**Quiz 3**

1. **Briefly explain the compiler’s scanner works?**

It converts the original program’s stream of characters into a stream of tokens. Spaces or comments in the program are removed in this process.

It uses Regular Expressions to define tokens, and uses Finite Automata to recognize tokens.

1. **Give an example of 5 tokens.**

* Operators: Factors like ‘=’, ‘+’, ‘-‘, ‘>’ and so on.
* Keywords: Word factors like “if”, “while”, “for” and so on.
* Identifiers: Factors such as pi in program fragment const pi = 3.14;
* Numeric literals: Factors such as 43, 6.035, -3.6e10 and so on.
* Character literals: Factors like ‘a’, ‘~’, and so on.

1. **What is the general approach for the scanner to recognize a token?**
2. Build a deterministic finite automaton (DFA) from regular expression E
3. Execute the DFA to determine whether an input string belongs to L(E)
4. **What are regular definitions?**

Regular definitions are regular expressions associated with suitable names.

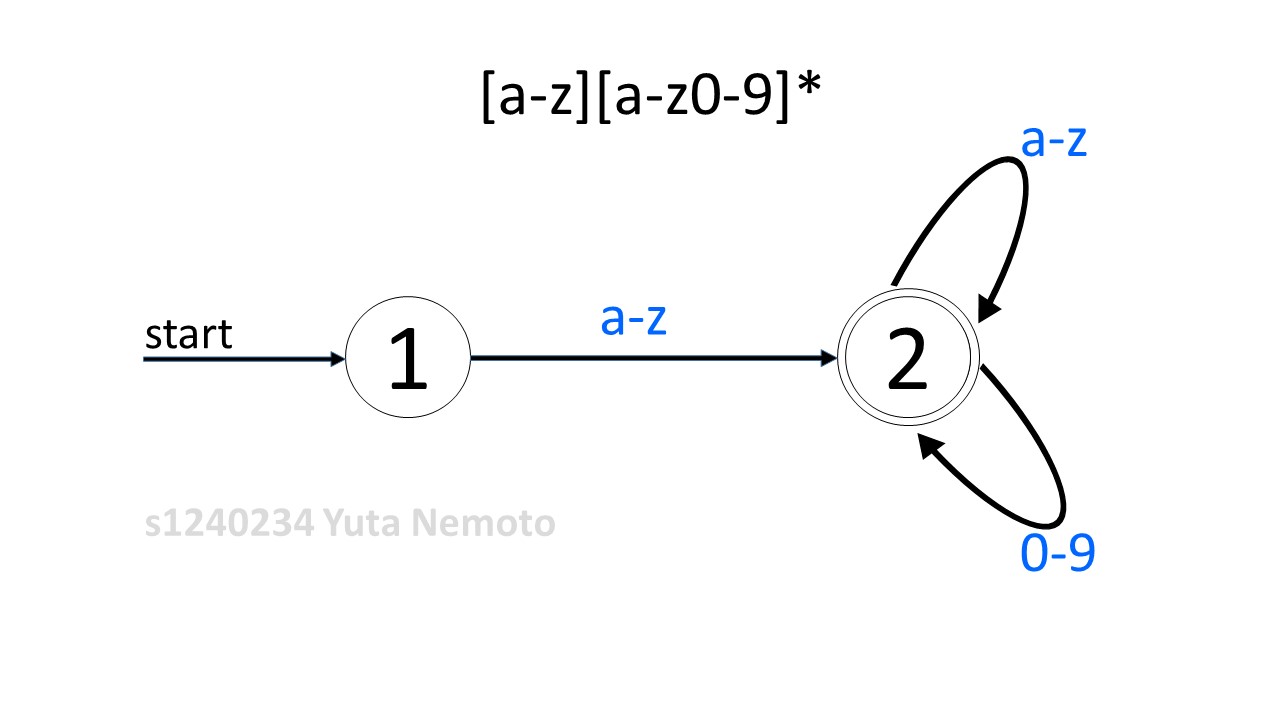
For example, the set of identifiers in Java can be expressed by the Following regular definition:

Letter → A|B|…|Z|a|b|…|z

Digit → 0|1|2|…|9

Id → letter (letter|digit)\*

1. **Write an automaton for the regular expression [a-z][a-z0-9] \***



1. **What is the general approach for writing a scanner?**

The construction is done automatically by a tool such as the Unix program lex.

1. **Briefly explain how to use Lex for scanner writing?**

Using the source program language grammar to write a simple lex program and save it in a file named lex.l

Using the unix program lex to compile lex.l resulting in a C (scanner) program named lex.yy.c

Compiling and linking the C program lex.yy.c in a normal way resulting the required scanner.