Task

In this project, you need to write in **Java** a simple compile time sanity checker. The sanity checker parses an input program written in "**weeJava**" and finds its compile time errors, similarly to what Eclipse does for you. **weeJava** is a subset of **Java** programming language (**see the weeJava specification document**). You can compare the result from your checker with Eclipse.

Questions

Questions in this assignment are organized into several steps. Note that even if you do not complete fully an earlier question, you can still proceed to later ones.

Question 1 (30 marks)

To complete this project, you need to parse the program and convert it into a sequence of tokens (strings with an assigned and thus identified meaning). In this step, we first define some help methods.

a. Create a class named **Q1** and add the **main** method

b. Create an **enum** type named **TokenType** for **weeJava** operators, symbols, keyworks, identifiers and literals: OP_MULTIPLY, OP_DIVIDE, OP_MOD, OP_ADD...

```
public enum TokenType {OP_MULTIPLY, OP_DIVIDE, OP_MOD, OP_ADD, …., IDENTIFIER, DOUBLE, STRING, BOOLEAN}
```

You should fulfil the missing part by yourself.

(Don't remember what an **enum** type is?

https://canvas.qub.ac.uk/courses/8455/files/379872?module_item_id=190922)

c. Add a method called **getOp**, which takes a parameter **ch** of type **char** and returns a value of type **TokenType**. If **ch** is one of the single char operator (e.g., +,-,*,/) in **weeJava, getOp** returns the corresponding operator name; otherwise, it returns **null**. For example, **getOp('+')** should return **TokenType.OP_ADD**.

(Don't know what **null** is?

https://canvas.qub.ac.uk/courses/8455/files/379869?module_item_id=190924)

d. Add a method (also) called **getOp**, which takes a parameter **s** of type **String** and returns a value of type **TokenType**. If **s** is one of the double char operator (e.g. >=, ==) in **weeJava**, **getOp** returns the corresponding operator name; otherwise, it returns **null**.

e. Add a method called **getSymbol**, which takes a parameter **ch** of type **char** and returns a value of type **TokenType**. If **ch** is one of the symbols (e.g., (,),[,}) in **weeJava**, **getSymbol** returns the corresponding symbol name; otherwise, it returns **null**.

f. Add a method called **getKeyword**, which takes a parameter **s** of type **String** and returns a value of type **TokenType**. If **s** is one of the keywords in **weeJava**, **getKeyword** returns the corresponding operator name; otherwise, it returns **null**.

g. Add a new method called **isLetter** that takes a parameter **ch** of type char and it returns a value of type **boolean** to indicate if the input is one of the: abcdefghijklmnopqrtsuvwxyzABCDEFGHIJKLMNOPQRTSUVWXYZ. For example, **isLetter('a')** returns **true** and **isLetter('/')** returns **false**

h. Add a new method called **isDigit** that takes a parameter **ch** of type **char** and it returns a value of type **boolean** to indicate if the input is one of the: 0123456789. For example, **isDigit('0')** returns **true** and **isDigit('c')** returns **false**

i. Add a new method called **isWhiteSpace** that takes a parameter **ch** of type **char** and it returns **true** if **ch** is whitespace; otherwise its returns **false**. Whitespace includes: **space**, **tab space** and **line break**.

*Note: To finish this project, you are likely to **also** add other methods of your own. Please put proper comments to justify the use of them.*

Question 2 (25 marks)

You need to use these help methods defined in Question 1 to complete this step.

a. Create a new class called **Q2**; copy the code from **Q1** to **Q2**.

b. In **Q2**, add a method called **scan**, which takes a parameter **prog** of type **String**. It parses the string into tokens and prints them out.

For example, let us consider the following string input

```
String prog = "public class HelloWorld {\n" +
              "  public static void main(String[] args) {\n" +
              "   int a=10; \n" +
              "   double b=0.5; \n" +
              "   double c=a+b*10;\n" +
              "   System.out.println(\"The num is: \" + c);\n" +
              " }\n" +
              "}\n";
```

The output from **scan(prog)** looks like the following: the 1st column is **line number**,

the 2nd column is **token type** and the 3rd column is the corresponding text (**token name**).

```
1, KEYWORD_PUBLIC, public
1, KEYWORD_CLASS, class
1, IDENTIFIER, HelloWorld
1, LEFT_BRACE, {
2, KEYWORD_PUBLIC, public
2, KEYWORD_STATIC, static
2, KEYWORD_VOID, void
2, KEYWORD_MAIN, main
2, LEFT_PAREN, (
2, KEYWORD_STRING, String
2, LEFT_BRACKET, [
2, RIGHT_BRACKET, ]
2, IDENTIFIER, args
2, RIGHT_PAREN, )
2, LEFT_BRACE, {
3, KEYWORD_INT, int
3, IDENTIFIER, a
3, OP_ASSIGN, =
3, INTEGER, 10
3, SEMICOLON, ;
4, KEYWORD_DOUBLE, double
4, IDENTIFIER, b
4, OP_ASSIGN, =
4, DOUBLE, 0.5
4, SEMICOLON, ;
5, KEYWORD_DOUBLE, double
5, IDENTIFIER, c
5, OP_ASSIGN, =
5, IDENTIFIER, a
5, OP_ADD, +
5, IDENTIFIER, b
5, OP_MULTIPLY, *
5, INTEGER, 10
5, SEMICOLON, ;
6, IDENTIFIER, System
```

```
6, OP_DOT, .
6, IDENTIFIER, out
6, OP_DOT, .
6, IDENTIFIER, println
6, LEFT_PAREN, (
6, STRING, "The num is: "
6, OP_ADD, +
6, IDENTIFIER, c
6, RIGHT_PAREN, )
6, SEMICOLON, ;
7, RIGHT_BRACE, }
8, RIGHT_BRACE, }
```

c. In the **main** method, read the **Q2Example.java** file into a string, which is then passed to **scan** method as an argument. You should design **Q2Example.java** by yourself such that it contains all token types in **weeJava**.

Question 3 (5 marks)

a. Create a new class called **Q3**; copy the code from **Q2** to **Q3**.

b. Extend **scan** method so that it skips comments: there are two kinds of comments in **weeJava**: // and /* */

For example, if we consider the following string

```
String prog = "public class HelloWorld {\n" +
              "  public static void main(String[] args) {\n" +
              "     // to print out hello world\n" +
              "     System.out.println(\"Hello World!\");\n" +
              " }\n" +
              "}\n";
```

The output from **scan(prog)** will now look like the following:

```
1, KEYWORD_PUBLIC, public
1, KEYWORD_CLASS, class
1, IDENTIFIER, HelloWorld
1, LEFT_BRACE, {
2, KEYWORD_PUBLIC, public
2, KEYWORD_STATIC, static
2, KEYWORD_VOID, void
2, KEYWORD_MAIN, main
```

```
2, LEFT_PAREN, (
2, KEYWORD_STRING, String
2, LEFT_BRACKET, [
2, RIGHT_BRACKET, ]
2, IDENTIFIER, args
2, RIGHT_PAREN, )
2, LEFT_BRACE, {
4, IDENTIFIER, System
4, OP_DOT, .
4, IDENTIFIER, out
4, OP_DOT, .
4, IDENTIFIER, println
4, LEFT_PAREN, (
4, STRING, "Hello World!"
4, RIGHT_PAREN, )
4, SEMICOLON, ;
5, RIGHT_BRACE, }
6, RIGHT_BRACE, }
```

c. In the **main** method, read the **Q3Example.java** file into a string, which is then passed to **scan** method as an argument. You should design **Q3Example.java** by yourself in the way such that it has comments of both // and /* */ forms.

Question 4 (20 marks)

a. Create a new class called **Q4**; copy and paste the code from **Q3** to **Q4**.

b. Add a method called **variableNameCheck** that takes parameter **varName** of type **String** and returns **true**/**false** to indicate if **varName** is a valid variable name in **weeJava**. A valid variable name in **weeJava**
   - can only contain letter, digit and/or '_' and
   - it must start with '_' or a letter

c. Extend the code in **Q4** so that your sanity checker can detect invalid variable names

   For example, if at the 5th line of the **weeJava** program, there is

```
int 0var123;
```

   Then, at the end of your sanity checker output there should be something like

   Line 5: invalid identifier name "0var123"

   In the **main** method, read the **Q4Example1.java** file into a string and apply your sanity

checker on it. You should design **Q4Example1.java** by yourself such that it has invalid variable names.

d. Extend your code to detect various (at least 10) **types** of compile time errors. You should prepare another 9 **weeJava** example programs, each having a type of compile time error (they should differ from Question 4 (c)). In the **main** method of **Q4**, apply your sanity checker on each of them.

*Note: There are at least two choices to complete this step: 1) implement the compilation error detection directly inside the **scan** method, and (2) save the token line numbers/types/names from **scan** (using array(s)) and define method(s) to analyze these tokens after the scanning. Choice (2) may help you in Question 5.*

Question 5 (20 marks)

a. Create a new class called **Q5**; copy the code from **Q4** to **Q5**.

b. Suppose that a **weeJava** program contain **only** arithmetic operations between integers in the **main** method and in the end it prints out the result (*that is, there is no selection/loop in the code and the only method call is System.out.println*). Can you extend **Q5** to print out all integer variables and their values in the **weeJava** code.

For example, given the following **weeJava** code

```
public class Example {

    public static void main(String[] args) {

        int a = 190;

        int b =5;

        int c = 10;

        int d = 5;

        int result = a+b-c*d/2;

        System.out.println("The result is: " + result);

    }

}
```

The output of your program should be:

a : 190

b : 5

c : 10

d : 5

result : 170

c. You should propose a **weeJava** example program **Q5Example.java** as input to demonstrate that your code works.

Submitting your work

On completion, you should **zip** and upload your **.java files** to Canvas (.class files will **NOT** be accepted or marked). The number of files you have will depend on the parts you have completed. The following list highlights the files you should have for each part:

Completed Question 1	**Completed Question 2**	**Completed Question 3**	**Completed Question 4**	**Completed Question 5**
Q1.java TokenType.java	Q2.java Q2Example.java	Q3.java Q3Example.java	Q4.java Q4Example1.java Q4Example2.java Q4Example3.java Q4Example4.java Q4Example5.java Q4Example6.java Q4Example7.java Q4Example8.java Q4Example9.java Q4Example10.java	Q5.java Q5Example.java

Upload Instructions

1. Log into Canvas and navigate to the **Assignments** section of CSC1025. There should be an assignment called **Individual Course Work**

2. You should upload your zip

Remember that it is **YOUR** responsibility to submit the **correct** files **on time** (before the submission page closes).