# FAB LANGUAGE

## DATA TYPES:

- **Number** (includes int, double, float) **Declared as:** Number (type) VARIABLE NAME
  **RE:**
  Number( i ): (“^ [0-9](+|-) $”)
  Number( d / f): (“^[+-]?\\d*\\.?\\d+$”)
  (double and float have same values the difference is values allowed after decimal point and rounding off)
- **Word** (used as string)
  **RE:** (“ ^ [ . ] $”)
- **Bool**
  **RE:** (“^ [true false]{1} $”)
- **Char**
  **RE:** (“^ [0-9 a-b A-B]{1} $”)

## OPERATORS: (Operator precedence same as c#)

- **ARITHMATIC:** + , - , * , / , %

  **Class: MDM**

  **1st Level Precedence:** *, /, %

  **Class: PM**

  **2nd Level Precedence:** +, -

- **RELATIONAL:** < , > , <= , >= , != , == **Precedence Level:** Equal

- **LOGICAL:** && , ||

  **1st Level Precedence:** &&

  **2nd Level Precedence:** ||

- **ASSIGNMENT:** = , +=, -= , /= , %=

  **Class:** Assignment Operators **Precedence**
  **Level:** Equal

- **UNARY OPERATOR:** ++, --, !

  **Class: INC DEC**

  **1st Level Precedence:** ++, --

  **Class: Not**

  **2nd Level Precedence:** !

## KEYWORDS:

- **Pass** (used as return)
- **Class**
- **Interface**
- **Void**
- **Return**
- **New**
- **Static**
- **Abstract**

## LOOPS:

Class: **FOR**
Syntax: **for** (initialization ; condition ; increment/decrement ) / (statement)

Class: **WHILE**
Syntax: **while :** condition / statement

Class: **DO WHILE**
Syntax: **do /** statement; **while :** condition

## CONDITIONS:

Class: **IF**
Syntax: **if** : condition / statement

Class: **ELSE**
Syntax: **else** / statement

Class: **ELIF**
Syntax: **elf** (used as elseif) : condition / statement

# ARRAYS:

Class: **Array**
Syntax: number (i) [] newArr = [1,2,3,4,5];
Syntax: word [] newArr1 = [“Apple”, “Banana”, “Orange”];
Syntax: // 2D Array number (i) [[]] array2 = [[1,2,3],[4,5,6]];

# COMMENTS:

For Single Line Comments:
Syntax: // This is single Line Comments

For Multiline Comments:
Syntax: /* This is Multiline Comments */

# ACCESS MODIFIERS:

- **Public**
- **Private**
- **Sealed**
- **Protected**

# FUNCTION:

DATA TYPE ( parameters ) : pass

# END OF LINE(EOL):

**Semicolon ( ; )**

# OOP:

**Class** (CLASS NAME)

{

DATA TYPE METHOD NAME ( ) : return

}

Class CHILD CLASS => PARENT CLASS

{

}

**Interface** INTERFACE NAME

{

METHOD NAME( );

}

Class CHILD CLASS => INTERFACE NAME

{

METHOD NAME( ) : return

}

**// Overriding Methods**

Public number (i) addNum(number (i) firstNum, number (i) secondNum):

pass firstNum + secondNum

Public number(i) addNum(number (i) firstNum, number (i) secondNum,
number(i) thirdNum):

pass firstNum + secondNum + thirdNum;

Main( )

{

CLASS NAME OBJECT NAME = new CLASS NAME ( );

NAME = OBJECT NAME-METHOD NAME ( parameter values );

}