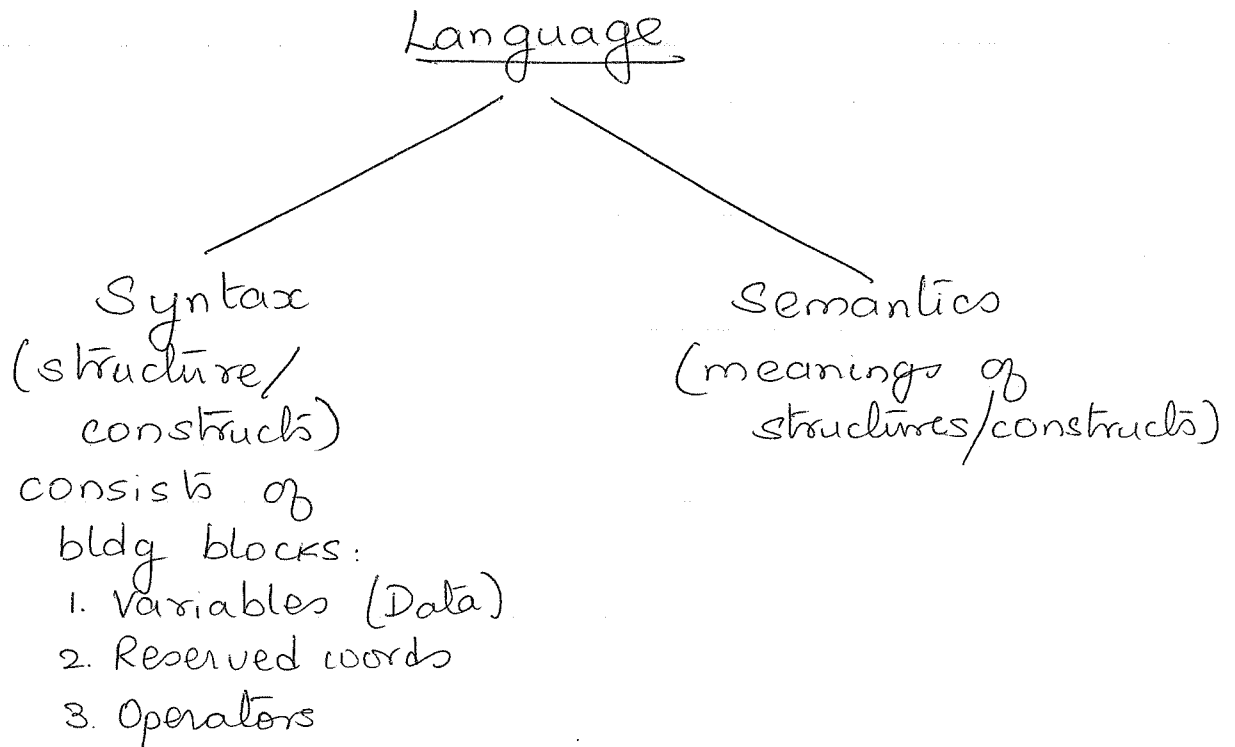


Big Picture:



- Variables are related by operators
- Reserved words actually instruct

Variables are of different types and levels:

- Types: (primitive, lowest levels of each category).
 - integer (short, long unsigned etc)
 - float/double/real
 - boolean
 - character
 - string etc.

- Levels: (capture aggregation of the data types above)
 - Lists, Arrays
 - Trees
 - Graphs etc.

In OO languages such as Java,
all implementations are only via classes.

We first take a look at generic classes and
then take a more in-depth look at classes in Java.

Classes:

Background:

At some point in the programming language
evolutionary scale, people realized that
(a) programs would get incomprehensibly
long and indecipherable if they were written
in on one long sequence, and,

(b) some parts of the code in programs would
be re-used multiple times in exactly the
same format.

(a) & (b) together pushed language development
towards modular or functional programming.

Functions of programs are pieces of code
segments that take in defined inputs and
produce expected outputs - in other words,
they are like programs themselves, except
in comprehensiveness.

This concept of functional or modular programming
has a lot in common with (mechanical)
engineering domains as well:

- Think of any tool (such as a wrench,
a hammer, a drill etc)
- Each class of tools has a defined function
that it performs, given a predetermined input,
producing an expected outcome.

For instance, a wrench takes as input a force provided by the user, operates on a nut (the variable analog in a function) and produces tightening or loosening of the nut, according as the force direction dictates.

A little later in the evolutionary ladder, the realization came that the variables could themselves be captured/encapsulated into the function to produce a far-more rational module \Rightarrow classes.

In our wrench analog, the variable is the nut size that the wrench can tackle.

The abstraction is the "wrench". (class)

Each wrench (with the specific size handled) is an instance of the class wrench.

Let us go to a more complex example:

Take the "human being" abstraction.

Depending upon the situation, we may be interested upon different ^{aspects} of "humanness":

- (1) Physiological characs (Biology)
- (2) Demographical characs (Sociology)
- (3) Existential characs
(such as SSN, address etc) (General)
- (4) Occupational characs (Categorical)
(if students: year, major etc
if employed: dept, years on serv. etc)

Classes encapsulate both the pertinent characs of the entity and the functions performed on these variables, relevant to the program's focus.

For instance, the student class:

Variables of interest could be: 0. Name !!

1. Year/Semester
2. Major/School
3. GPA (current)
4. Current course registration

Functions on these variables could be:

1. Initialization/Modification of variables.
 2. Computation of GPA
- etc.

Each student such as

John Smith

Jane Doe

~~are~~ ^{is} said to be an instance of the class student.

What sets John Smith apart from Jane Doe are the specific values assigned to the class variables.

Since the object instances are declared as variables of type student, classes are higher level data types themselves.

All data structures learnt in this class will be implemented as classes.

Data structures refers to the organization of the (primitive) ^{lower level.} data types

For instance,

- arrays organize similar data in contiguous linear memory segments.
- lists (linked) organize similar data in random memory locations.
- trees organize similar data in ^{semi-}two-dimensional structures dictated by tree types.
- graphs organize similar data in two-dimensional or more dimensional structures dictated by the graph types.

• "similar data" refers to the type of data.

For instance,

we can have an array or list of integers only
or floats only
or chars only

etc.

But we cannot have a array or list of integers and floats or any other data type.

An array or other data structure may consist of object instances of a class, since class itself is a type of data.

Eg. an array of students.

Abstraction of Datastructure and Implementation

Linked List

At one level:

List

Array

At a lower level:

Linked list
or
Array

class.

At a higher level:

Tree & Graph

Listsequences.

Any class design, therefore must ask 2 questions:

1. what variables characterize the class?
(as relevant to the current domain/task)
2. what functions need to be performed on these variables?

Program Design:

1. Determine goal of program.
2. Break tasks into functionally coherent modules.
3. Encapsulate modules into classes.

In OO Programming:

Every implementation of a module is as a class.

The class main is the driver of the program.