```
public class Parser {

    public static void main(String[] args) {
        ProgramText programText = new ProgramText();
        Scanner scanner = new Scanner(programText);
        Token token = new Token(programText);
        Parser parser = new Parser(scanner, programText, token);
        parser.parse();
    }

    private Scanner scanner;
    public ProgramText programText;
    private Token curToken,nextToken;
    private int rightCurly = 0, leftCurly = 0;

    Parser(Scanner scanner, ProgramText programText, Token token) {

        this.scanner = scanner;
        this.programText = programText;
        this.curToken = token;
    }

    void parse() {
        curToken = scanner.nextToken();
        while (!(curToken instanceof EOFToken)) {
            if (!(curToken instanceof EOFToken)) {
                if (curToken != null) {
                    //System.out.printf("Type: %s, text: %s\n",
curToken.getTokenType(), curToken.getText());
                    S();
                }
            }
            curToken = scanner.nextToken();
        }
        if (!curlyController()) {
            System.out.println("1Something is wrong.. " +
curToken.getTokenType());
            System.exit(0);
        }
    }

    void S() {
        if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
            if (curlyController()) {
                return;
            } else {
                System.out.println("2Something is wrong.. " +
curToken.getTokenType());
                System.exit(0);
            }
        }
        S1();
    }

    void S1() {
        if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
            System.out.println("end of file");
            if (curlyController()) {
                System.exit(0);
            } else {
                System.out.println("3Something is wrong.. " +
```

```
curToken.getTokenType());
                System.exit(0);
            }
            return;
        } else if (curToken.getTokenType().equals(TokenType.RIGHT_CURLY)) {
            rightCurly++;
        } else if (curToken.getTokenType().equals(TokenType.LEFT_CURLY)) {
            leftCurly++;
        }

        //WHILE

        else if (curToken.getTokenType().equals(TokenType.WHILE)) {
            curToken = scanner.nextToken();
            while (curToken == null) {
                curToken = scanner.nextToken();
            }
            if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                if (curlyController()) {
                    System.exit(0);
                } else {
                    System.out.println("4Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else if (curToken.getTokenType().equals(TokenType.LEFT_PAR))
{
                Boolean();
                //Exp();
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }

                if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                    if (curlyController()) {
                        System.exit(0);
                    } else {
                        System.out.println("5Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                    System.exit(0);
                } else if
(curToken.getTokenType().equals(TokenType.RIGHT_PAR)) {

                    curToken = scanner.nextToken();
                    while (curToken == null) {
                        curToken = scanner.nextToken();
                    }
                    if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                        if (curlyController()) {
                            System.exit(0);
                        } else {
                            System.out.println("6Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                        System.exit(0);
```

```
                    } else if
(curToken.getTokenType().equals(TokenType.LEFT_CURLY)) {
                        leftCurly++;
                    } else {
                        System.out.println("7Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                } else {
                    System.out.println("8Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else {
                System.out.println("9Something is wrong.. " +
curToken.getTokenType());
                System.exit(0);
            }
        }

        //IF

        else if (curToken.getTokenType().equals(TokenType.IF)) {
            curToken = scanner.nextToken();
            while (curToken == null) {
                curToken = scanner.nextToken();
            }

            if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                if (curlyController()) {
                    System.exit(0);
                } else {
                    System.out.println("10Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else if (curToken.getTokenType().equals(TokenType.LEFT_PAR))
{
                Boolean();
                //Exp();
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }

                if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                    if (curlyController()) {
                        System.exit(0);
                    } else {
                        System.out.println("11Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                    System.exit(0);
                } else if
(curToken.getTokenType().equals(TokenType.RIGHT_PAR)) {

                    curToken = scanner.nextToken();
                    while (curToken == null) {
                        curToken = scanner.nextToken();
```

```
                    }
                    if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                        if (curlyController()) {
                            System.exit(0);
                        } else {
                            System.out.println("12Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                        System.exit(0);
                    } else if
(curToken.getTokenType().equals(TokenType.LEFT_CURLY)) {
                        leftCurly++;
                    } else {
                        System.out.println("13Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                } else {
                    System.out.println("14Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else {
                System.out.println("15Something is wrong.. " +
curToken.getTokenType());
                System.exit(0);
            }
        }

        //IDENITIFIER

        else if (curToken.getTokenType().equals(TokenType.IDENITIFIER)) {
            curToken = scanner.nextToken();
            while (curToken == null) {
                curToken = scanner.nextToken();
            }
            if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                if (curlyController()) {
                    System.exit(0);
                } else {
                    System.out.println("16Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
                System.exit(0);
            } else if (curToken.getTokenType().equals(TokenType.EQUAL)) {
                Exp();
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }
                if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                    if (curlyController()) {
                        System.exit(0);
                    } else {
                        System.out.println("17Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
```

```
                    }
                    System.exit(0);
                } else if
(curToken.getTokenType().equals(TokenType.SEMI_COLON)) {

                } else {
                    System.out.println("18Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else {
                System.out.println("19Something is wrong.. " +
curToken.getTokenType());
                System.exit(0);
            }
        }

        //OUT

        else if (curToken.getTokenType().equals(TokenType.OUT)) {
            curToken = scanner.nextToken();
            while (curToken == null) {
                curToken = scanner.nextToken();
            }
            if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                if (curlyController()) {
                    System.exit(0);
                } else {
                    System.out.println("20Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else if (curToken.getTokenType().equals(TokenType.LEFT_PAR))
{
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }

                if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                    if (curlyController()) {
                        System.exit(0);
                    } else {
                        System.out.println("21Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                    System.exit(0);
                } else if
(curToken.getTokenType().equals(TokenType.IDENITIFIER)) {

                    curToken = scanner.nextToken();
                    while (curToken == null) {
                        curToken = scanner.nextToken();
                    }

                    if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                        if (curlyController()) {
                            System.exit(0);
```

```
                        } else {
                            System.out.println("22Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                        System.exit(0);
                    } else if
(curToken.getTokenType().equals(TokenType.RIGHT_PAR)) {
                        curToken = scanner.nextToken();
                        while (curToken == null) {
                            curToken = scanner.nextToken();
                        }

                        if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                            if (curlyController()) {
                                System.exit(0);
                            } else {
                                System.out.println("23Something is wrong..
" + curToken.getTokenType());
                                System.exit(0);
                            }
                            System.exit(0);
                        } else if
(curToken.getTokenType().equals(TokenType.SEMI_COLON)) {

                        } else {
                            System.out.println("24Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }

                    } else {
                        System.out.println("25Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                } else {
                    System.out.println("26Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            } else {
                System.out.println("27Something is wrong.. " +
curToken.getTokenType());
                System.exit(0);
            }
        }

        //IN

        else if (curToken.getTokenType().equals(TokenType.IN)) {
            curToken = scanner.nextToken();
            while (curToken == null) {
                curToken = scanner.nextToken();
            }
            if (curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                if (curlyController()) {
                    System.exit(0);
                } else {
                    System.out.println("28Something is wrong.. " +
```

```
curToken.getTokenType());
                    System.exit(0);
                }


            } else if (curToken.getTokenType().equals(TokenType.LEFT_PAR))
{
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }

                if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                    if (curlyController()) {
                        System.exit(0);
                    } else {
                        System.out.println("29Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                    System.exit(0);
                } else if
(curToken.getTokenType().equals(TokenType.IDENITIFIER)) {

                    curToken = scanner.nextToken();
                    while (curToken == null) {
                        curToken = scanner.nextToken();
                    }

                    if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                        if (curlyController()) {
                            System.exit(0);
                        } else {
                            System.out.println("30Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                        System.exit(0);
                    } else if
(curToken.getTokenType().equals(TokenType.RIGHT_PAR)) {
                        curToken = scanner.nextToken();
                        while (curToken == null) {
                            curToken = scanner.nextToken();
                        }

                        if
(curToken.getTokenType().equals(TokenType.END_OF_FILE)) {
                            if (curlyController()) {
                                System.exit(0);
                            } else {
                                System.out.println("31Something is wrong..
" + curToken.getTokenType());
                                System.exit(0);
                            }
                            System.exit(0);
                        } else if
(curToken.getTokenType().equals(TokenType.SEMI_COLON)) {

                        } else {
```

```
                                System.out.println("32Something is wrong.. " +
curToken.getTokenType());
                                System.exit(0);
                            }

                        } else {
                            System.out.println("33Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                    } else {
                        System.out.println("34Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);
                    }
                } else {
                    System.out.println("35Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            }


        }

        //EXPRESSION

        void Exp() {
            String rezerve = "";
            boolean control = true;
            while (control) {
                if
(String.valueOf(scanner.chNext).equals(TokenType.RIGHT_PAR.getText()) ||

String.valueOf(scanner.chNext).equals(TokenType.SEMI_COLON.getText())) {

                    control = false;

                }else{
                    curToken = scanner.nextToken();
                    while (curToken == null) {
                        curToken = scanner.nextToken();
                    }

                    if (curToken.getTokenType().equals(TokenType.END_OF_FILE))
{
                        if (curlyController()) {
                            System.exit(0);
                        } else {
                            System.out.println("36Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                        System.exit(0);
                    } else if
(!(String.valueOf(scanner.chNext).equals(TokenType.RIGHT_PAR.getText()) ||

String.valueOf(scanner.chNext).equals(TokenType.SEMI_COLON.getText()))) {
                        rezerve += curToken.getText();
                    } else if
((String.valueOf(scanner.chNext).equals(TokenType.RIGHT_PAR.getText()) ||
```

```
String.valueOf(scanner.chNext).equals(TokenType.SEMI_COLON.getText()))) {
                    rezerve += curToken.getText();
                    //System.out.println(rezerve);
                    break;
                } else {
                    System.out.println(rezerve + " 37Something is wrong.. "
+ curToken.getTokenType());
                    System.exit(0);
                }
            }

        }
        return;
    }

    //curlyController

    boolean curlyController() {
        boolean control = false;
        if (rightCurly == leftCurly) {
            control = true;
        }
        return control;
    }
    void ExpTail(){
    }

    void Term() {
    }

    void TermTail() {
    }

    void Factor() {
    }

    void FactorTail() {
    }

    void Id() {
    }

    void Char() {
    }

    void Num() {
    }

    void Boolean() {
        boolean control = true,control2=true;
        String booleanValue = "",rezerve="";
        curToken = scanner.nextToken();
        while (curToken == null) {
            curToken = scanner.nextToken();
        }
        if (!(curToken.getTokenType().equals(TokenType.NUMBER) ||
curToken.getTokenType().equals(TokenType.IDENITIFIER))) {

            System.out.println(" 38Something is wrong.. " +
curToken.getTokenType());
```

```
            System.exit(0);

        } else {
            if (curToken.getTokenType().equals(TokenType.NUMBER) ||
curToken.getTokenType().equals(TokenType.IDENITIFIER)) {
                rezerve+=curToken.getText();
                curToken = scanner.nextToken();
                while (curToken == null) {
                    curToken = scanner.nextToken();
                }
                nextToken=scanner.nextToken();
                while (nextToken == null) {
                    nextToken = scanner.nextToken();
                }
                    if
(nextToken.getTokenType().equals(TokenType.IDENITIFIER)||
nextToken.getTokenType().equals(TokenType.NUMBER)) {
                        booleanValue+=curToken.getText();

rezerve=rezerve+curToken.getText()+nextToken.getText();
                        for (BooleanOperationType type :
BooleanOperationType.values()) {
                            if (booleanValue.equals(type.getText())) {
                                //System.out.println(rezerve+"
"+type.getText()+" "+type);
                                control2=false;
                            }
                        }
                        if(control2){
                            System.out.println(" 39Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                    } else
if(nextToken.getTokenType().equals(TokenType.WHILE)||
nextToken.getTokenType().equals(TokenType.IF)||

nextToken.getTokenType().equals(TokenType.OUT)||
nextToken.getTokenType().equals(TokenType.IN)){
                        System.out.println(" 40Something is wrong.. " +
curToken.getTokenType());
                        System.exit(0);

                    } else {

booleanValue=booleanValue+curToken.getText()+nextToken.getText();

rezerve=rezerve+curToken.getText()+nextToken.getText();
                        for (BooleanOperationType type :
BooleanOperationType.values()) {
                            if (booleanValue.equals(type.getText())) {
                                curToken= scanner.nextToken();
                                while (curToken == null) {
                                    curToken = scanner.nextToken();
                                }
                                rezerve+=curToken.getText();
                                //System.out.println(rezerve+"
"+type.getText()+" "+type);
                                control2=false;
                            }
                        }
```

```
                        if(control2){
                            System.out.println(" 41Something is wrong.. " +
curToken.getTokenType());
                            System.exit(0);
                        }
                    }
                } else {
                    System.out.println(" 42Something is wrong.. " +
curToken.getTokenType());
                    System.exit(0);
                }
            }

        return;
    }
}
```

```
//is responsible for scanning for tokens (it will return tokens)
//to the parser.
public class Scanner {
    private ProgramText source;
    public String string = "";
    public char chNext;

    Scanner(ProgramText source) {
        this.source = source;
    }

    boolean isSpecial(char chNext) {
        boolean control = false;
        if (!Character.isWhitespace(chNext)) {
            for (TokenType type : TokenType.values()) {
                if (String.valueOf(chNext).equals(type.getText())) {
                    control = true;
                    break;
                }
            }
        }
        return control;
    }

    //Scanner will ask the Source for characters and one a sequence of
    //characters form a token it will return immediately.
    //Scanner needs to know some of rules (for example, what constitutes
    //a number, what constitutes an identifier and so forth)
    Token nextToken() {
        Token token;

        char chCur = source.curChar();
        chNext = source.nextChar();
        while (Character.isWhitespace(chCur)) {
            chCur = source.curChar();
            chNext = source.nextChar();
        }
        if (!Character.isWhitespace(chCur)) {
            for (TokenType type : TokenType.values()) {
                if (String.valueOf(chCur).equals(type.getText())) {
                    token = new SpecialToken(source, String.valueOf(chCur),
type);
                    return token;
                }
```

```
            }
            if (Character.isDigit(chCur)) {
                //number token
                //System.out.println(chCur);
                string += chCur;
                if (isSpecial(chNext)) {
                    token = new NumberToken(source, string,
TokenType.NUMBER);
                    string = "";
                    return token;
                }

            } else if (Character.isLetter(chCur)) {
                //identifier token
                string += chCur;
                if (isSpecial(chNext)) {
                    if (string.equals(TokenType.WHILE.getText())) {
                        //System.out.println(TokenType.WHILE.getText());
                        token = new KeywordToken(source, string,
TokenType.WHILE);
                        string = "";
                        return token;
                    } else if (string.equals(TokenType.IF.getText())) {
                        token = new KeywordToken(source, string,
TokenType.IF);
                        string = "";
                        return token;
                    } else if (string.equals((TokenType.OUT.getText()))) {
                        token = new KeywordToken(source, string,
TokenType.OUT);
                        string = "";
                        return token;
                    } else if (string.equals((TokenType.IN.getText()))) {
                        token = new KeywordToken(source, string,
TokenType.OUT);
                        string = "";
                        return token;
                    } else {
                        token = new IdentifierToken(source, string,
TokenType.IDENITIFIER);
                        string = "";
                        return token;
                    }


                }

            } else {
                token = new EOFToken(source);
                return token;
            }
        }

        return null;


    }


}
```

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

//the purpose of the ProgramText class is to abstract away
//from where the program is coming. ProgramText provides a
//single character to the Scanner class when asked for.
//it reads the program (from a file or as String) line by line
//from top to bottom
public class ProgramText {

    //private BufferedReader reader;
    public String progText;
    private int curPos, rez = 0;
    public static char EOF = 'Ł';

    ProgramText() {

        curPos = -1;

        try {
            progText = readWholeProgram();

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    private String readWholeProgram() throws IOException {
        return new String(Files.readAllBytes(Paths.get("program2.txt")));

    }

    char curChar() {
        if (curPos == -1)
            curPos++;

        if (curPos == progText.length())
            return EOF;
        if (rez <= progText.length()) {
            return progText.charAt(curPos);
        }
        return EOF;
    }

    char nextChar() {
        curPos++;
        rez = curPos;
        if (rez == progText.length())
            return EOF;

        for (int i = rez; i < progText.length(); i++) {
            if (Character.isWhitespace(progText.charAt(rez))) {
                rez++;
                if (rez == progText.length()) {
                    return EOF;
                }

            }
```

```
        }

        if (rez == progText.length()) {
            return EOF;
        }
        if (rez <= progText.length()) {
            return progText.charAt(rez);
        }


        return EOF;

    }


}
```

```
public enum BooleanOperationType {

EQUAL_AND_EQUAL("=="),NOT_EQUAL("!="),LESS_AND_EQUAL("<="),GRATER_AND_EQUAL
(">="),
    LESS("<"),GRATER(">");

    public String getText() {
        return text;
    }

    private String text;

    BooleanOperationType(String text) {
        this.text = text;
    }

    BooleanOperationType() {
        this.text = this.toString();
    }
}
```

```
public class EOFToken extends Token {

    EOFToken(ProgramText source) {
        super(source);
        type = TokenType.END_OF_FILE;

    }

}
```

```
public class IdentifierToken extends Token{

   IdentifierToken(ProgramText source,String text, TokenType type) {
      super(source);
      this.text=text;
      this.type=type;


   }


}
```

```java
public class KeywordToken extends Token {
    KeywordToken(ProgramText source,String text, TokenType type) {
        super(source);
        this.text=text;
        this.type=type;
    }
}
```

```java
public class NumberToken extends Token{

   NumberToken(ProgramText source,String text,TokenType type) {
      super(source);
      this.text=text;
      this.type=type;
   }
}
```

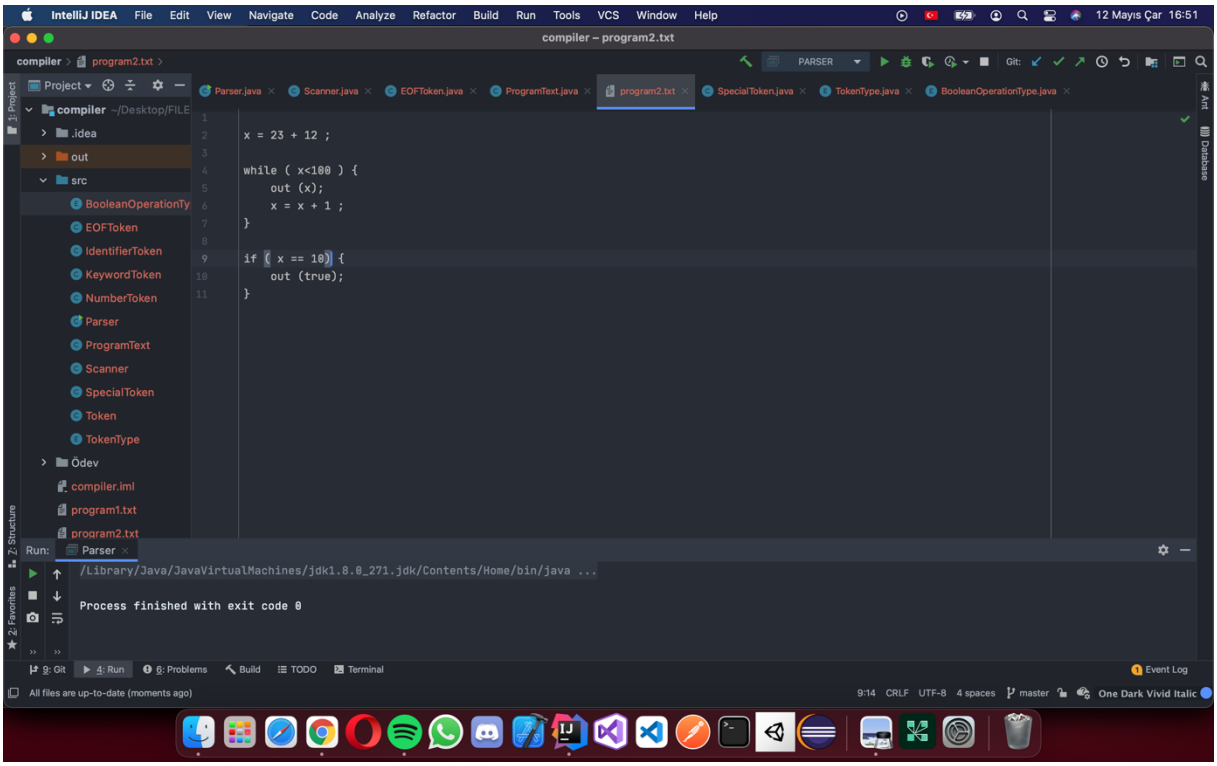
```java
public class SpecialToken extends Token {

    SpecialToken(ProgramText source, String text, TokenType Specialtype) {
        super(source);
        this.text = text;
        this.type = Specialtype;
        // TODO Auto-generated constructor stub
    }

}
```

```java
public class Token {

   public TokenType type;
   public String text;
   private ProgramText source;

   Token(ProgramText source){
      this.source = source;
   }
   public TokenType getTokenType() {
      return type;
   }
   public String getText() {
      return text;
   }
}
```

```java
public enum TokenType {
    LEFT_CURLY("{"), RIGHT_CURLY("}"), LEFT_PAR("("), RIGHT_PAR(")"),
    EQUAL("="), SEMI_COLON(";"), LESS_THAN("<"),GRATER_THAN(">"),
    MINUS("-"), MULTIPLY("*"), DIVIDE("/"), PLUS("+"),NOT("!"),

    WHILE("while"), IF("if"), OUT("out"), IN("in"),
    IDENITIFIER, NUMBER, END_OF_FILE;

    public String getText() {
        return text;
    }

    private String text;

    TokenType(String text) {
```

```
            this.text = text;
        }

        TokenType() {
            this.text = this.toString();
        }
}
```

True



False