Disclaimer:
This lecture material is prepared using the book *JAVA: The Complete Reference* by *Herbert Schildt*

Chapter 1 Introduction to Programming Language

Programming language

A programming language is a language that is designed to be used (read and written) by humans to create programs that can be executed by computers. In other words we can say that programming languages provides the way so that the users may interact with the computer to give it commands and instructions to perform certain tasks. There are two main types of computer programming languages.

- **Low-level languages**
- **High-level languages**

Low Level Programming Languages

These languages are near to computer hardware and far from human languages. Computer can understand these languages easily. Following are two low-level languages:

- Machine Language
- Assembly Language

Machine Language

A computer language in which instructions are written in binary form (0 and 1) is called machine language. It is the only language that is directly understood by the computer. Machine language is the native language of computer. Machine language is also known as first generation language.

Advantages of Machine Languages

- Very fast program execution: Because the machine language is the native language of computer that computers directly understand and execute. There is no need of a translator program, because computer can already understand and execute the machine language instructions directly without the need of translation.

Disadvantages of Machine Languages

- Machine Language is difficult to understand
- Machine Language is difficult to learn
- Programs of Machine Language are difficult to modify

- Machine Language requires deep knowledge of hardware
- Programs of Machine Language are difficult to remove errors
- Programs of Machine Language are Machine dependent

Assembly Language

Assembly language is a low-level language. In assembly language, symbols are used instead of binary code. These symbols are easy to remember. For example Add instruction is used to add two numbers. Assembly language is also known as second generation language

Advantages of Assembly Language

- Assembly language programs are executed with fast speed
- Assembly language programming is easier to learn, understand and modify than machine language

Disadvantages of Assembly Language

- Assembly language programs are machine dependent
- Assembly language programming requires deep knowledge of hardware

High Level Programming Languages

A type of language that is close to human languages is called high level language. High-level languages are easy to understand. Instructions of these languages are written in English-like words e.g. Print, Display, Write etc.

Examples of High Level Programming Languages

- COBOL
- BASIC
- PASCAL
- C Programming Language
- C++
- JAVA
- Visual Basic

Advantages of High Level Programming Languages

- High Level Programming Languages are Easy to learn and understand
- Programs written in High Level Programming Languages are Easy to modify

- It is Easy to remove errors in the Programs written in High Level Programming Languages
- Programs written in High Level Programming Languages are Machine independent
- High Level Programming Languages have Better documentation

Disadvantages of High Level Programming Languages

- A disadvantage of High Level Programming Languages is slower program execution.
- High Level Programming Languages provide programming facilities for performing certain operations.

Few examples of High Level Programming Languages

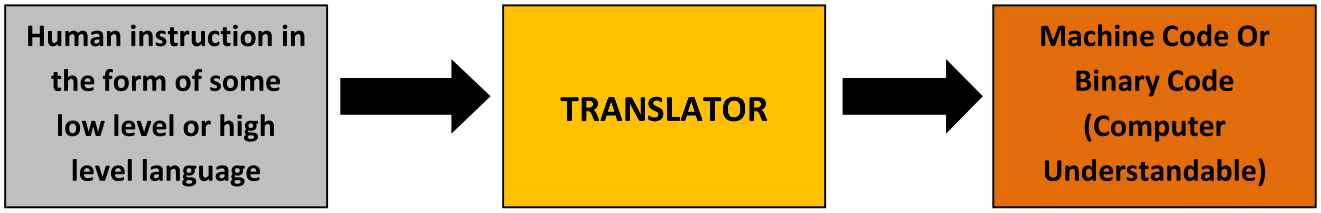
- **FORTRAN:** FORTRAN stands for **FOR**mula **TRAN**slation. It is mainly used for writing applications related to science and engineering fields.
- **BASIC:** BASIC stands for Beginners All-purpose Symbolic Instruction Code. It was invented in late 1960's to teach programming skills to students.
- **COBOL:** COBOL stands for Common Business Oriented Language. It was especially designed for business applications.
- **C Language:** C Language is one of the most popular programming languages among students and beginners in the field of computer programming. C Language was designed by Dennis Ritchie at AT&T Bell Labs in 1972. Sometimes C Language is called a middle level language because it provides facilities to develop application software as well as system software. So C Language combines some important qualities of a high level language and a low level programming language.

Translators

Computers only understand machine code (binary). But on the other hand, programmers prefer to use a variety of high and low-level programming languages. To get rid of this issue, the high-level and low-level program code (source code) needs to pass through a translator. A translator converts the source code into machine code (object code). Different types of translators are:

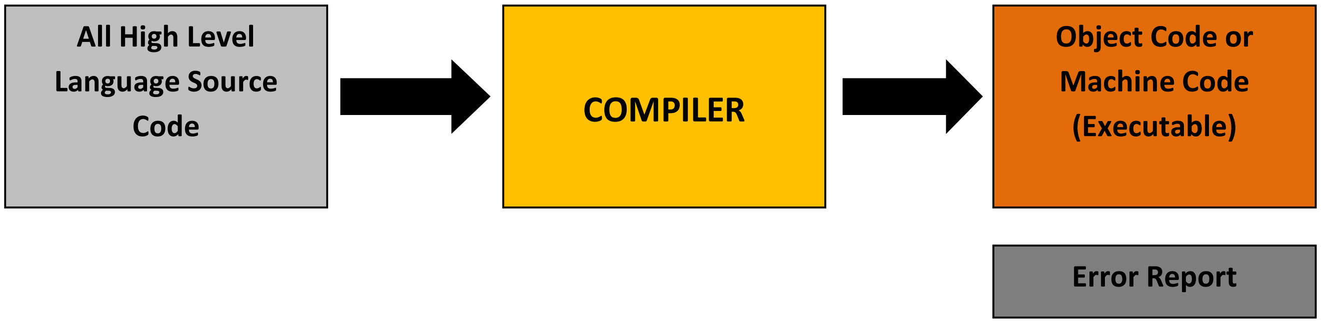
- Compiler
- Interpreter
- Assembler

- These errors can only be fixed by changing the original source code and compiling the program again.


Human instruction in the form of some low level or high level language
TRANSLATOR
Machine Code Or Binary Code (Computer Understandable)

Compiler

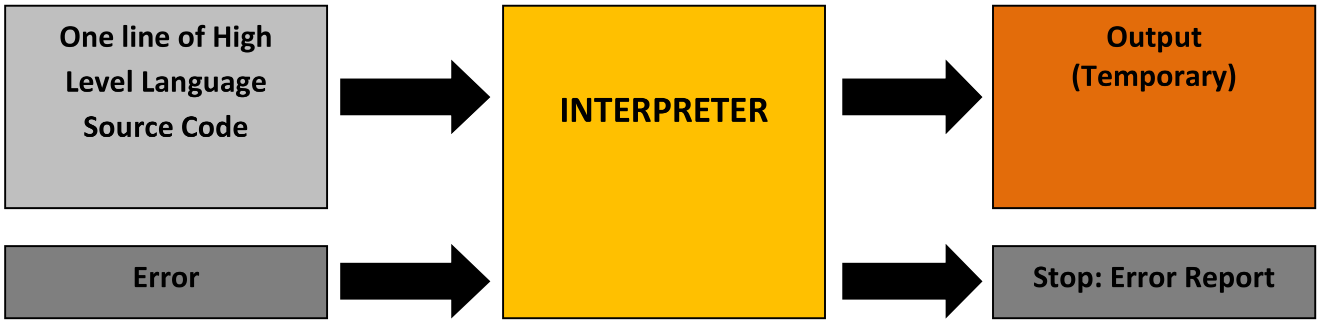
- It is a translator which is used to translate program written in a high-level language into machine code (object code). A compiler takes the whole program and translates it into corresponding machine code which can be executed thereafter.
- The process of compilation may take some time but the translated program can be used again and again without the need for recompilation.
- An error report is often produced after the full program has been translated. Errors in the program code may cause a computer to crash.
- These errors can only be fixed by changing the original source code and compiling the program again.


All High Level Language Source Code
COMPILER
Object Code or Machine Code (Executable)
Error Report

Interpreter

- Interpreter is a translator which is able to read, translate and execute one statement at a time from a high-level language program.
- The interpreter stops when a line of code is reached that contains an error. Interpreters are often used during the development of a program.

- They make debugging easier as each line of code is analyzed and checked before execution. Interpreted programs will launch immediately, but our program may run slower than a complied file.
- No executable file is produced.
- The program is interpreted again from scratch every time you launch it.



Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Assembler

Assemblers are used to translate a program written in a low-level assembly language into a machine code (object code) file so it can be used and executed by the computer. Once assembled, the program file can be used again and again without re-assembly.

Difference between Low-level & High-level Language

High-level Language	Low-level Language
High-level languages are easy to learn	Low-level languages are difficult to learn.
Near to human languages.	Far from human languages.
Programs in high-level languages are slow in execution.	Programs in low-level languages are fast in execution.
Programs in high-level languages are easy to modify.	Programs in low-level languages are difficult to modify.
Deep knowledge of hardware is not required to write programs.	Deep knowledge of hardware is required to write programs.

High-level programming languages

The high-level programming languages can be categorized into different types on the basis of the application area in which they are employed as well as the different design paradigms supported by them. The high-level programming languages are designed for use in a number of areas. Each high-level language is designed by keeping its target application area in mind. Some of the high-level languages are best suited for business domains, while others are apt in the scientific domain only. The high-level language can be categorized on the basis of the various programming paradigms approved by them. The programming paradigms refer to the approach employed by the programming language for solving the different types of problem.

Categorization based on Application

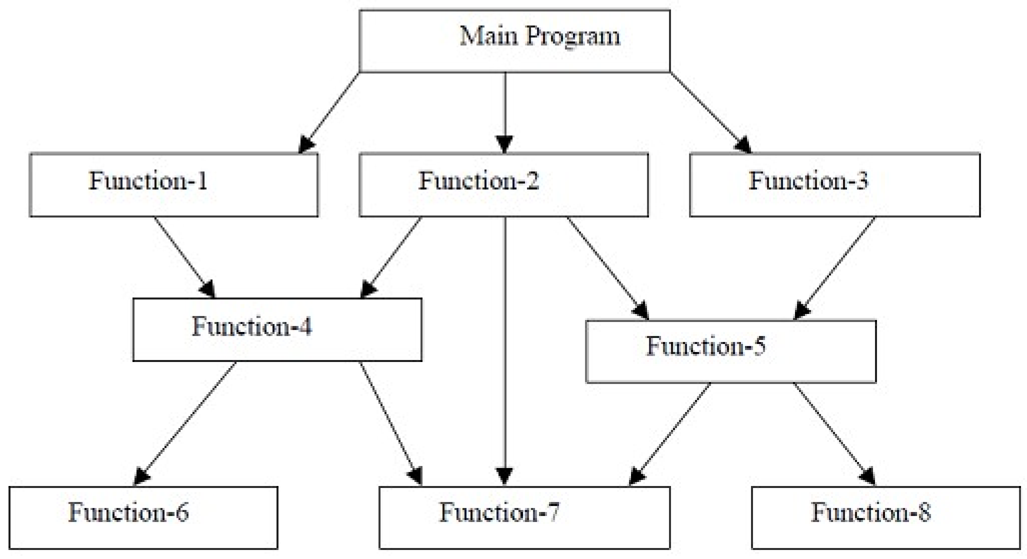
On the basis of application area the high level language can be divided into the following types:

- **Commercial languages:** These programming languages are dedicated to the commercial domain and are specially designed for solving business-related problems. These languages can be used in organization for processing handling the data related to payroll, accounts payable and tax building applications. COBOL is the best example of the commercial based high-level programming language employed in the business domain.
- **Scientific languages:** These programming languages are dedicated to the scientific domain and are specially designed for solving different scientific and mathematical problems. These languages can be used to develop programs for performing complex calculation during scientific research. FORTRAN is the best example of scientific based language.
- **Special purpose languages:** These programming languages are specially designed for performing some dedicated functions. For example, SQL is a high-level language specially designed to interact with the database programs only. Therefore we can say that the special purpose high-level language is designed to support a particular domain area only.
- **General purpose languages:** These programming languages are used for developing different types of software application regardless of their application area. The various examples of general purpose high-level programming languages are BASIC, C, C++, and java.

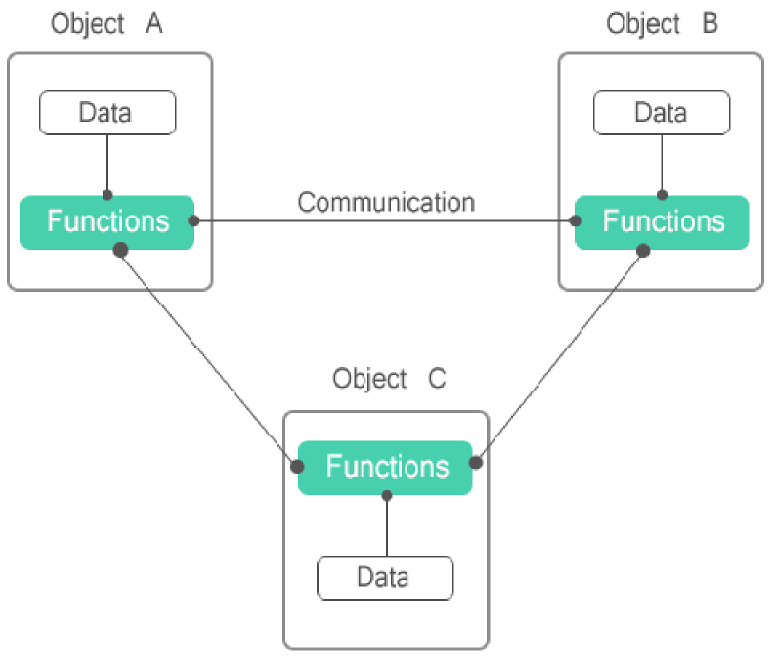
Categorization based on Design paradigm

On the basis of design paradigms the high level programming languages can be categorised into the following types:

- **Procedure or function oriented languages:** These programming languages are also called an imperative programming language. In this language, a program is written as a sequence of procedures. Each procedure contains a series of instruction for performing a specific task. Each procedure can be called by the other procedures during the program execution. In this type of programming paradigms, a code once written in the form of a procedure can be used any number of times in the program by only specifying the corresponding procedure name. Therefore the procedure-oriented language allows the data to move freely around the system. The various examples of procedure-oriented language are FORTRAN, ALGOL, C, BASIC, and ADA.



- **Logic-oriented languages:** These languages use logic programming paradigms as the design approach for solving various computational problems. In this programming paradigm, predicate logic is used to describe the nature of a problem by defining the relationship between rules and facts. *Prolog* is the best example of the logic-oriented programming language.
- **Object-oriented languages:** These languages use object-oriented programming paradigms as the design approach for solving a given problem. In this programming language, a problem is divided into a number of objects which can interact by passing messages to each other. C++, JAVA etc are the examples of object-oriented programming language.



Advantages of Object Oriented Programming

Simplicity: software objects model real world objects, so the complexity is reduced and the program structure is very clear.

Modularity: each object forms a separate entity whose internal workings are decoupled from other parts of the system.

Modifiability: it is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.

Extensibility: adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.

Maintainability: objects can be maintained separately, making locating and fixing problems easier.

Re-usability: objects can be reused in different programs.

Features of Object Oriented Programming

Various features of object oriented programming are:

- Object
- Class

- Data encapsulation
- Data hiding
- Data abstraction
- Inheritance
- Polymorphism etc.

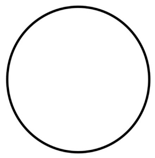
Let us briefly discuss all these features one by one.

Object: Objects are basic entities in an object oriented system. An object is a real world entity which posses certain properties (also called attributes) and behavior (also called operation).

- Objects may represent a person, place, flower, a bank account, a table of data or any other item that a program has to handle.
- In a program, properties of an object are represented using variables which are also known as data (or data members) and behaviors are represented using functions which are also known as member functions.
- Objects are also known as instance of a class.

Few Examples of object are:

Student **Book** **Mango** **Rose** **Circle**



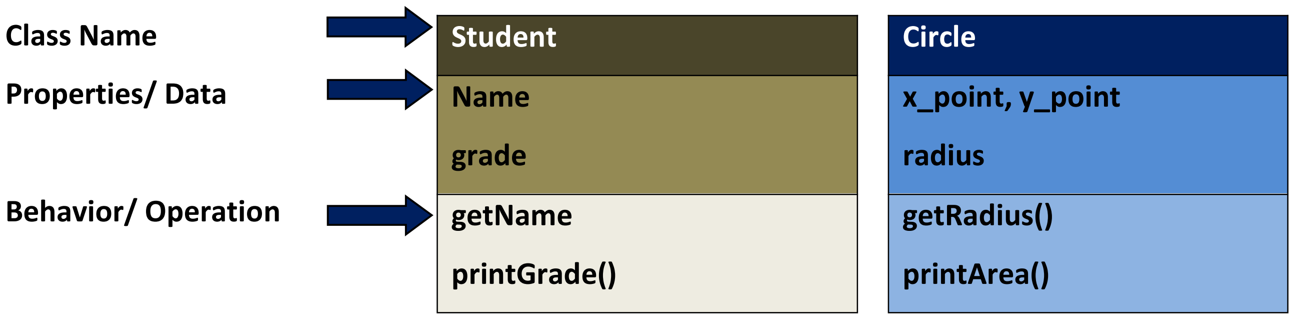
A student object has properties like name, roll no, DOB, email etc. A book object has properties like title, author name, pages price etc. Similarly a circle object has properties like radius, coordinates of center etc.

Class: Collection of similar objects which possesses common properties (also called attributes) and behavior (also called operation) is known as class. It is a **template** or **blueprint** from which objects are created.

Few examples of class:

Student **Book** **Fruit**

Data Encapsulation: The wrapping of data members and member functions into a single unit is known as data encapsulation.



Data hiding: The insulation of data and methods of a class from direct access by other classes is known as data hiding. Essentially, access privileges members of a class is assigned to other classes with the help of access specifiers (like default, private, protected & public) and OOP principles like inheritance, polymorphism etc.

Data abstraction: The act of representing the essential features of an entity by omitting their complete details is known as data abstraction. Class supports the principle of data abstraction with the concept of abstract classes and hence class is also known as abstract data type (ADT).

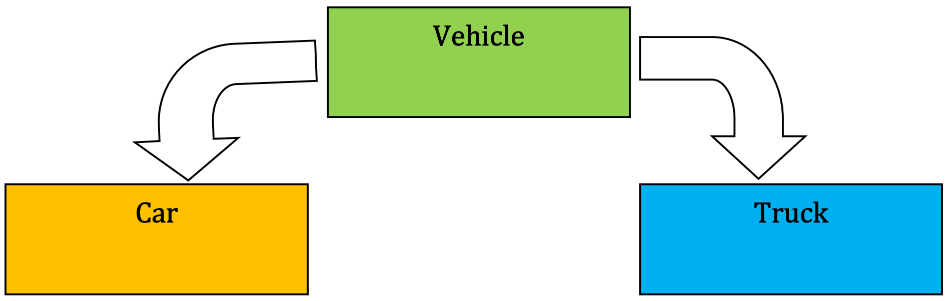
Inheritance: Inheritance is a mechanism by which objet of one class acquires the properties of object of another class. The class from which properties are taken into a new class is known as parent class or **super class**. The class which takes properties from an existing class is known as child class or **sub**

class. It is an important part of OOPs (Object Oriented programming system). The main advantages of inheritance are:

- Code Reusability
- Code Minimization
- Maintaining class Hierarchy
- Well Programming Construct etc

There are various types of inheritance (like single, hierarchical, multi level and hybrid etc) in java which are discussed later in this course. In java inheritance is achieved through the key word **extends**.

A simple example of hierarchical inheritance:



Here, class Car and Truck inherits the properties of the class vehicle. Class Vehicle is called super class and class Car and Truck are called sub classes.

Syntax of above example:

```
class Vehicle
{
        -----------------
        -----------------
}
class Car extends Vehicle
{
        -----------------
        -----------------
}
class Truck extends Vehicle
{
        -----------------
        -----------------
}
```

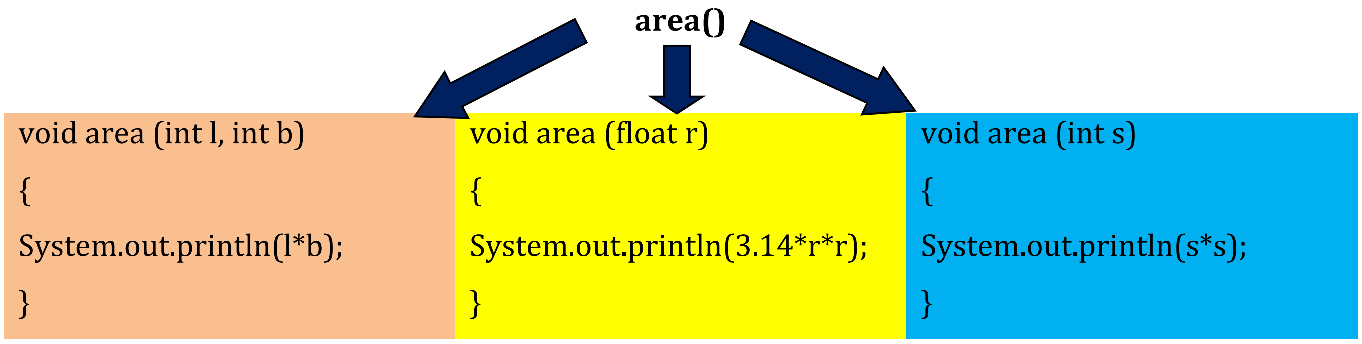
Polymorphism:

The word polymorphism means having many forms (Poly means many and morph means forms). In simple words, we can define polymorphism as the ability of an entity to exhibit multiple behaviors or definitions depending upon the working environment.

Real life example of polymorphism: A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So, the same person has different behavior in different situations.

Example of polymorphism in programming context:

Same function area can be use to find area of a rectangle, circle, square etc (this mechanism is known as *function* or *method overloading*) which is shown in the below diagram:



```
void area (int l, int b)
{
System.out.println(l*b);
}
```

```
void area (float r)
{
System.out.println(3.14*r*r);
}
```

```
void area (int s)
{
System.out.println(s*s);
}
```

Review Questions of Chapter 1

MCQ Type

1. **Choose the most appropriate answer:**

   a) **The type of language understandable by a computer is**
      a) Assembly Language b) Machine Language
      c) High level language d) JAVA language

   b) **A software which translates high level language to machine language by considering the program a whole is called**
      a) Compiler b) Interpreter
      c) Either a) or b) d) Both a) and b)

   c) **Which one of the following provides a template or blue print for similar objects?**
      a) Class b) Structure
      c) Inheritance d) Polymorphism

   d) **Which one of the following is/ are examples of low level language?**
      a) Assembly language b) Machine Language
      c) JAVA d) Both a) and b)

   e) **Which one of the following is/ are OOPs characteristics?**
      a) Inheritance b) Data hiding
      c) Encapsulation d) All of these

   f) **The property of OOPs by which an entity can acquire multiple forms is known as**
      a) Inheritance b) Polymorphism
      c) Encapsulation d) Data Aggregation

   g) **The act of representing essential feature of a class without specifying its complete details is known as**
      a) Inheritance b) Polymorphism
      c) Data Encapsulation d) Data Abstraction

   h) **The act of wrapping data and methods into a single unit is known as**
      a) Inheritance b) Polymorphism
      c) Data Encapsulation d) Data Abstraction

   i) **The technique by which object of one class acquires the properties of object of some other class is known as**
      a) Inheritance b) Polymorphism

c) Data Encapsulation d) Data Abstraction

j) What is use of interpreter?
a) They convert byte code to machine language code
b) They read high level code and execute them
c) They are intermediated between JIT and JVM
d) It is a synonym for JIT

k) The technique by which object of one class acquires the properties of object of some other class is known as
a) Inheritance b) Polymorphism
c) Data Encapsulation d) Data Abstraction

Short Type

2. Answer briefly

a) Define translator. Differentiate between a compiler and an interpreter.
b) Differentiate between high level and low level programming language.
c) Define data encapsulation.
d) Define inheritance. List different types of inheritance.
e) What is polymorphism?
f) What is data abstraction?
g) List feature of object oriented programming system.
h) List advantages & disadvantages of high level languages.
i) List advantages & disadvantages of low level language.
j) Why C language is sometimes called middle level language?

Long Type

3. Answer in details

a) Discuss the characteristics of Object oriented Programming System.
b) Explain the advantages of Object Oriented Programming System.
c) Explain different types of programming languages with their advantages & disadvantages in details.

****