Limitation of Command Line Argument :

As we know by using Command Line Argument, we can pass some value at runtime, These values are stroed in String array variable and then only the exceution of the program will be started.
The basic limitation of Command line Argumenet is, we **can't ask to enter the value from our end user as shown in the Program.**

```
CommandLimitation.java :
-----------------------------
void main(String [] args)
{
  IO.print("Enter your Gender :"); //will executed after providing the value
  char gender =  args[0].charAt(0);
  IO.println("Your gender is :"+gender);
}
```

* In order to avoid this limitation, Java software people has introduced IO.readln() method to accept the input from the user.

* The following are the ways to accept the input from the user :

  a) By using DataInputStream class of java.io package
  b) By using BufferedReader class of java.io package
  c) By using Console class of java.io package (reading the password)
  d) By using System.in.read() method
  e) By using java.util.Sacnner class
  f) By using java.lang.IO class.

How to read the client data by using IO class :

* IO class has provided a predefined static method called **readln()** through which we can read the data from the client in **String format** because the return type of this method is String.

  public static String readln()

WAP to read the name from the Client :

```
ReadName.java
-------------------
 void main()
 {
      IO.print("Please enter your name :");
      String name = IO.readln();
    IO.println("Your Name is :"+name);
 }
```

WAP to read your age from the client :

```
ReadAge.java
----------------
 void main()
 {
      IO.print("Please enter your Age :");
      String ag = IO.readln();
    IO.println("Your Age is :"+ag);

    //Converting String(ag) to integer(age)
      int age = Integer.parseInt(ag);

    if(age>18)
      {
          IO.println("Go for a movie");
      }
      else
      {
          IO.println("Try After some year");
      }

 }
```

WAP to read the salary from the Client :

```
 void main()
 {
     IO.print("Enter your Salary :");
     String sal = IO.readln();
     IO.println("Your Salary is :"+sal);

    //Converting String into double type
    double salary = Double.parseDouble(sal);

     if(salary >=50000)
     {
         IO.println("Your bonus amount is 5000");
     }
     else
     {
         IO.println("Your bonus amount is 3000");
     }
 }
```

WAP to read gender from the client :

```
 void main()
 {
     /*
     //Approach 1
     IO.print("Enter your Gender[Male/Female] :");
     String gender = IO.readln();
     char gen = gender.charAt(0);
     IO.println("Your Gender is :"+gen);

    */

    //Approach 2
     IO.print("Enter your Gender[Male/Female] :");
     char gender = IO.readln().charAt(0);
     IO.println("Your Gender is :"+gender);

 }
```

Tokens in Java :

  3) Literal
  4) Punctuators (Separators)
  5) Operator

Keyword

A keyword is a predefined word whose meaning is already defined by the compiler.

In java all the keywords must be in lowercase only.

A keyword we can't use as a name of the variable, name of the class or name of the method.

true, false and null look like keywords but actually they are literals.

As of now, we have 67+ keywords in java.

Identifiers :

A name in java program by default considered as identifiers.

Assigned to variable, method, classes to uniquely identify them.

We can't use keyword as an identifier.

Ex:-

```
class Fan
{
  int coil  ;

  void switchOn()
  {
  }
}
```

Here Fan(Name of the class), coil (Name of the field) and switchOn(Name of the Method) are identifiers.

Rules for defining an identifier :

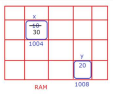
1) Can consist of uppercase(A-Z), lowercase(a-z), digits(0-9), $ sign, and underscore (_)
2) Begins with letter, $, and _
3) It is case sensitive
4) Cannot be a keyword
5) No limitation of length

Variable :

* It is a name given for the memory location.
* It is used to hold some meaningful value.

```
  int x = 10;
  int y = 20;

  x = x + y;
```



* It can change its value during the execution of the program.

  VARIABLE

  VARY + ABLE

  CHANGE + ABLE

Drawback of an ordinary variable :

We have 2 drawbacks :

a) It can hold **only one value** at a a time in **random** memory location.
   Example :

   int x = 12, 90; //Invalid

   In order to avoid this we introduced array concept in java.
   It can hold multiple values in a sequence memory allocation (CACHE)

b) A variable which stores its value in RAM is a volatile memory so, once the program execution is over variable value will be deleted from the RAM so we cannot use variable value back in the future.
   In order to avoid this we introduced "**File Handling**" concept.