chapter 6:- Object oriented System Design.

→ Focus on the objects handled by the system, rather than algorithms.

→ programs are designed and implemented as collection of objects not as collection of procedures.

principles of object programming.

ⓐ Object

- is a basic unit of oop.
- is a component of a program that knows how to perform certain ~~certain~~ actions and how to ~~perform certain actions~~ interact with other elements of the program.
- Contains some data and defines a set of operations on that data that can be invoked by other parts of program.

Eg:- Consider symbol-table as an object used by assembler.
Here, set of operation or methods are like
invert_symbol and lookup_symbol
its data would be contents of hash table used to store symbols and their addresses.

ⓑ Class

- is a blueprint or template or set of instructions to build a specific type of object.
- defines the instance variables and methods of an object
- an instance is a specific object from specific class.
- many objects can be created from same class.

eg:- for an assembler to translate programs for different versio of machine, class could be opcode_table.
from this class, object could be created to define instruction set for machine.

ⓒ Encapsulation

Date |
Page |

- means that the interval representation of an object is generally hidden from view outside of objects definition.
- is the hiding of data implementation by restricting access to accessors and mutators.

③ Abstraction
- is a model, a view or some other focused representation for an actual item.
- is the implementation of an object that contains same essential properties and actions we can find in the original object we are representing.

④ Inheritance
- is a way to reuse code existing objects or to establish a subtype from an existing object.
- the relationship of classes through inheritance gives rise to a hierarchy.

Subclass :- is a modular, derivative class that inherits one or more properties from another class.

Superclass :- establishes a common interface and foundation functionality, which specialized subclass can inherit modify and supplement.

⑤ Polymorphism
- means one name, many forms
- manifests itself having multiple methods all with some name, but slightly different functionality.

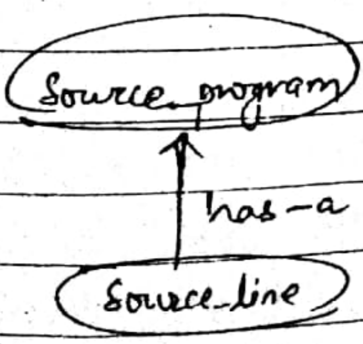


fig:- has-a relationship

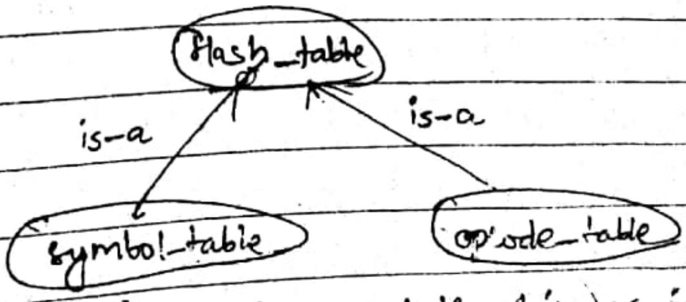


fig 2:- is-a relationship \or inheritance.

Date |
Page |

→ Hash_table is base class
→ other two are subclasses.
→ if insert_item and search_item are methods of base class then other subclasses automatically contains definition of methods.

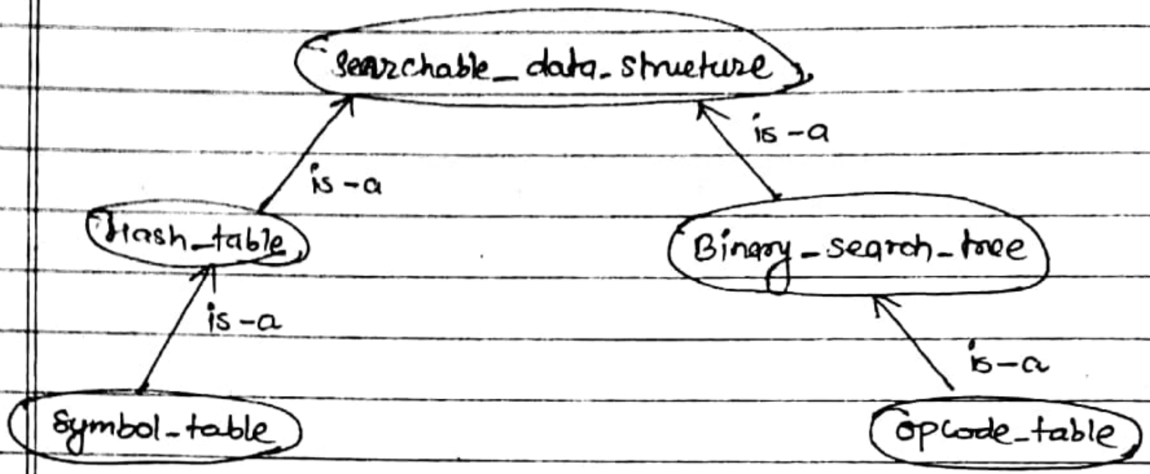


Fig 3 polymorphism.

→ here,
superclass Searchable_data_structure defines two methods insert_item and search_for_item
→ Hash_table and Binary_search_tree are subclasses, so inherits ~~and~~ above methods.
→ implementation of the methods are different ~~development~~ ~~processes (i) micro (ii) macro~~
→ ~~Booch's macro~~ ~~process~~ ~~represents overall activities of development~~ ~~on a~~ ~~long~~
in these subclasses. But ~~more~~ names of methods and why of invokation are same.
→ if Search_for_item method is invoked as instance of symbol_table, it will result in retrival from hash table
→ if same method is invoked on an instance of opcode table it will result in binary_search_tree.
→ This shows polymorphism.

<u>Object</u> oriented design of an assembles

→ according to Booch, two different development processes (i) micro (ii) macro

→ Booch's macro process represents overall activities of development on a long range scale

i) Establish the requirement for the s/w (conceptualization)

ii) Develop an overall model of system behaviour (analysis

iii) create an architecture for the implementation (design)

iv) Develop the implementation through successive refinements. (evolution)

v) Manage the continued evolution of a delivered syst (maintenance)

→ this macro process repeats itself after each release of

→ similar to waterfall model.

→ Booch's Micro process ~~elements~~ represents daily activities of system developen

i) ~~Ded~~ Identify the class and objects of system.

ii) Establish the behaviour and other attributes of the classes and objects.

iii) analyze the relationship among the classes and obj

iv) Specify the implementation of classes and objects.

→ These activities may be repeated as needed wi increasing level of details.

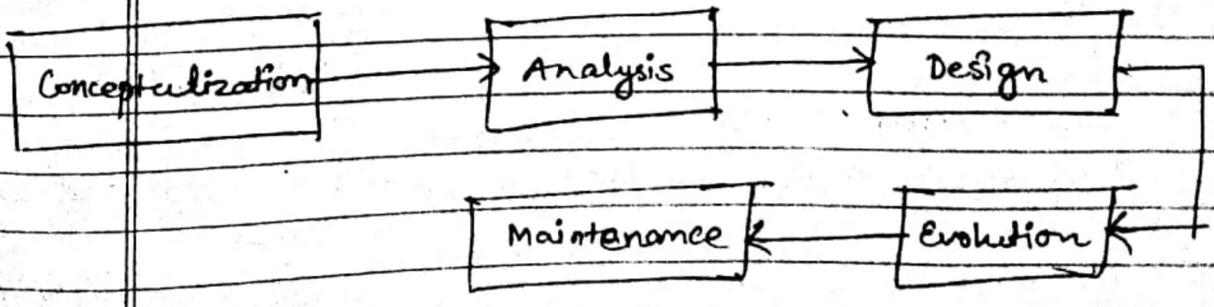
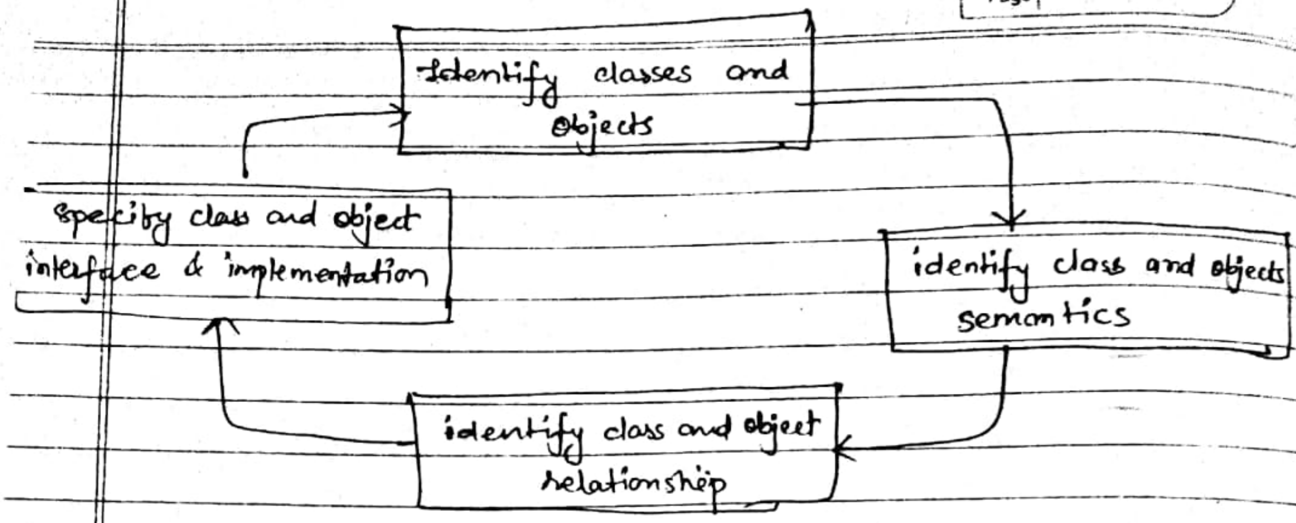


Fig:- Booch's Macro process.


Identify classes and objects
identify class and objects semantics
identify class and object relationship
specify class and object interface & implementation

Fig! – Booch's Micro process.

* Objects identified during design of assembler.

1) Source_program

i) Contents

- program
- current location counter values, errors.
- one object of class source_line for each line of program.

ii) Methods

ⓐ Assemble

- translate source program, produce an object program, and an assembly listing

2. Source_line

(i) Contents.

- line of source program
- location counter value, error

(ii) Methods

ⓐ create ~~and~~

- Create and intialize new instance of source-line

ⓑ Assign_location

- assign location counter value to line
- return updated location counter value.
- enter label on line (if any) in symbol table.

ⓒ Translate

- translate the instruction or

Date |
Page |

data definition on the line into machin language.
- make entries in object program & assembly listing.

(2) Record - errors
- record error detected.

3. Symbol - table

(i) Contents
- labels defined in src program with its location center value.

(ii) methods

(a) Error
- enter a label and location counter value into table.
- return error if label is already defined.

(b) Search
- Search table for specified label.
- return location counter value of label or error if label is not defined.

4. opcode - table

(i) Contents
- mnemonic instruction
- includes machine instruction format and code.

(ii) Methods

(a) Search :-
- Search table for specified mnemonic instruction
- return information about instruction format and operand required.

Akash Shrestha

- return error if mnemonic instruction not defined.

5. Object_program

i) Content

- object program after assembly.
- includes machine language translation of instruction and data from object program.
- includes program length.

ii) Methods

ⓐ enter_text

- enter machine language translation of an instruction or dat def^n into object program.

ⓑ Complete.

- enter program length and complete generation of external o program file.

6. Assembly_listing

i) Contents

- listing of lines of source program and corresponding machin language translation.
- includes errors for each line & summary of errors in progr

(ii) methods

ⓐ Enter_line

- Enter source_line, the corresponding machine language translation and description of errors detected for the line into assembly listing.

ⓑ Complete.

- Enter summary of errors detected and complete the generat of external assembly listing file.

* <u>Object program</u>

- indicates the methods that are invoked by each object.
- eg:- source_program object invokes method create, Assign and translate on source_line object.

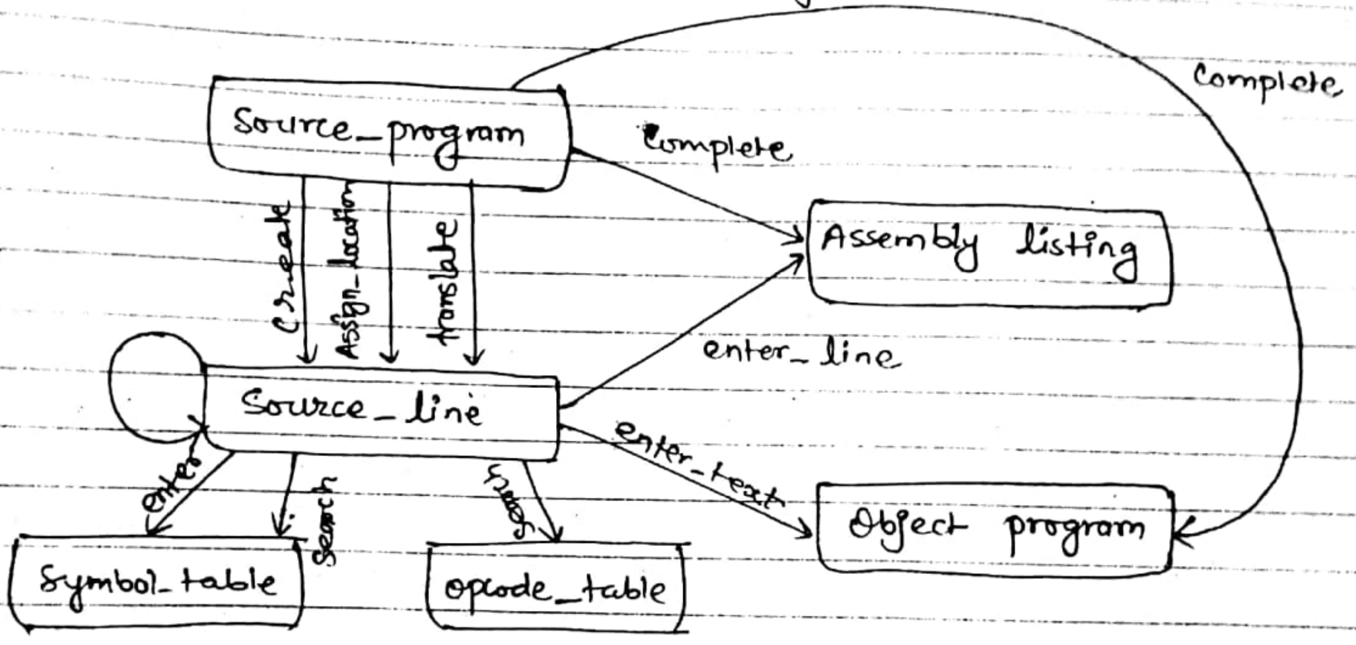


Fig:- Object diagram of assembler.

→ Object diagram ~~many~~ may also indicate the class of object.

* Interaction Diagram.

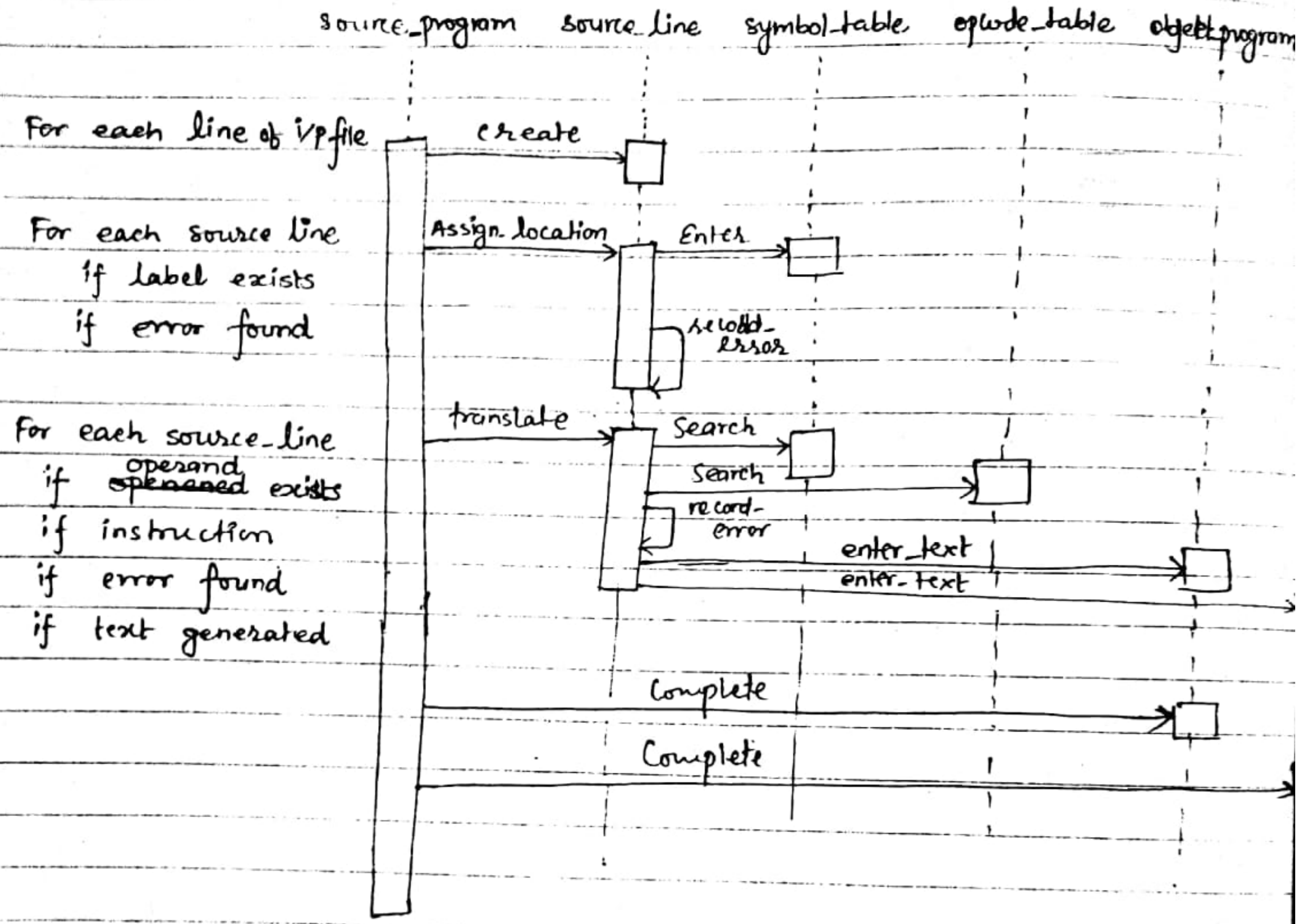


Fig:- Interaction Diagram.

→ interaction diagram makes easy to visualize the sequence of obj invocation and flow of control between objects.

→ each object is represented by dashed vertical line.

→ invocation of method is shown by horizontal line between objects.

→ the sequence is indicated by their vertical position in diagram.

→ a script is often written at L.H.S of diagram to describe condition and iteration.

→ a narrow vertical box can be used to indicate the time flow of control is focused in each object.