

# ESC101 : Fundamental of computing

## Lecture 2 (30 July, 2008)

In the first lecture, we gave definition of an algorithm. An algorithm for a problem is a method to solve a given problem. This method should be a sequence of instructions with the following rules :

1. Each instruction has to be precise, unambiguous. Each instruction should be executable on a *machine*.
2. The algorithm must work for all the instances of the problem.
3. the description of the algorithm must be finite.

We also defined a computer program. A program is description of an algorithm in some computer language : C,C++,JAVA,Assembly,...

Though the prime aim of the course is to teach basic skills for solving problems by computer, that is, the algorithmic approach, we shall spend a few weeks on JAVA. This is because, we would eventually like to see our problems getting solved in REAL life.

The overall strategy in solving a problem would be the following :

- *step 1:* First we shall design an algorithm for the given problem. This will require paper, pen and some creative skills. We shall also try to make sure here that our algorithm satisfies the three properties mentioned above. It would also be advised to convince yourself, at least informally, about the correctness of the algorithm.
- *step 2:* In this step, we translate our algorithm into JAVA language. As discussed in the class, JAVA is a high level language in which it is much easier to express our algorithm rather than in the machine language which is a sequence of 0's and 1's. We have to write our program in very very formal manner, following the rules of (the grammar of) JAVA.
- *step 3:* In the 3rd step, we use JAVA compiler to translate the JAVA program into low level language and then execute/run the program. This step has been explained vaguely at this stage (keeping in mind the background of the audience), and we shall discuss more on it later in the course.

The following is a skeleton of a typical JAVA program.

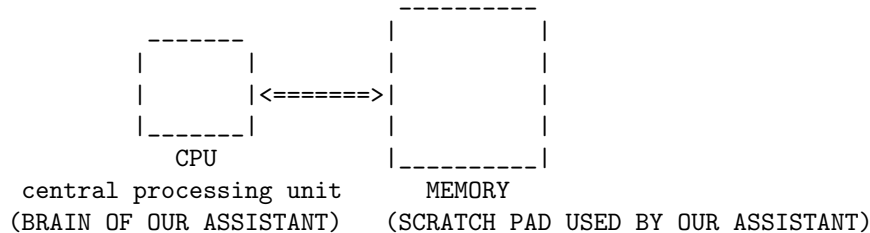
```
class class_name
{
    public static void main(String args[])
    {
=> Instruction1;
    Instruction2;
    Instruction3;
        :
        :
    Instructionk;
    }
}
```

The execution will start from the first instruction pointed by => in sequential manner. The meaning of other words like “class”, “public”, “static” will be explained later. The student should focus on the sequence of Instructions starting from => and understand that the during execution, the program executes these instructions one by one. Please note that we shall use the word statement for an instruction in the JAVA program.

We mentioned above that JAVA program is a sequence of instructions. Another view to visualize a JAVA program is as a sequence of symbols/words. Some of these words have a predefined meaning or

function in JAVA. These are called **keywords** of JAVA. For example class, for, main, void, .... The programmer will introduce some words for its data (and other stuff). These words will be called **identifiers**. These have to be different from these keywords. In today's lecture and future, we shall use yellow color to denote the keywords and white color to represent identifiers.

In order to get a better understanding and insight into JAVA program, the following simplified model of the computer will be useful. It consists of CPU which is the central processing unit (where all the instructions are executed) and Memory which is a storage device.



In the first lecture, we mentioned that a computer can be seen as an office assistant who is capable of simple arithmetic and logical calculations, dumb (just executes *mechanically* the instructions we give to him/her), but works very very fast once we have given the instructions in a very formal manner. In this analogy, memory can be viewed as a scratch pad of our office assistant, and CPU can be viewed as the *dumb/mechanical* brain of our office assistant. In some portion of the memory, the program is kept, and in the remaining portion some data to be processed.

## Data

A typical problem to be solved by computer has some input and some output. The input as well as output is some sort of explicit information called data. For example, consider one of the problem mentioned in the first lecture :

- given two line segments, determine if they intersect.  
The input is the coordinates of the endpoints of each line segment, and output is yes or no.
- given a set of  $n$  points in a plane, compute the smallest enclosing circle.  
The input is the coordinates of all the  $n$  points, and output is coordinates of the center of the circle and its radius.

Based on the above discussion, the following is an alternate view of the program as a data processing unit.

Input ==>    | PROGRAM |    ==> Output

The input to a program would typically be given from the keyboard or from a file. The output will usually be displayed on the monitor/terminal. (Our first toy program was an exception in that it did not have any input, and its output was the sentence "Welcome to IIT Kanpur".) We need to first understand the facilities provided by JAVA for handling (declaring/defining and processing) data.

## Data Declaration

Each data item which we are going to use has to be declared before its use. Declaration means assigning it some name/label, and mentioning its type. For example if we need a data item of type integer, we write the following statement in JAVA.

```
int counter;
```

What does JAVA compiler do when it reads this statement ? It creates a location in memory, gives it a label “counter” so that we can refer to it by this label in future. It also notes down that we shall always store “INTEGER” in this location.

There is nothing special about the word counter. We could have given some other name as well provided it is not some **keyword** of JAVA. We shall mention the rules followed in giving names to **identifiers** in next class.

## Operations on data

The basic operations on a data item would be to access its value and change its value. Due to the second property, we call counter a(n integer) variable.

The following analogy will be very useful in understanding a variable. A variable is equivalent to a box in the memory.

- Declaring a variable  $x$  of type `int` is equivalent to creating a box for holding an integer value and assigning the box the label  $x$ .
- the value of the variable is the contents of the box.
- changing the value of variable, say, from  $a$  to  $b$  is equivalent to replacing the contents of box (at present  $a$ ) by new content ( $b$ ).

Let us see how can we retrieve its value and change the value of a variable. Each variable has to be declared and the assigned some value before we use it. Assigning or changing the value of a variable is done using the operator ‘=’ which is called an assignment operator. Do not confuse it with equality operator in mathematics. For example, `i=1;` will assign value 1 to `i`. In terminology of box, it would mean that we have placed 1 into the box labeled `i`.

Rules of using operator ‘=’ are :

- the left hand side of ‘=’ must be a variable.
- the right hand side should be an expression of the same type as that of left hand side.
- its execution involves calculating the value of the expression the right hand side first and then assigning it to the variable on the left.

As home work, consider the following program, and fill in the blanks. The analogy of variable and box will be useful here.

```
class variable_intro
{
    public static void main(string args[])
    {
        int i;
        int sum;
        i = 1;
        sum = 0;           //at this point the value of i is ...,and the value of sum is ...
        sum = i;           //at this point the value of i is ...,and the value of sum is ...
        i = 3;             //at this point the value of i is ...,and the value of sum is ...
        sum = sum + i;     //at this point the value of i is ...,and the value of sum is ...
        i = i + 1;         //at this point the value of i is ...,and the value of sum is ...
        sum = sum + i;     //at this point the value of i is ...,and the value of sum is ...
    }
}
```