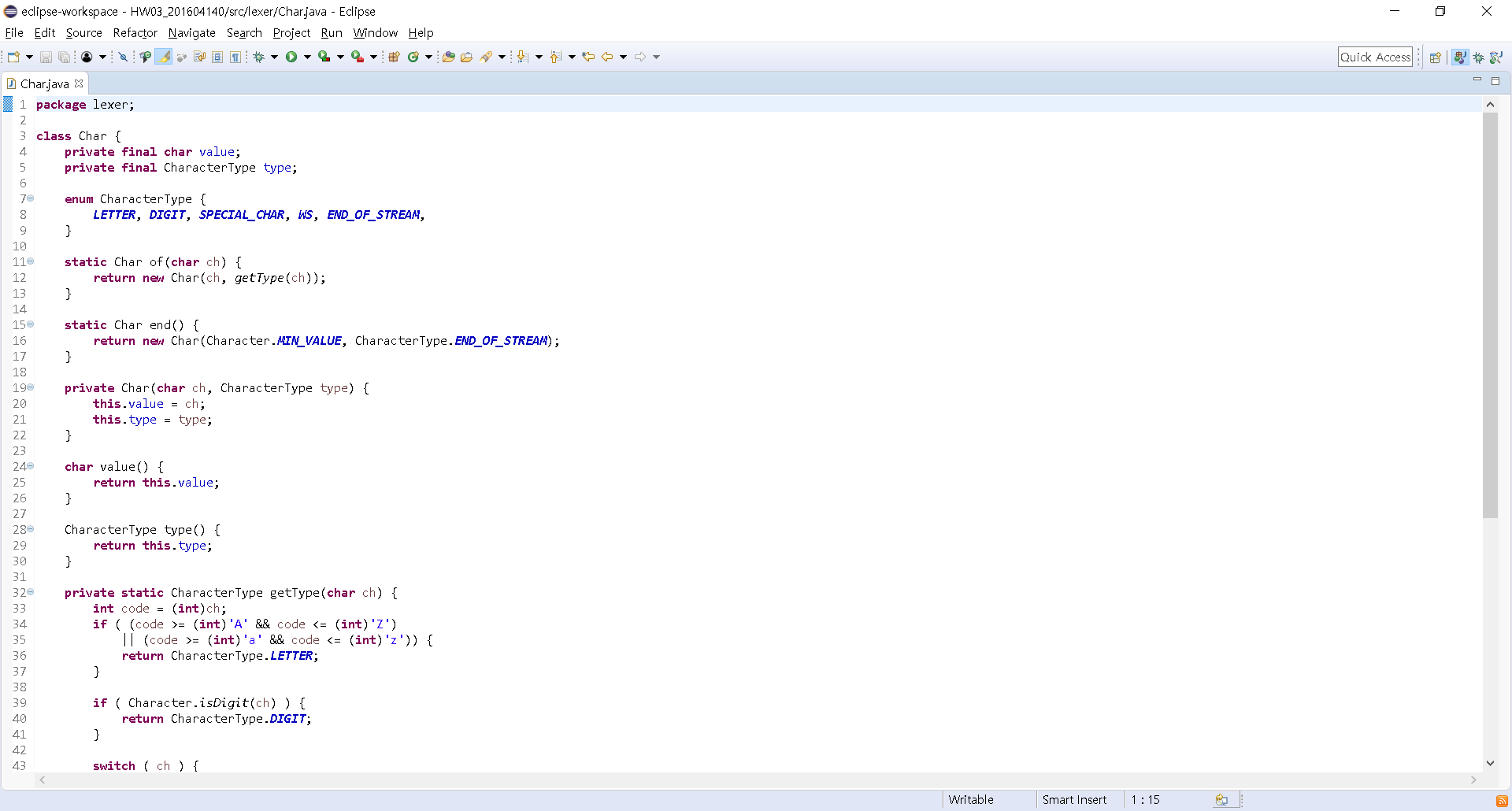
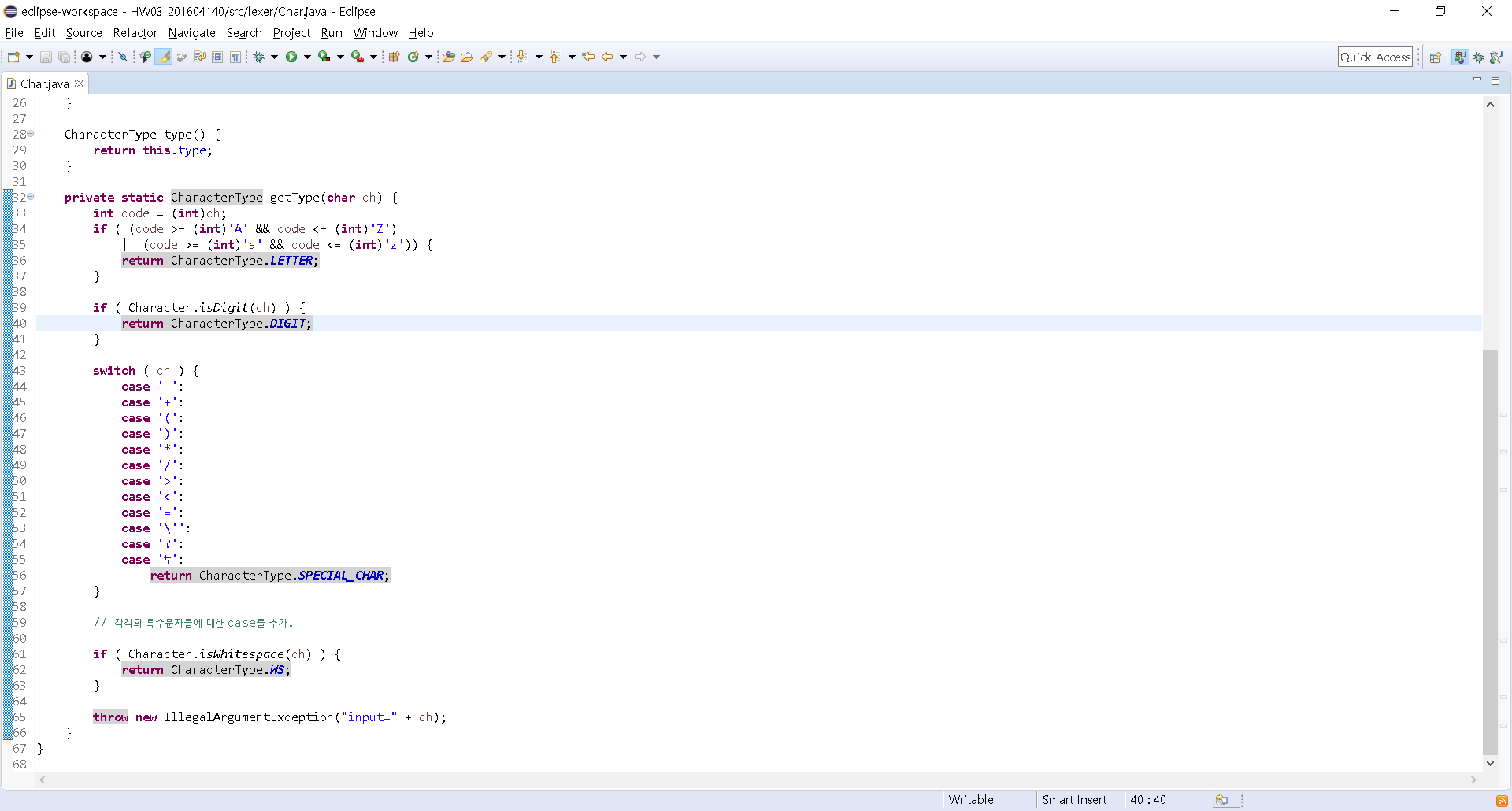
**프로그래밍언어개론 보고서**

**컴퓨터공학과 201604140 박경수**

**PL Assignment #3 : Cute14 Scanner**

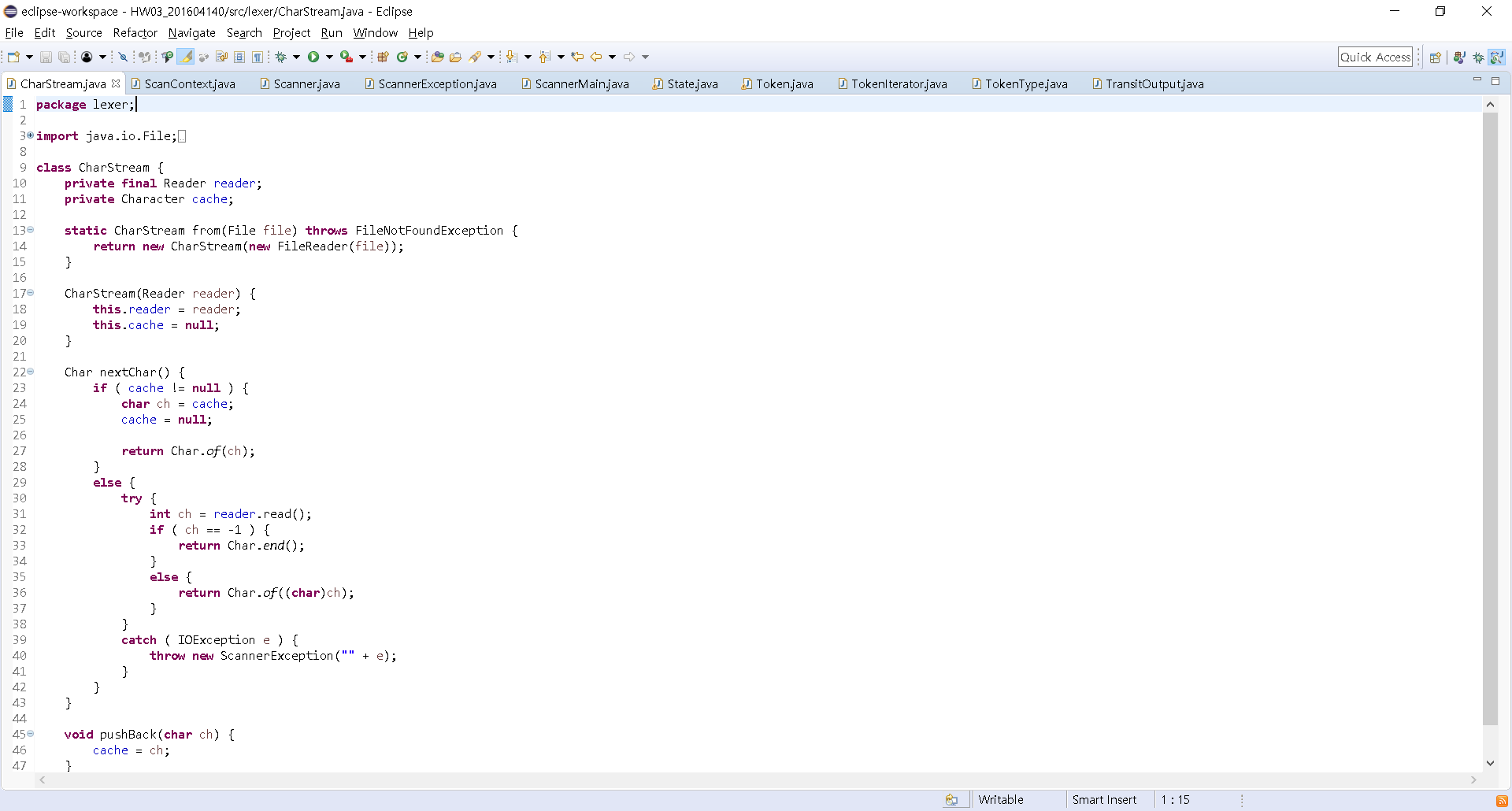
**Char.java**



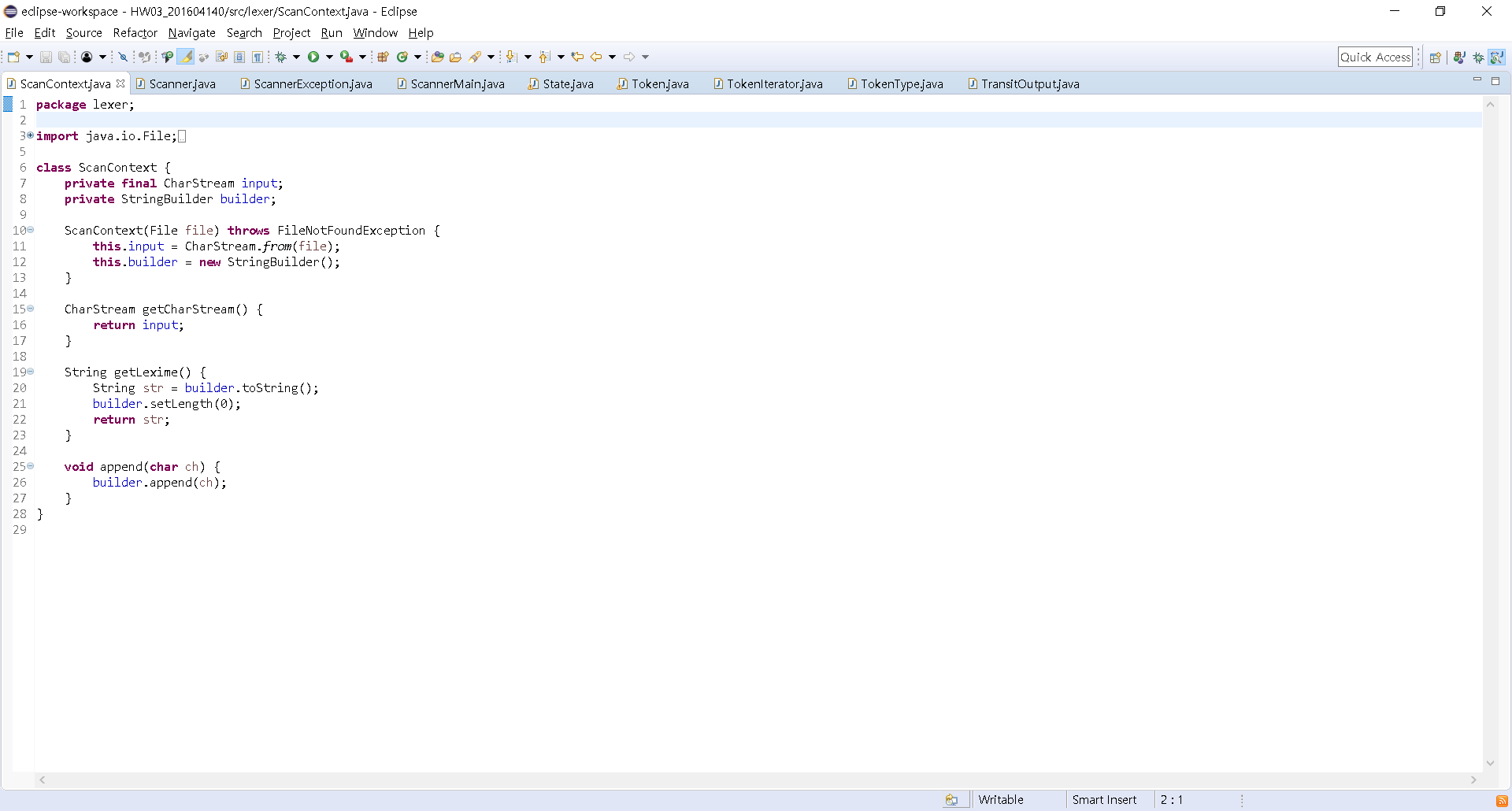


1. Char.java에서는 switch문에서 각각의 특수문자들에 대한 case를 추가시켜준다.

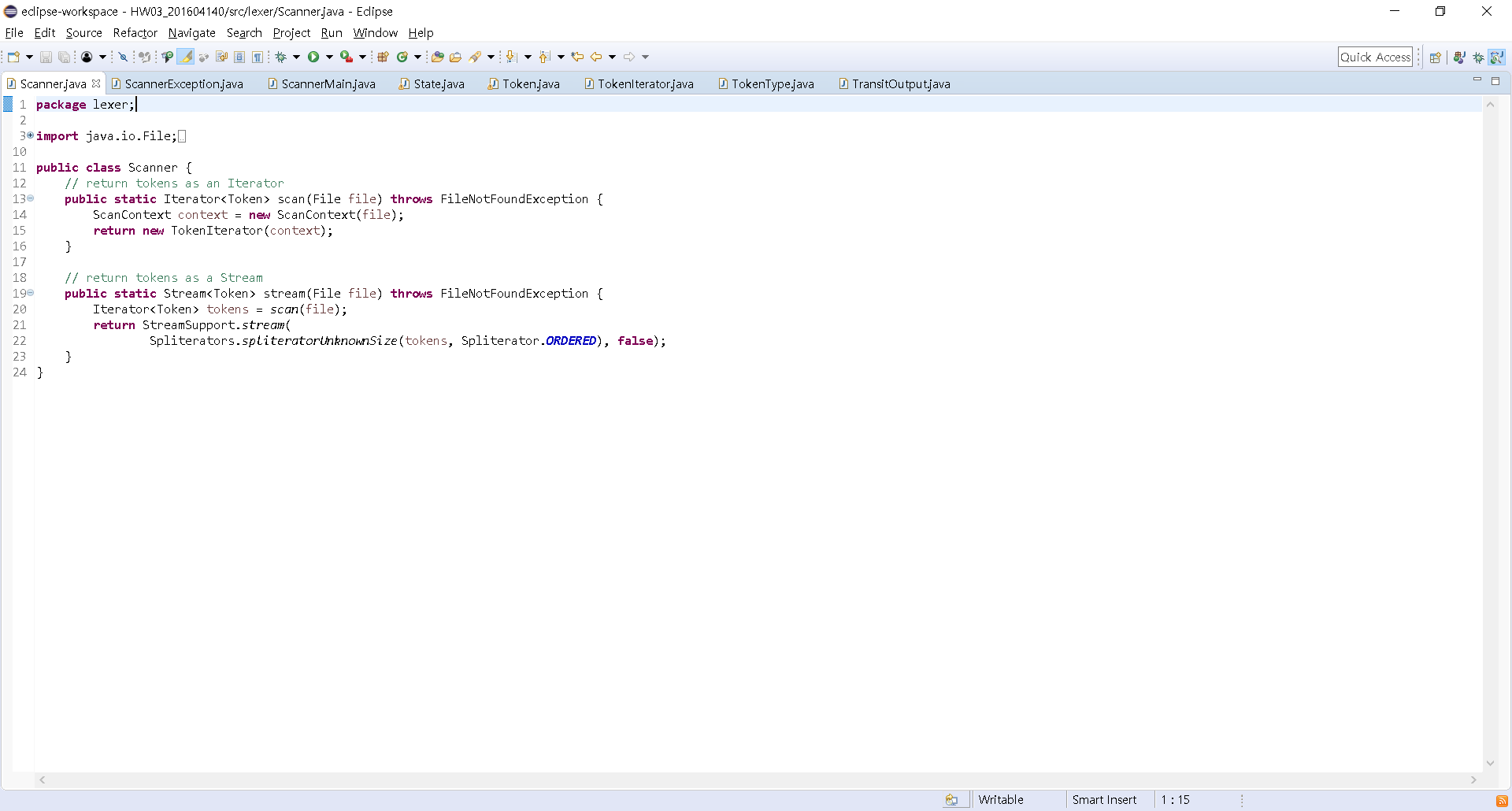
**CharStream.java**



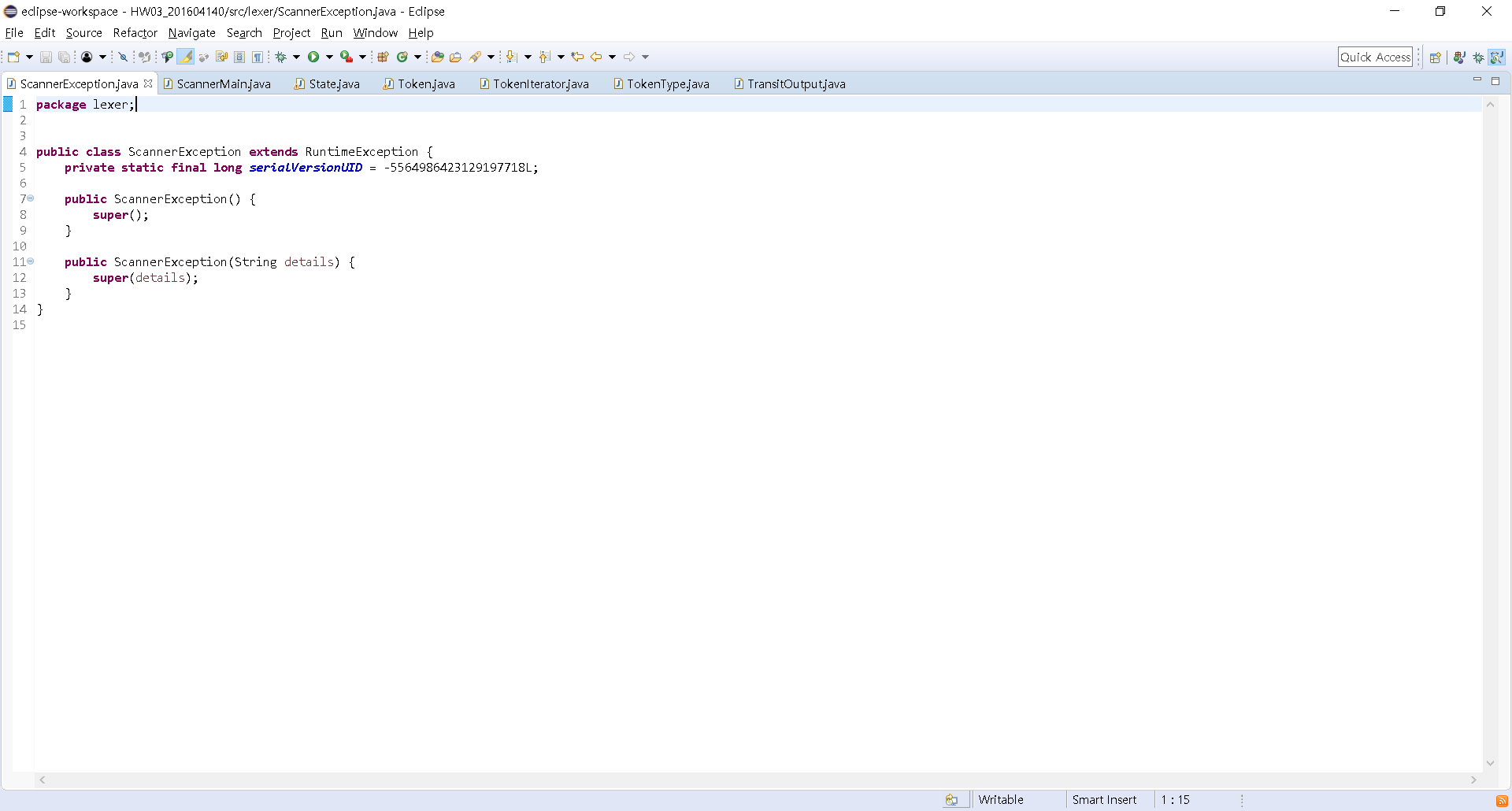
**ScanContext.java**



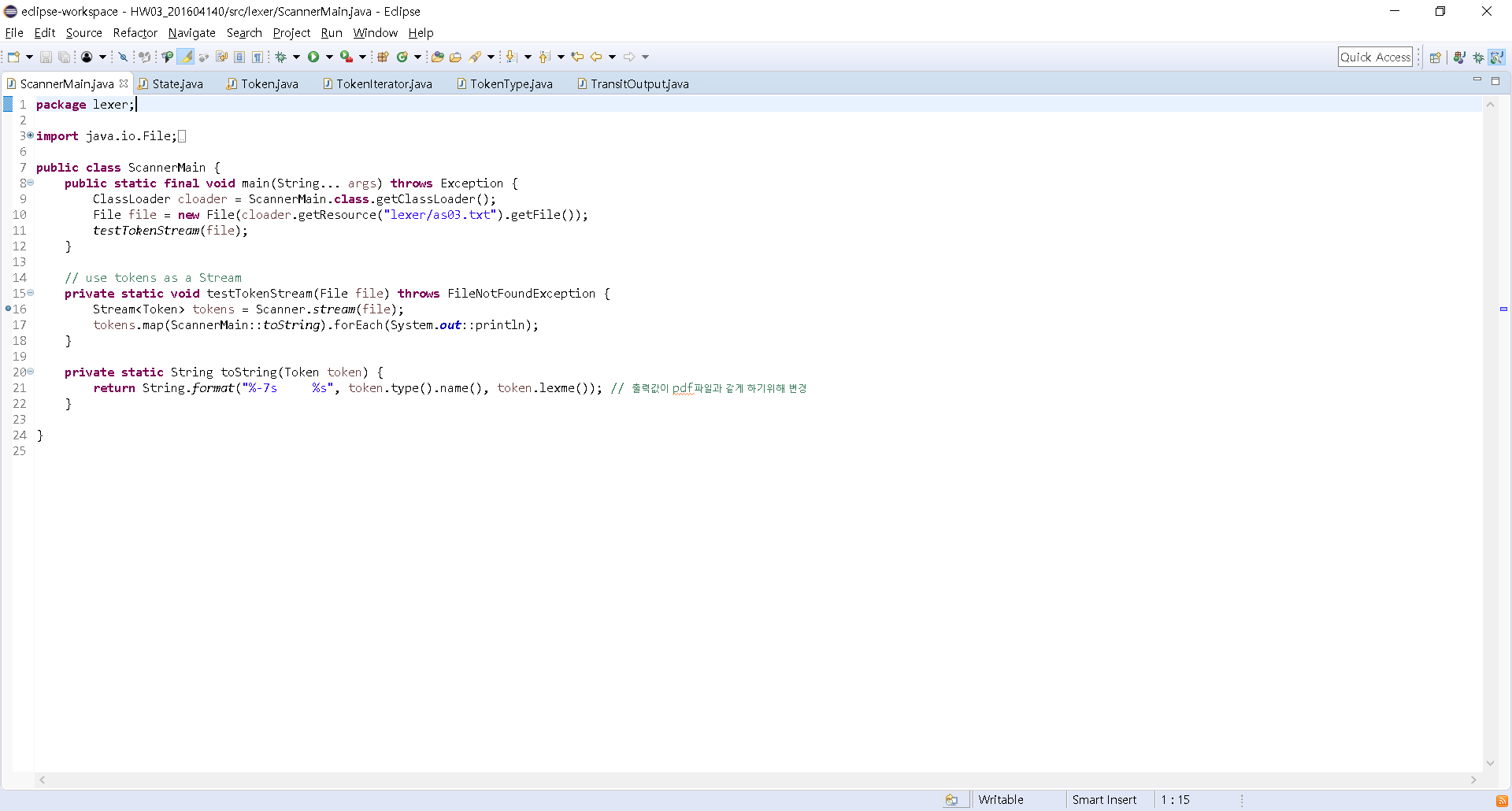
**Scanner.java**



**ScannerException.java**



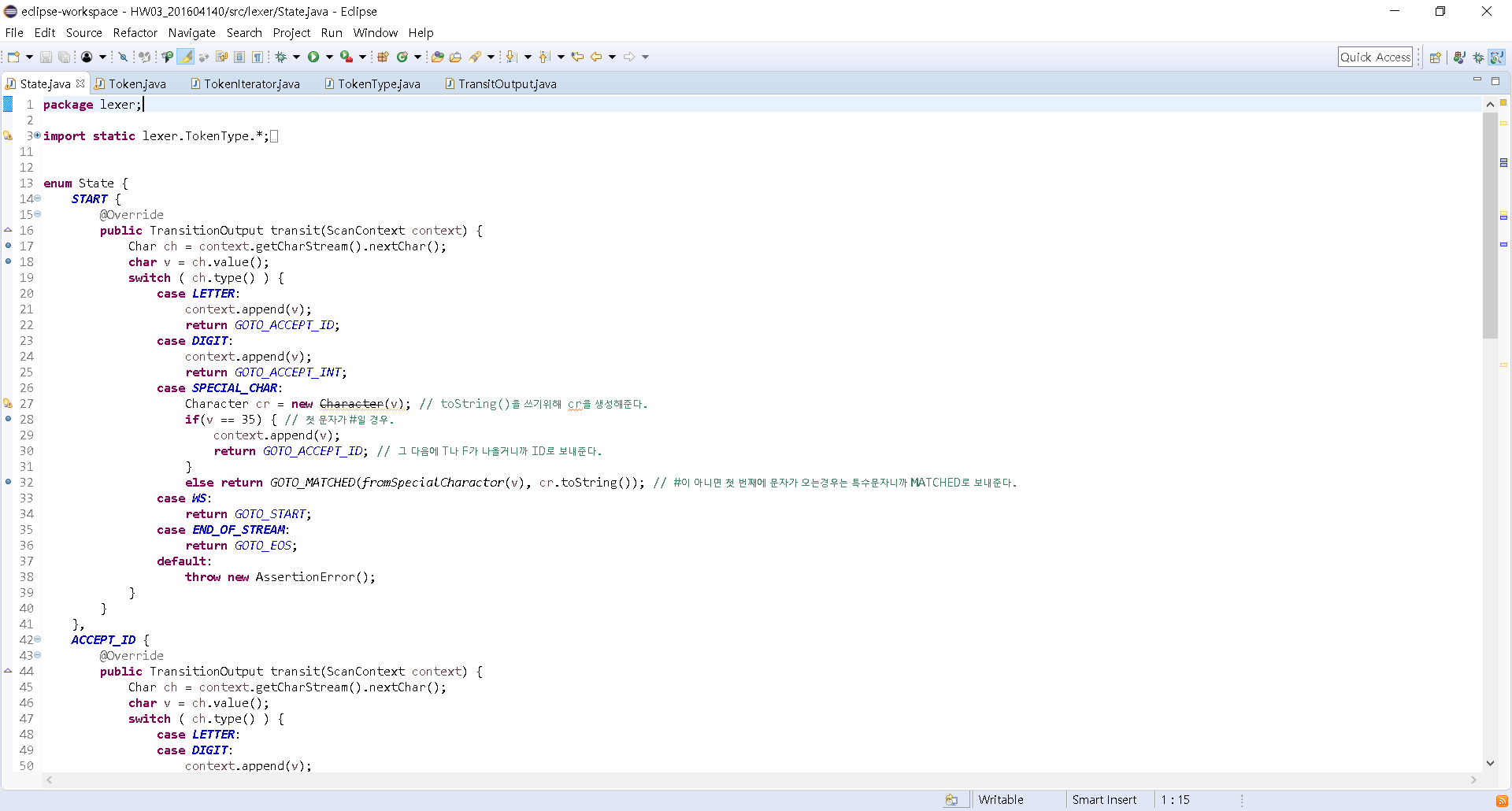
**ScannerMain.java**



1. 파일은 lexer 패키지에 있는 as03.txt를 불러온다.

2. 출력값이 pdf에서 나오는 것처럼 만들기위해 적절히 변경해준다.

**State.java - START**



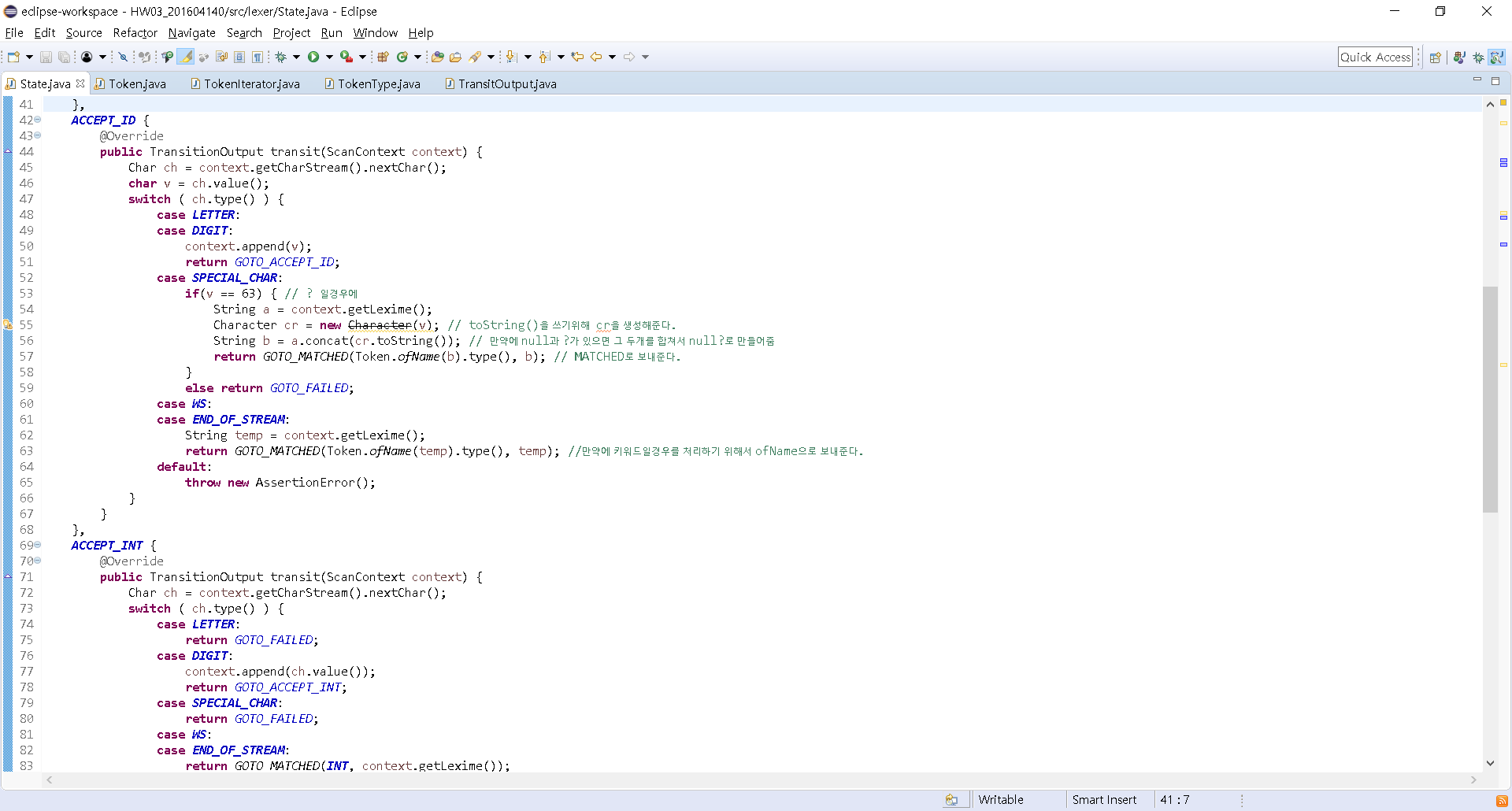
1. 토큰타입의 전체를 import 해준다.

2. 만약에 처음에(START 상태일때) SPECIAL\_CHAR가 나왔을 경우는 #T와 같은 경우 또는 특수문자가 나오는 경우이다.

3. #일 경우에는 그 다음에 T나 F가 나오기 때문에 ACCEPT\_ID로 보내준다.

4. 다른 특수문자가 오는 경우에는 formSpecialCharactor를 통해 TokenType을 리턴받아서 MATCHED상태로 보내준다.

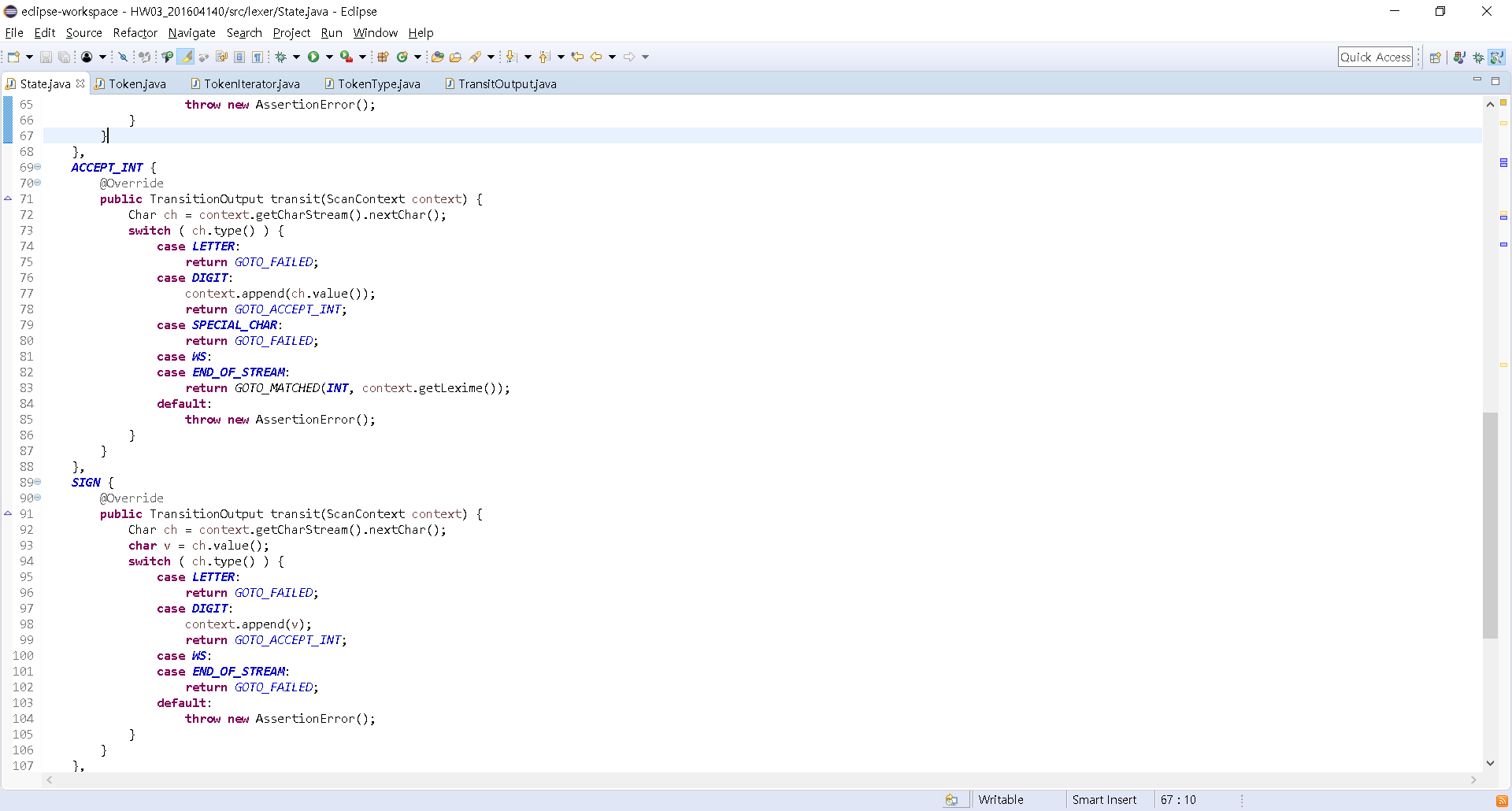
**State.java – ACCEPT\_ID**



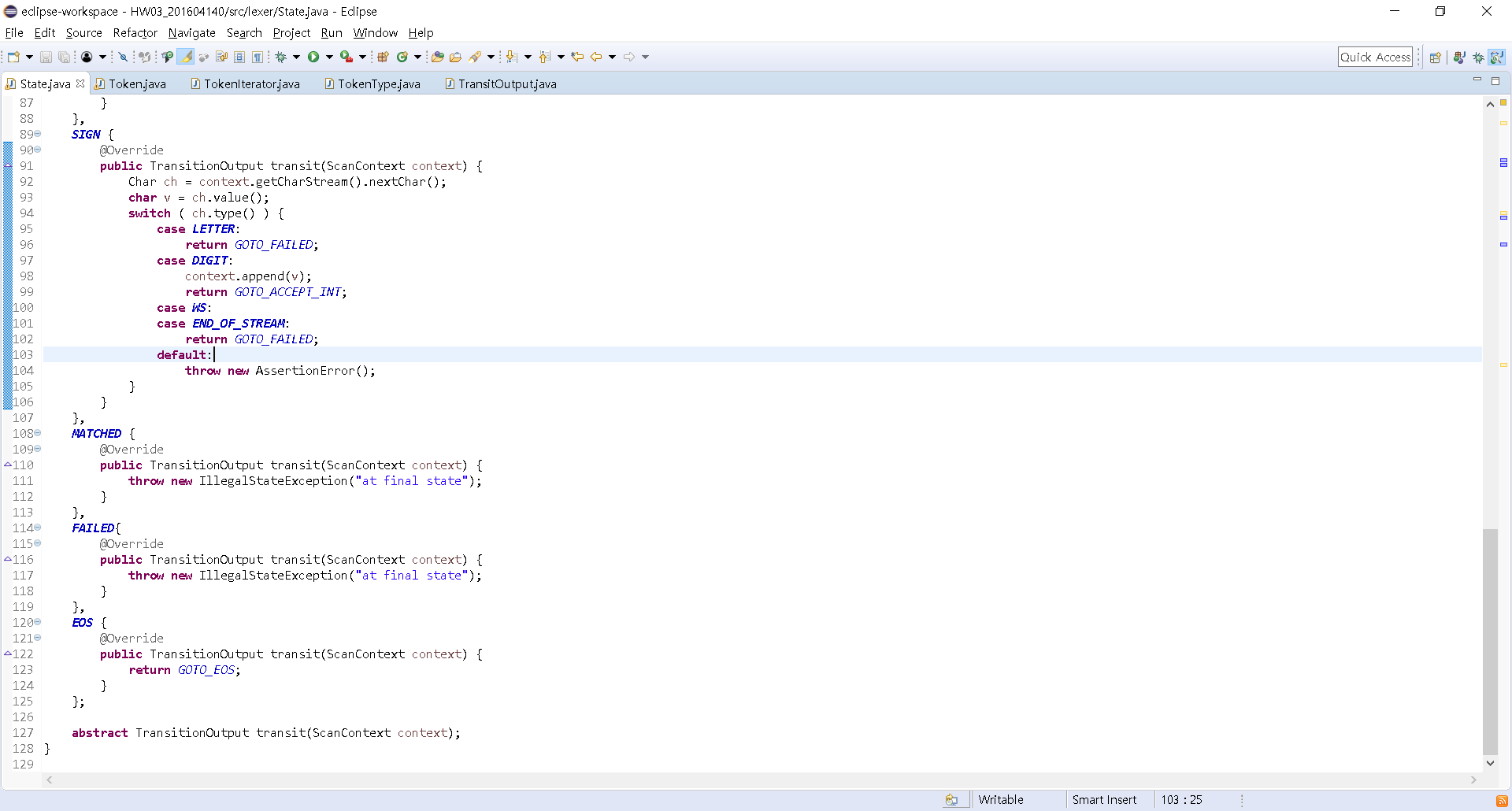
1. 만약에 다음 문자가 SPECIAL\_CHAR일 경우인데, 물음표가 오는 경우에 context.getLexime()을 통해 나오는 String과 (ex, null) nextChar()를 통해 나온 문자 (ex, ?)를 concat을 통해서 합쳐 준다음에 MATCHED로 보내준다. (그때, null?는 NULL\_Q의 키워드를 타입으로 받아야하기 때문에 ofName을 통해 Type을 받아온다.)

2. 만약에 람다 같은 키워드일 때도 알파벳으로 끝나기 때문에 END\_OF\_STREAM일 때 다시 ofName을 통해 Type을 받아와서 MATCHED로 보내준다.

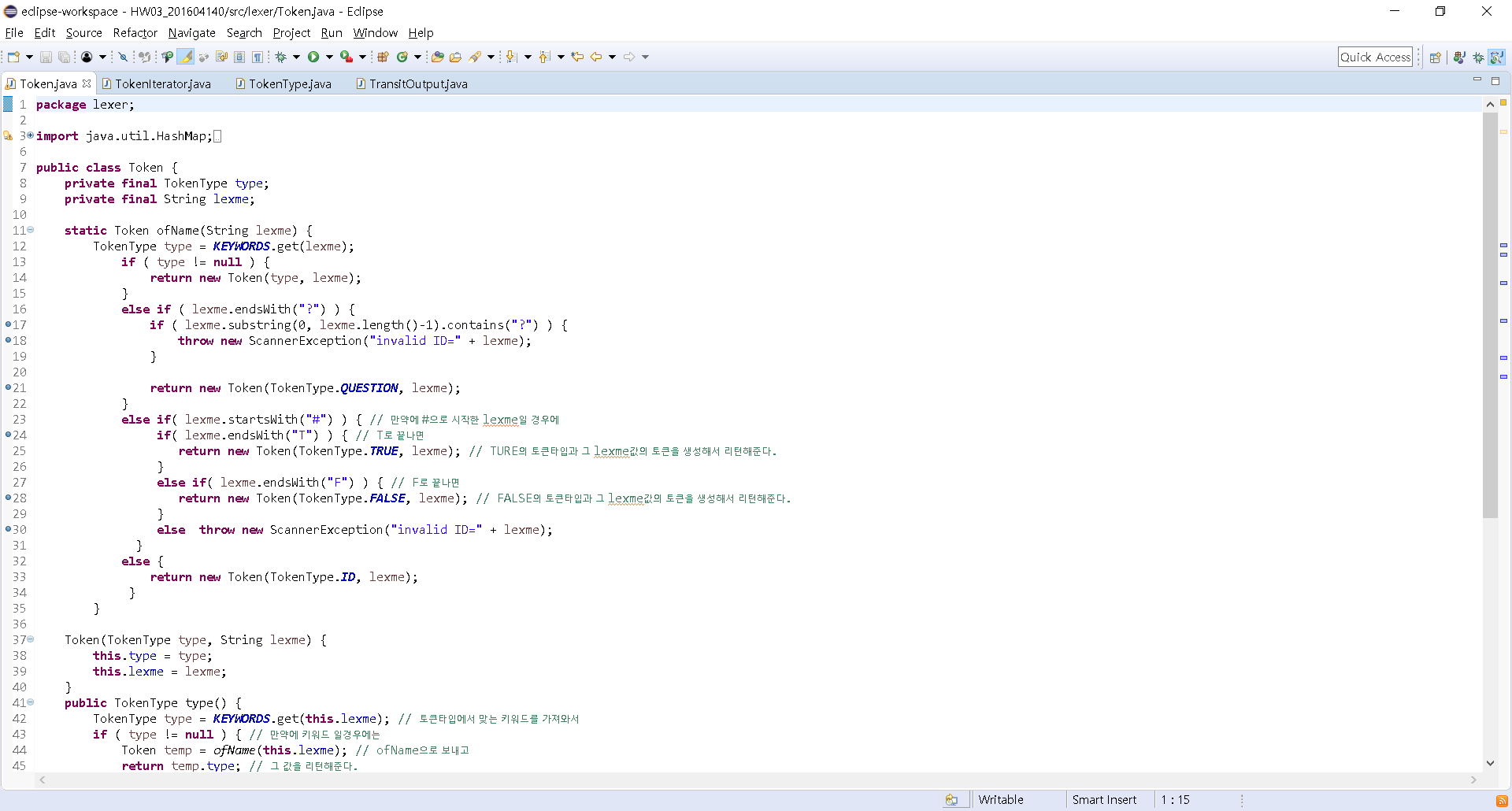
**State.java – ACCEPT\_INT, SIGN**



**State.java – MATCHED, FAILED, EOS**



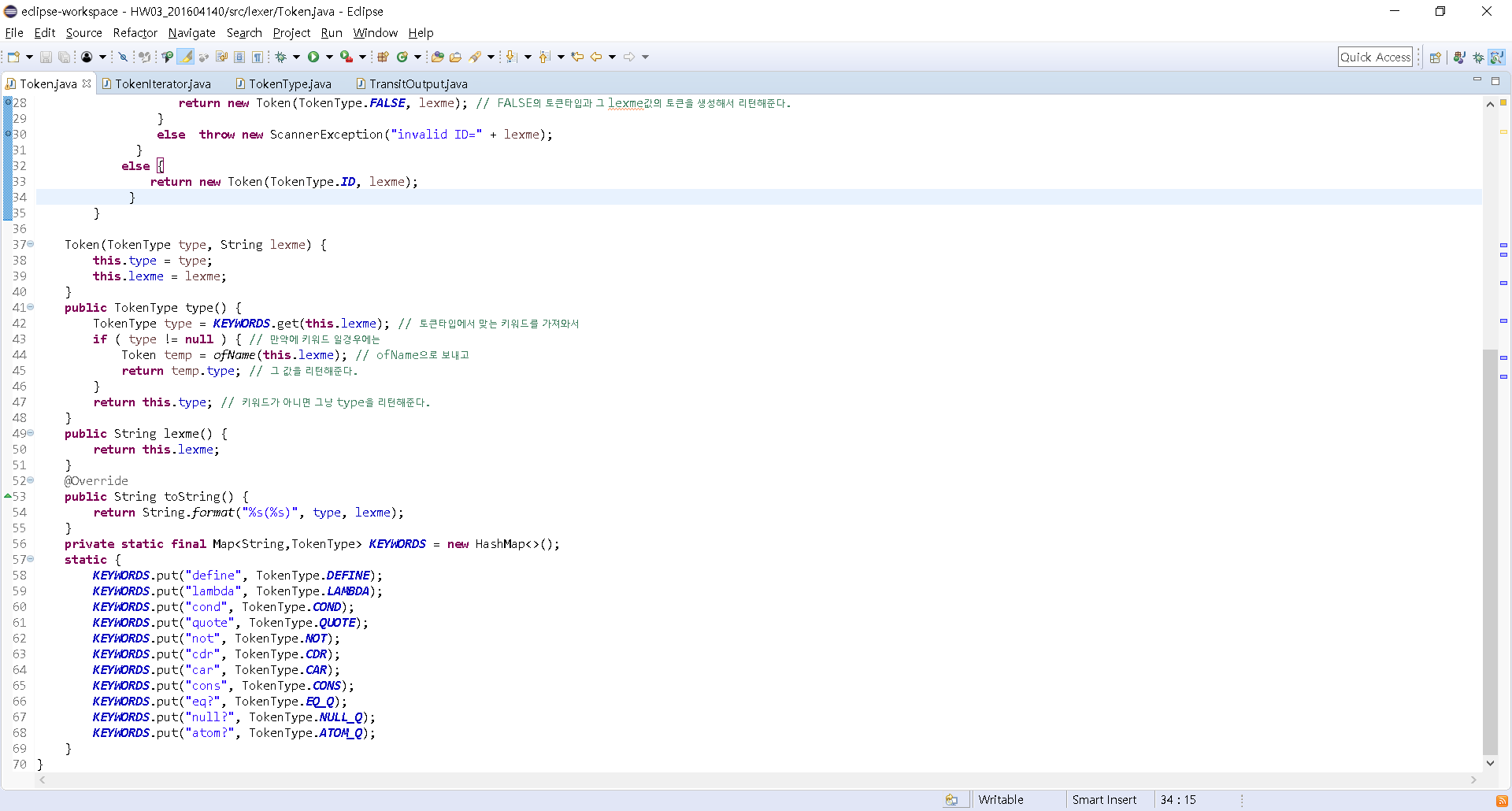
**Token.java - 1**



1. ofName에서 만약에 lexme의 시작이 #으로 시작할 때 T로 끝나면 TRUE의 토큰타입과 그 lexme값의 토큰을 생성해서 리턴해준다.

2. F로 끝나면 FALSE의 토큰타입과 그 lexme값의 토큰을 생성해서 리턴해준다.

**Token.java - 2**

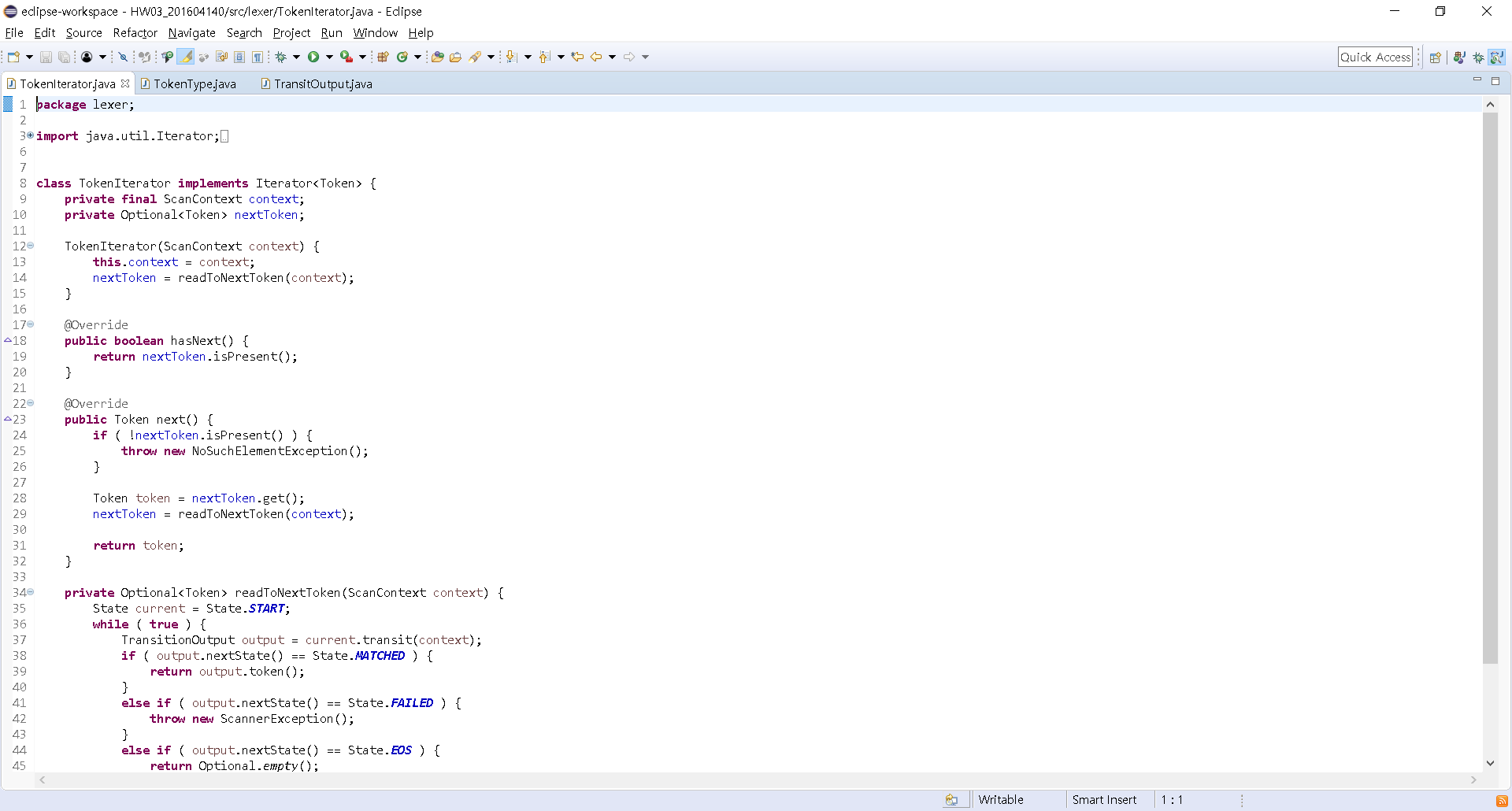


1. type()함수에서 토큰타입을 리턴해주는데, 해당 lexme에 맞는 키워드를 가져와서 그것이 키워드일 경우에는 ofName으로 보내고 그 값을 리턴해준다.

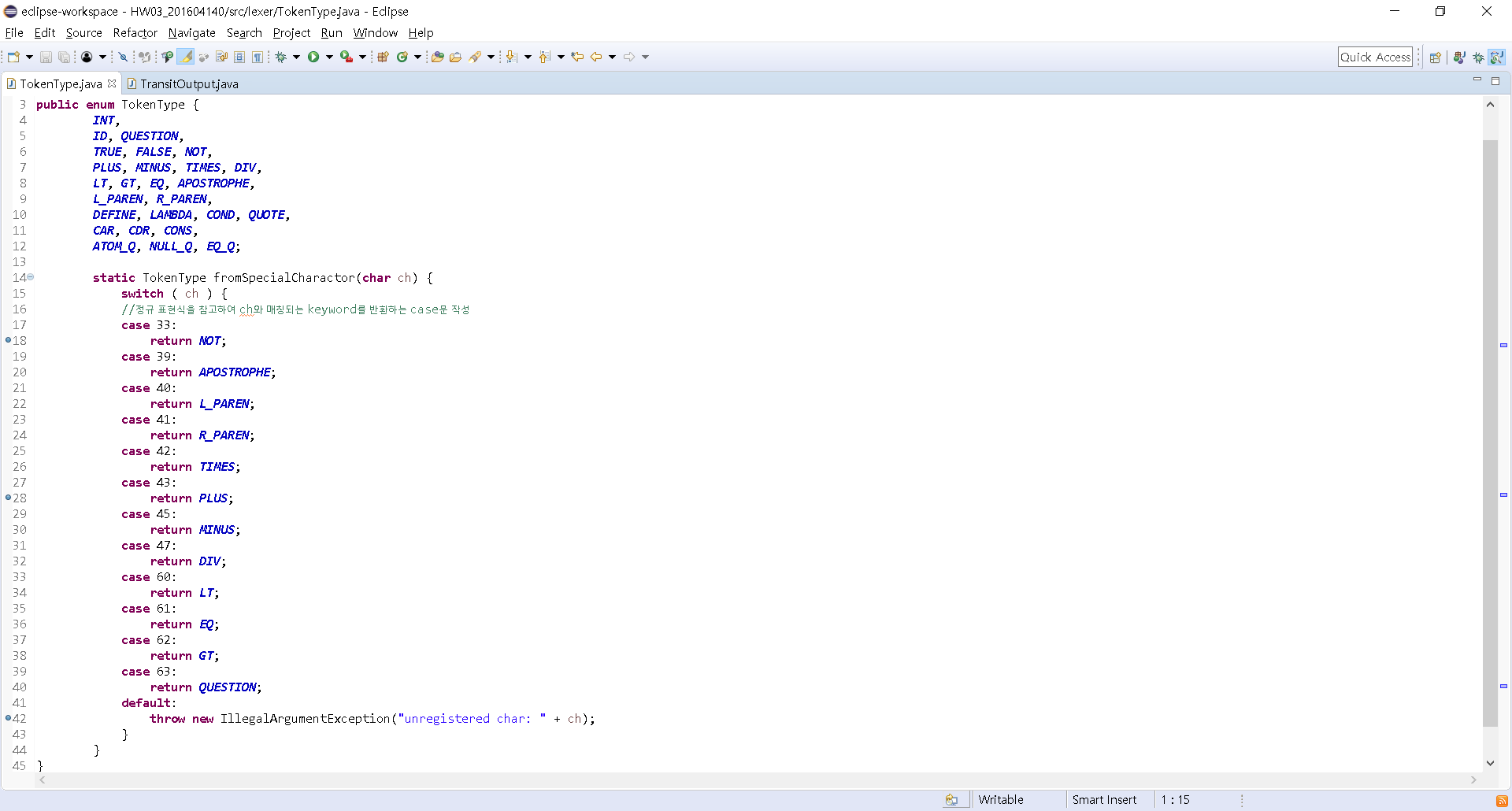
2. 키워드가 아닐 경우에는 그냥 원래 type을 리턴해준다.

(만약에 다 ofName으로 보내서 리턴받으면 키워드가 아니기 때문에 마지막 else문에서 토큰타입을 무조건 ID로 받아옴)

**Tokenlterator.java**

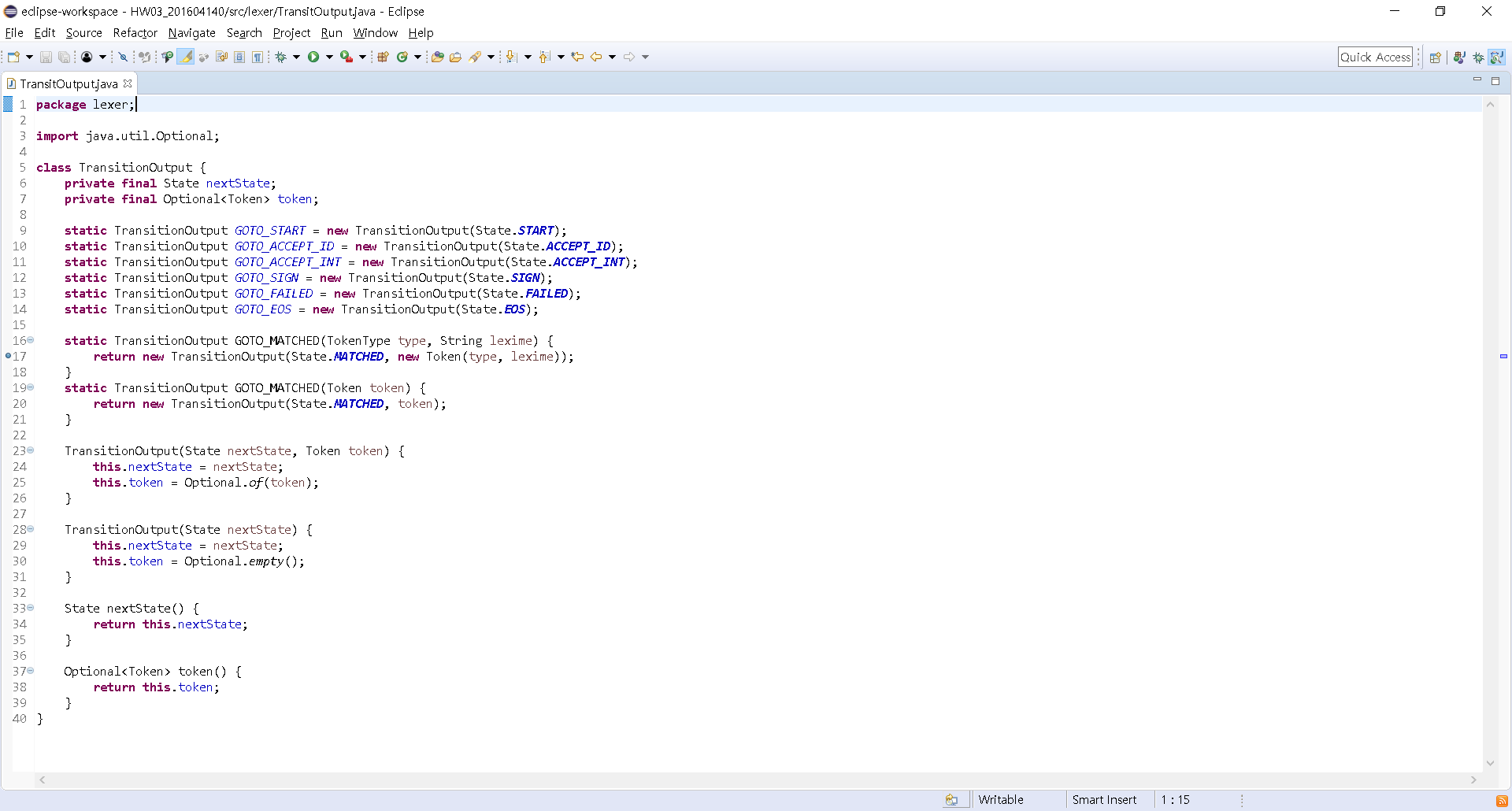


**TokenType.java**



1. 각각의 특수문자에 대한 토큰타입을 case문을 추가해줘서 리턴해준다.

**TransitOutput.java**



**실행화면**

