Chap 5

~~Chapter-5~~

1) Define OOP and write about its principles.
→ In OOP, we create object involved in the problem. We then solve the problem by message passing between objects.
→ It focus on objects rather than functions
→ It follows bottom up approach in program design.

The principles of OOP are:-

i) Object
= It is a basic unit of OOP. It may represent a person, place, bank, etc.
→ When program is executed, the objects interact by sending messages to one another.
→ each object contain data & code to manipulate data.
→ Object can interact without having to know details of each other's data or code.

ii) Classes
→ It is a collection of objects of similar types.
→ It is a blueprint or set of instructions to build a specific type of object.
→ many objects can be created from same class.
→ It is a user defined data type which consists of dat and function of objects.

iii) Encapsulation

→ The wrapping of data & function in a single unit i.e class is called encapsulation.

→ means that the internal representation of an object is generally hidden from the outside world.

→ The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.

→ It causes data hiding.

iv) Abstraction

→ It refers to the process of showing essential features without including the background details.

v) Inheritance

→ Inheritance is the process by which object of one class (child) acquire the properties of other class (parent)

→ In OOP, it provides the idea of ~~reusab~~ reusability i.e we can add additional features to an existing class without modifying it.

→ The new child class will have combined features of both the classes i.e parent class & its own.

vi) Polymorphism

→ It means one name but having many forms.

→ it have multiple methods all with same name but slightly different functionality.

* Objects identified during design of assembler.

1) Source-program
(i) Contents
→ the program to be translated.

(ii) Methods
a) Assemble
→ Translate the source program, producing an object program and an assembly listing.

2) Source-line
(i) Contents
→ a line of the source program

(ii) Methods
a) Create
→ create and initialize new instance of source-line.
b) Assign_location
→ assign a location counter value to the line.
c) Translate
→ translate the instruction or data definition on the line into machine language.
d) Record-error
→ record an error detected for the line.

3) Symbol_table

(i) Contents

→ labels defined in ~~each~~ the source program, with the location counter value assigned to each.

(ii) Methods

a) Enter

→ enter a label and location counter value into the table.

b) Search

→ search the table for a specified label

4) Opcode_table

i) Contents

→ mnemonics instructions to be recognized by the assembler.

ii) Methods

a) Search

→ search the table for a specified mnemonic instruction.

5) Object_Program

i) Contents

→ the object program resulting from the assembly

ii) Methods

a) Enter_text

→ enter the machine language translation of an instruction into the object program

b) Complete
→ enter the program length and complete the generation of the external object program file.

6) Assembly-listing
i) Contents
→ a listing of the lines of the source program and the corresponding machine language translation for each line.

ii) Methods
a) Enter-line
→ enter a source line and the corresponding machine language translation.
b) Complete
→ complete the generation of the external assembly listing file.