CHAPTER 5

GENERAL OOP CONCEPTS

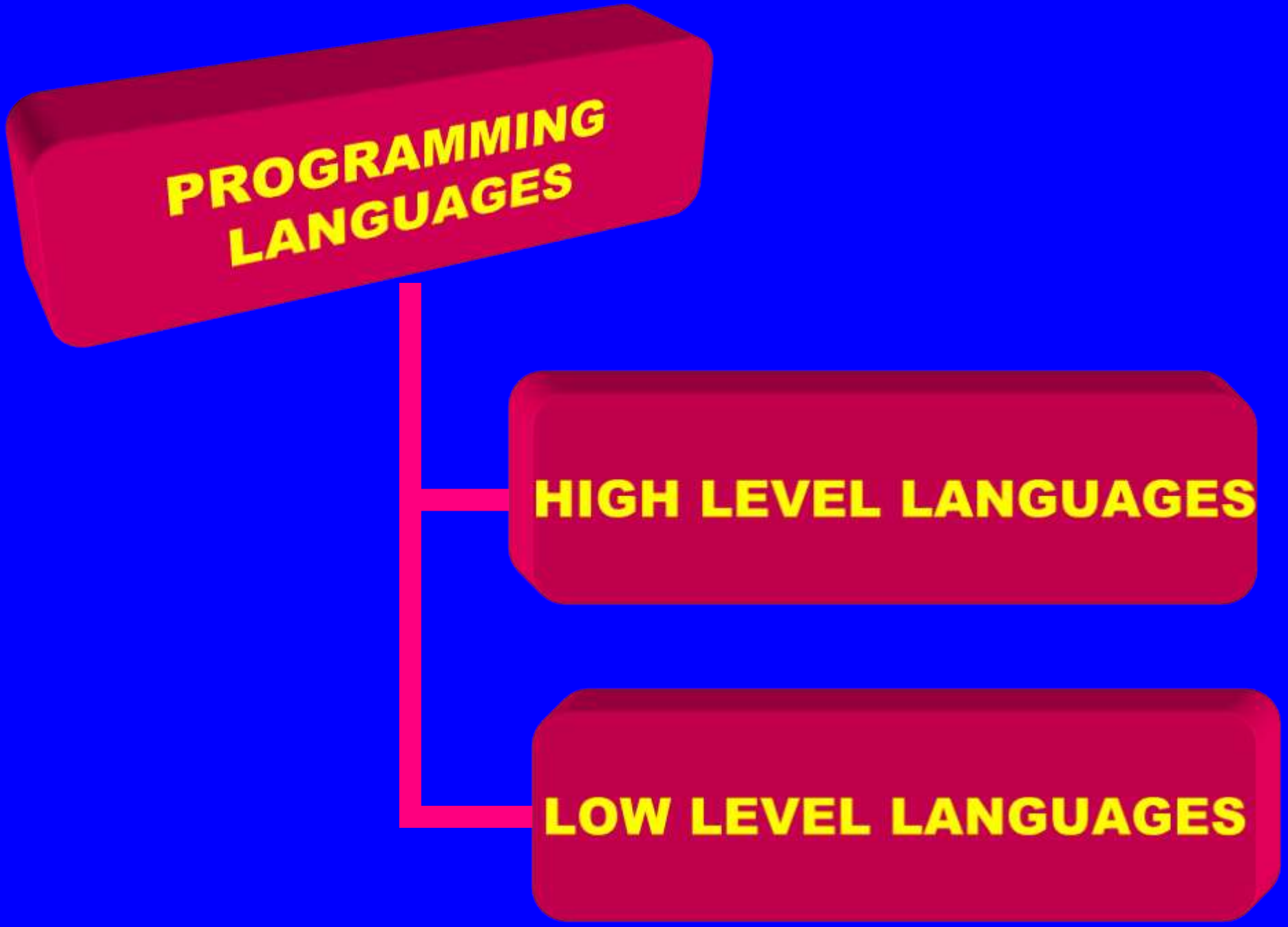
EVOLUTION OF SOFTWARE

A PROGRAMMING LANGUAGE SHOULD SERVE 2 RELATED PURPOSES :

1. It should provide a vehicle for programmer to specify actions to be executed.
2. It should provide a set of concepts for the programmer to use when thinking about what can be done.

A PROGRAMMING LANGUAGE SHOULD SERVE 2 RELATED PURPOSES :

1. The first aspect ideally requires a language that is “close to the machine”.
2. The second aspect ideally requires a language that is “close to the problem to be solved”


PROGRAMMING LANGUAGES
HIGH LEVEL LANGUAGES
LOW LEVEL LANGUAGES

LOW LEVEL LANGUAGES

- ✓ ARE MACHINE ORIENTED
- ✓ Require Extensive Knowledge of computer circuitry
- ✓ Ex : Machine language & Assembly language
- ✓ Serve only 1st aspect “close to machine”

LOW LEVEL LANGUAGES

- ✓ Machine Language :: instructions are written in binary code in it . Only language that computer can execute directly.

- ✓ Assembly Language :: instructions r written using symbolic names for machine operation {READ,ADD,STORE} & operands. Program is then converted into machine language using *ASSEMBLER* software.

HIGH LEVEL LANGUAGE

- ✓ OFFER English like keywords, constructs for sequence, selection & looping and use of variables & constants.
- ✓ Its programs r converted into machine language using *compiler* or *interpreter*
- ✓ Serve only 2nd aspect “close to programmer”

Languages like C & C++ serve
both the aspects
and hence, can be called as

Middle Level Languages

PROGRAMMING PARADIGMS

- ✓PARADIGM means organizing principle of a program . It is an approach to the programming.

- ✓Ex : Procedural programming
  Modular programming
  Object oriented programming

PROCEDURAL PROGRAMMING

- ✓ The focus is on *processing*
- ✓ It says :

DECIDE WHICH PROCEDURES YOU WANT
USE THE BEST ALGORITHMS YOU CAN FIND

- ✓ Languages support it by providing facilities for passing arguments to functions & returning values from functions

MODULAR PROGRAMMING

- ✓ A SET OF RELATED PROCEDURE WITH DATA THEY MANIPULATE IS CALLED A ***MODULE***.
- ✓ *It says*

DECIDE WHICH MODULES YOU WANT
PARTITION THE PROGRAM
SO THAT DATA IS HIDDEN IN MODULES

- ✓ Works on principle of grouping of components that carry out specific tasks.

Drawbacks of Procedural & Modular Approaches

- They do not model real world well. like a procedural program on library management aims at operations like *issued, returned* & give 2nd class status to real world entities : *books*.
- In modular programming, since many modules access same data, arrangement of data can't be changed without modifying all modules that access it

OBJECT ORIENTED PROGRAMMING

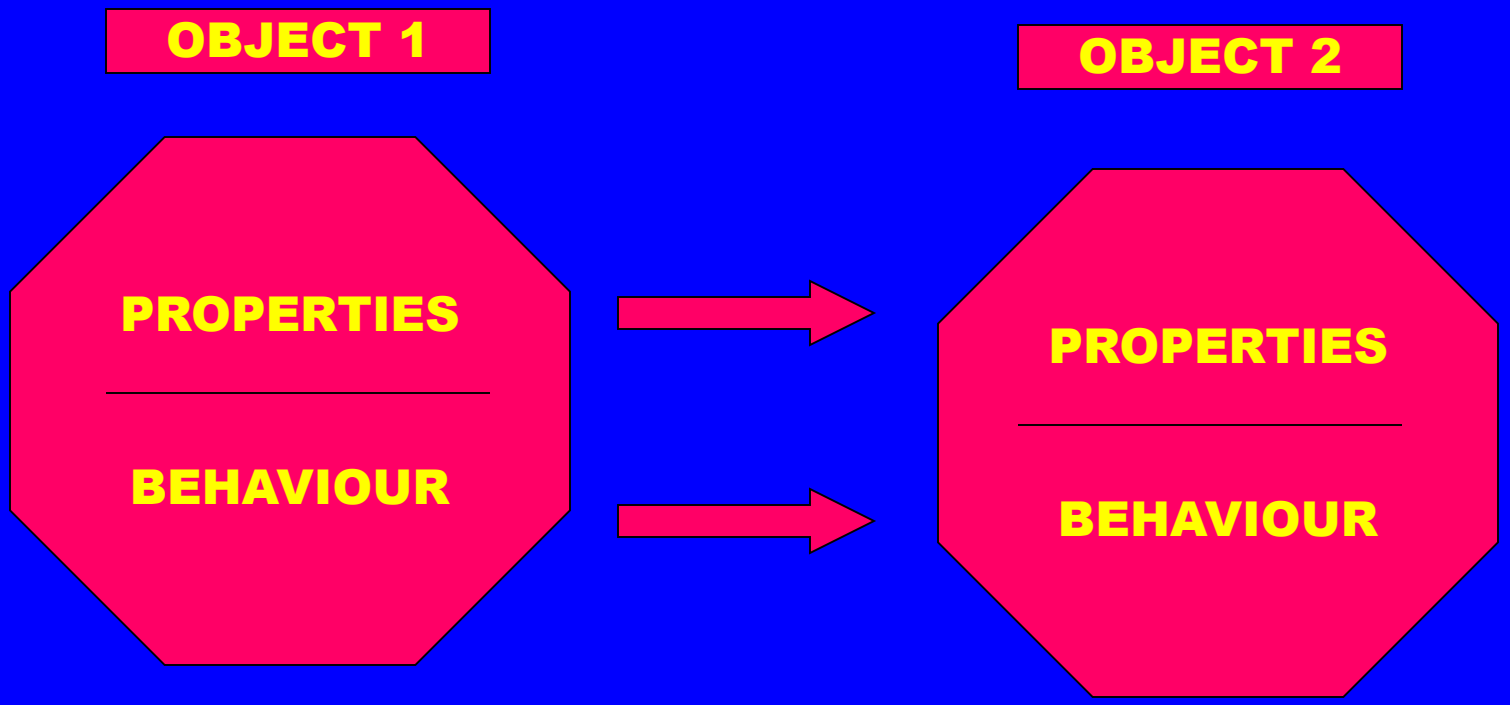
OBJECT ORIENTED PROGRAMMING

- ✓ OBJECT is an identifiable entity with some characteristics and behavior.
- ✓ In OOP programming object represents a entity that can store data and has its interface through functions.
- ✓ CLASS is a template/blue-print representing a group of objects that share common properties and relationships.

It says:

decide which classes and objects are needed

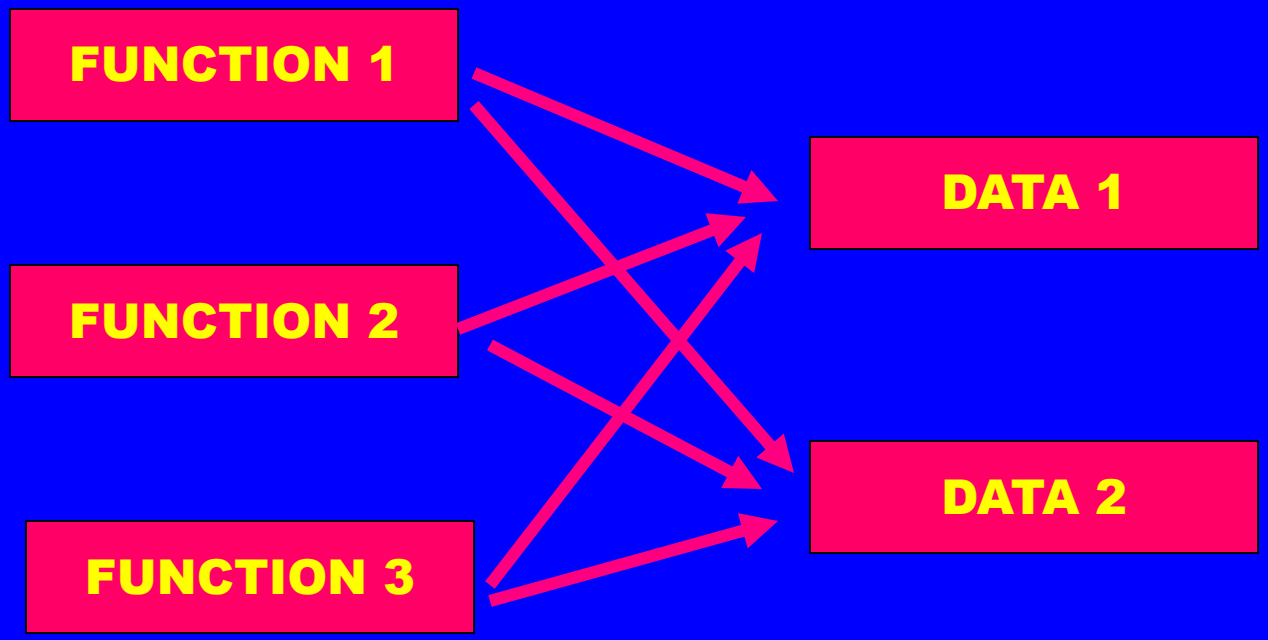
provide a full set of operations for each class

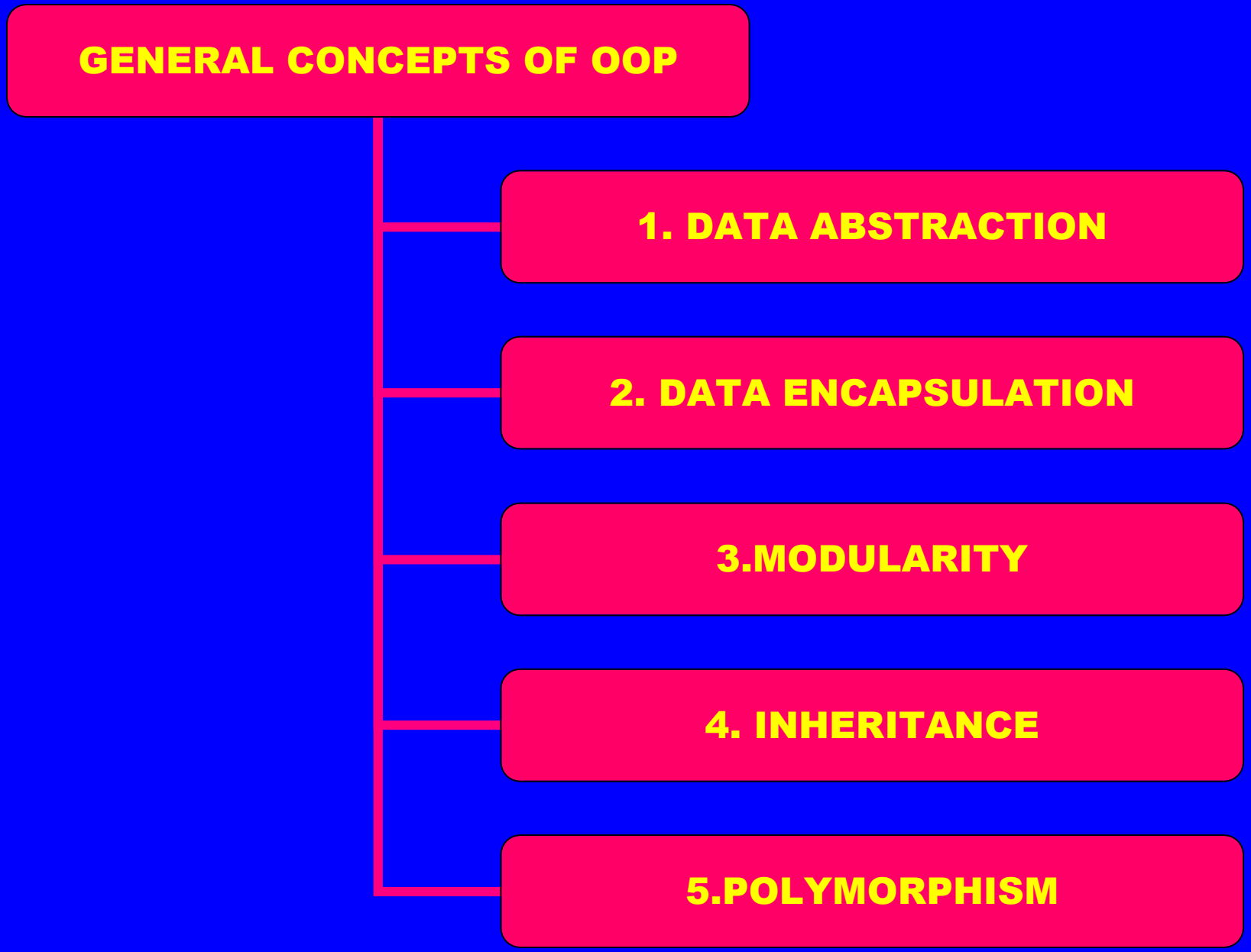


DATA & FUNCTIONS ENCLOSED WITHIN OBJECTS

NEW OBJECTS COMMUNICATE WITH EACH OTHER

PROCEDURAL PARADIGM



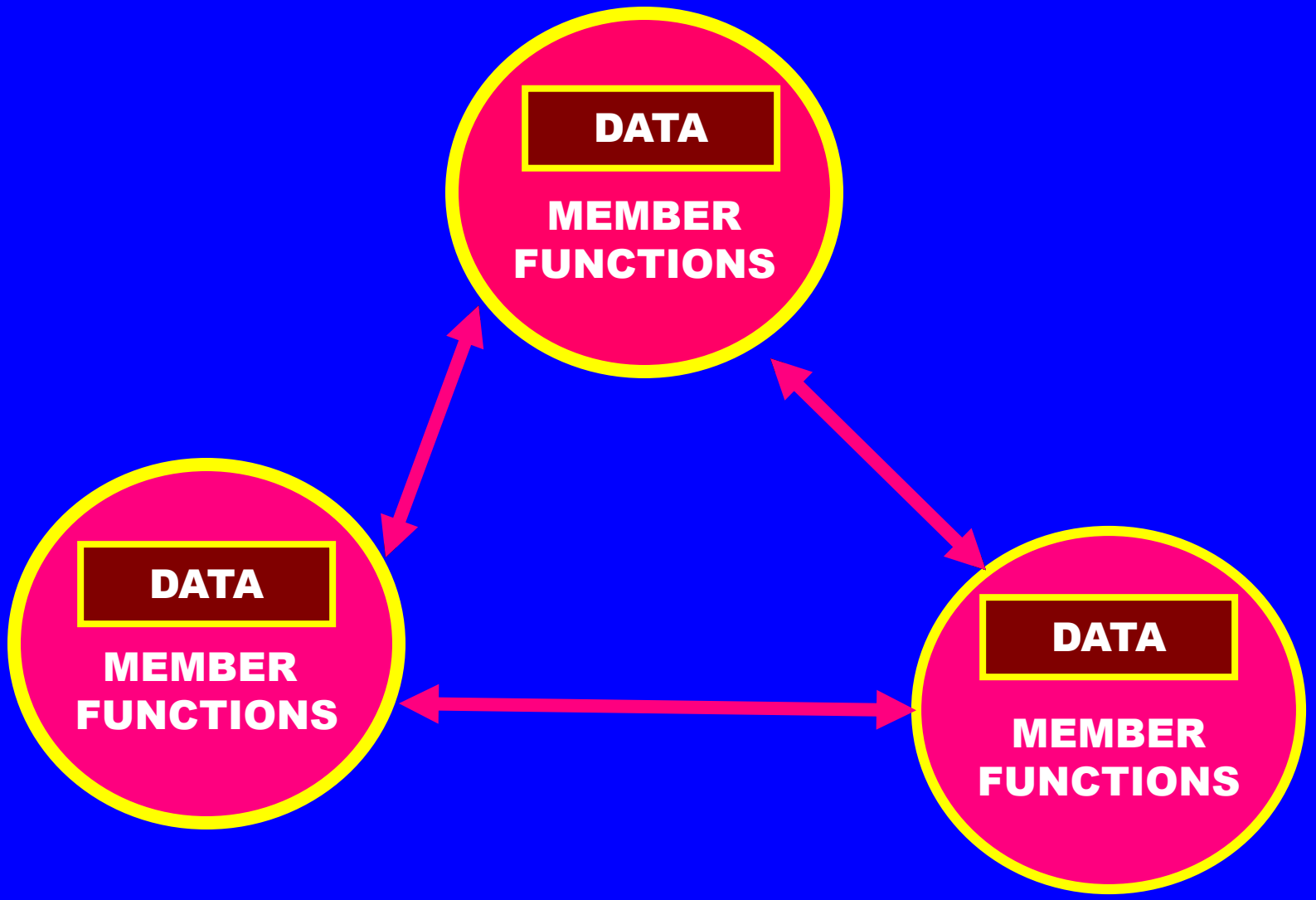

GENERAL CONCEPTS OF OOP
1. DATA ABSTRACTION
2. DATA ENCAPSULATION
3.MODULARITY
4. INHERITANCE
5.POLYMORPHISM

DATA ABSTRACTION

- ✓ ABSTRACTION refers to the act of representing essential features without including the background details or explanations.
- ✓ It focuses upon the observable behavior of an object.

ENCAPSULATION

- ✓ The wrapping up of data and operations / functions (that operate on the data) into a single unit (called *class*) is known as ENCAPSULATION.
- ✓ Encapsulation is a way to implement data abstraction. It hides the details of the implementation of an object.
- ✓ It focuses upon the implementation that gives rise to observable behavior.



THE OOP APPROACH

DATA ABSTRACTION & ENCAPSULATION

MODULARITY

- ✓ It is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.
- ✓ It reduces complexity of program
- ✓ It Creates a number of well-defined, documented boundaries within the program.

Inheritance is the capability of one class of things to inherit capabilities or properties from other class.

- ✓ Why inheritance was introduced into OO languages ?

INHERITANCE + POINTS

1. Capability to express the inheritance relationship which makes it ensure closeness with real world models.
2. Idea of reusability. It allows addition of additional features to an existing class without modifying it. A *subclass* defines only those features which are unique to it.
3. If a class A inherits properties of another class B, then all subclasses of A will automatically inherit the properties of B. This is called *transitive nature of inheritance*.

POLYMORPHISM

- ✓ It is the ability for a message or data to be processed in more than one form.
- ✓ It is a property by which the same message can be sent to objects of several different classes, & each object can respond in a different way depending on its class.

Languages that support classes but not polymorphism are called OBJECT BASED languages.

ADVANTAGES OF OOPs

1. It models real world well.
2. Programs are easy to understand.
3. It offers class reusability. Already created classes can be reused without having to write them again.
4. It facilitates quick development as parallel development of classes is possible.
5. Programs are easier to test, manage & maintain.

DIS-ADVANTAGES OF OOPs

1. Classes tend to be overly generalized.

2. The Relations Among Classes Becomes Artificial At Times.

3. OOP Programs Design Is Tricky.

4. One needs to do proper planning & proper design for OOP programming.

5. To program with OOPs, programmer needs proper skills such as design skills, programming skills, thinking in terns of objects etc.