

CE142:OBJECT ORIENTED PROGRAMMING WITH C++
December 2018 – May 2019

UNIT 1 & 2

**Introduction to Object Oriented Concept and
Design
&
Principles of Object-oriented Programming**

Content



Procedural Programming Language

Object Oriented Programming

How did OOP come into existence?

Understand OOP with real World Examples

Contd..



What is Object Oriented?



OO Methodology



OO Themes



Introduction to OO Models

Procedural Programming Language

- List of instructions to tell the computer, what to do step by step
- Procedures
- Top-Down Approach
- (Structured- Top to Bottom)

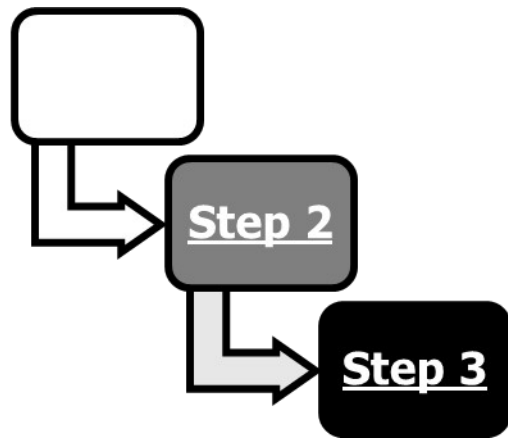
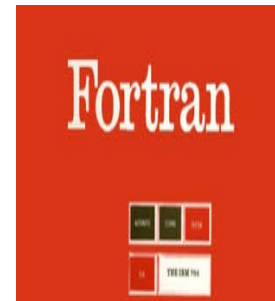


Fig:1 Top-Down Approach

Examples:



PASCAL



THE
C
PROGRAMMING
LANGUAGE



Procedural Oriented Programming Features

Emphasis on Doing Things (Algorithm).

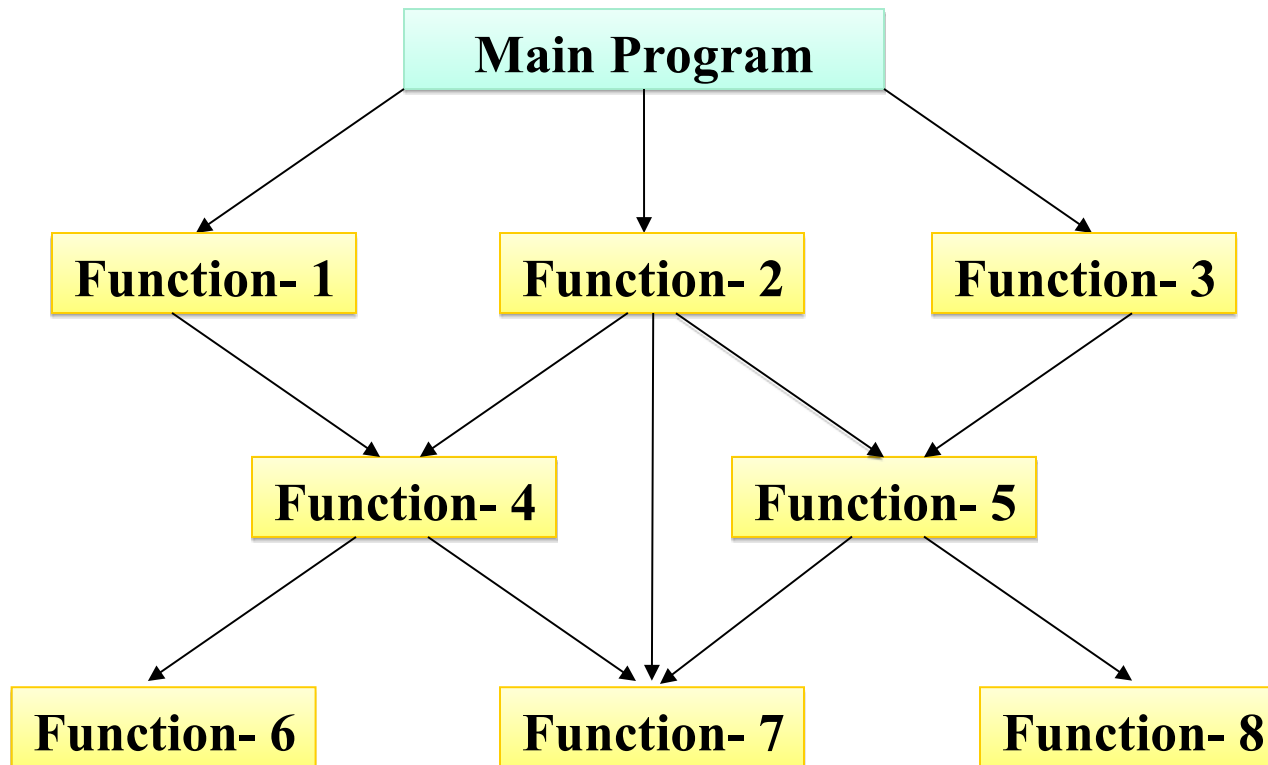
Large Programs are divided into functions.

Most of the functions are Global Data.

Data moves openly around system from function to function.

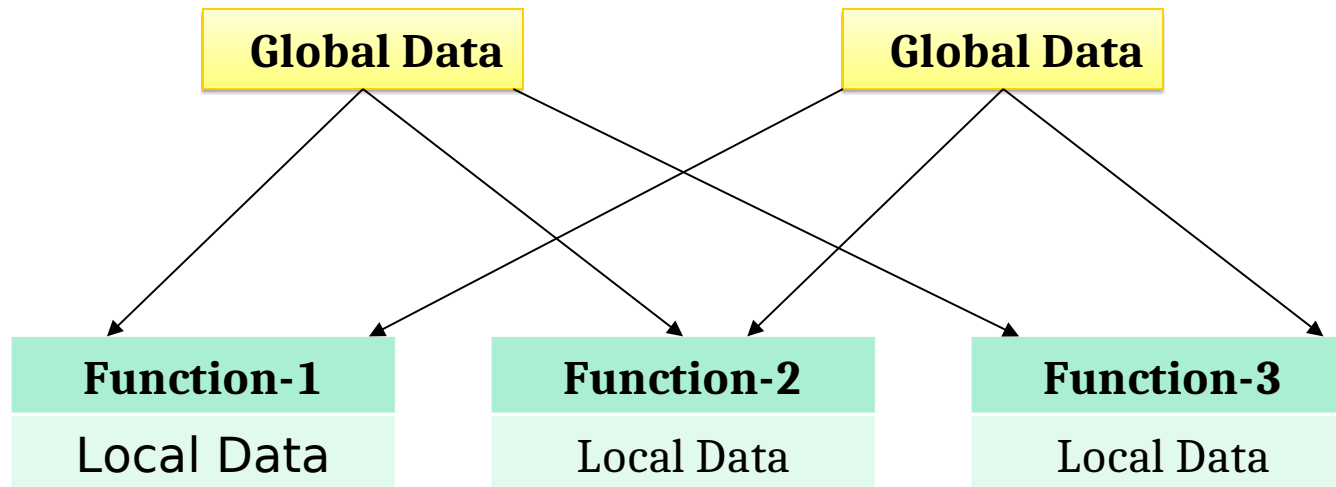
Top-Down Approach

Structure of Procedure Oriented Programming



Relationship of Data and Function in Procedure Oriented Programming.

- In multi functioned program, many important data items are placed as global so that they may be accessed by all the functions.
- Each function may have its own local data.



Relationship of data and functions in procedural programming

Drawbacks of Procedure Oriented Programming

1. The programs are made up of functions and functions are often not reusable.
2. Insecure data, if we declare a variable before main function, then it can be accessed freely from any function present in the program.
3. No better memory management.
4. No structure or code reusability. Hence time of development, testing and length of program increases.

What is Object Oriented Programming?

It is a method of Programming where code is designed and based on the functions and attributes of the objects.

Contd..

Object-Oriented (OO)

- The term Object-Oriented means that we organize software as a collection of discrete object that incorporate both data structure and behavior are only loosely connected.
- OO approach generally includes four aspects: identity, classification , inheritance and polymorphism.

History

- In the early days of object-oriented technology before the mid-1990s, there were many different competing methodologies for software development and object-oriented modeling, often tied to specific Computer Aided Software Engineering (CASE) tool vendors
- Terminology invoking "objects" and "oriented" in the modern sense of object-oriented programming made its first appearance at MIT in the late 1950s and early 1960s.
- In the environment of the artificial intelligence group, as early as 1960, "object" could refer to identified items with properties (attributes).

Objects lie at the heart of object-oriented technology

Object:

Examples:



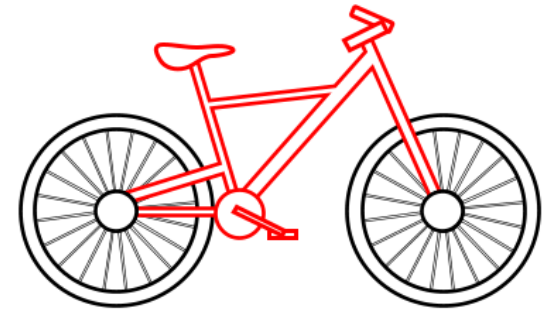
Monitor



Dog



Mango



Bicycle

Object Oriented Programming Language:

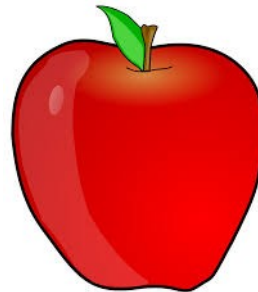
- **Object:** Entity
- **Examples:**



Table

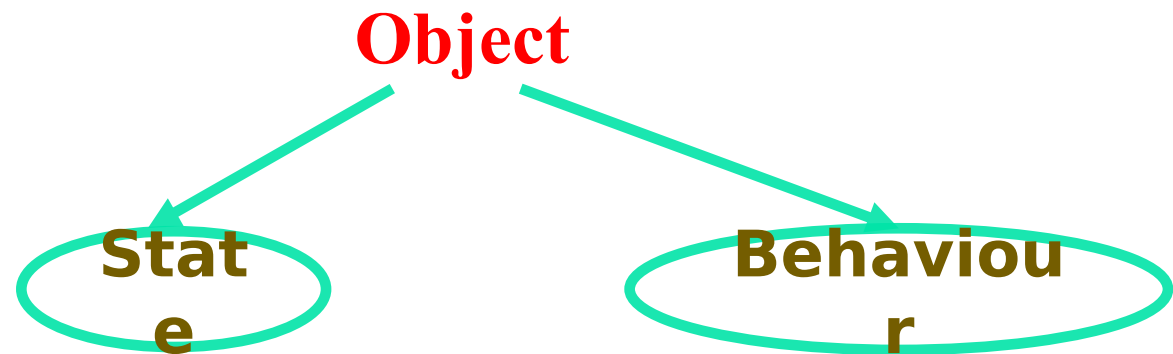


Dog



Man

Object Oriented Programming Language:



Example:

Dog

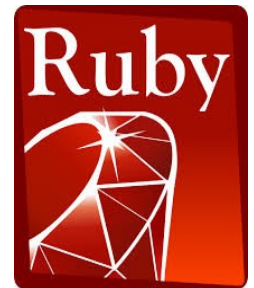
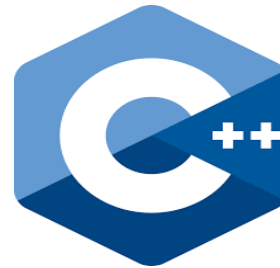


- Colour
- Name
- Height
- Eating
- Barking
- Wagging the tail

Object Oriented Programming Language:

- **Object Oriented:**
Approach to problem solving where all the problems are carried out using Objects.
- **Bottom-Up Approach**
- **Starts from basic level of programming feature i.e. class.**

Examples:



Object Oriented Programming Features

Emphasis on Data rather than Procedure.

Programs are divided into Objects.

Data is Hidden and cannot be accessed by external Function.

Object may communicate with each other through functions.

New data and Functions can be added.

Bottom-Up Approach

Programming Methodologies

- Difficult to interpret
- Difficult to enhance

NON-STRUCTURED PROGRAMMING

Basic

Fortran

Cobol

- Less efficient for enterprise level applications

STRUCTURED PROGRAMMING

C

Pascal

- Can interact with the other code
- Work on enterprise level applications

OBJECT ORIENTED PROGRAMMING

C++

Java

Example:

Bank Account Holder withdraws money from ATM.

Step 1: Insert the ATM.

Step 2: Enter the PIN.

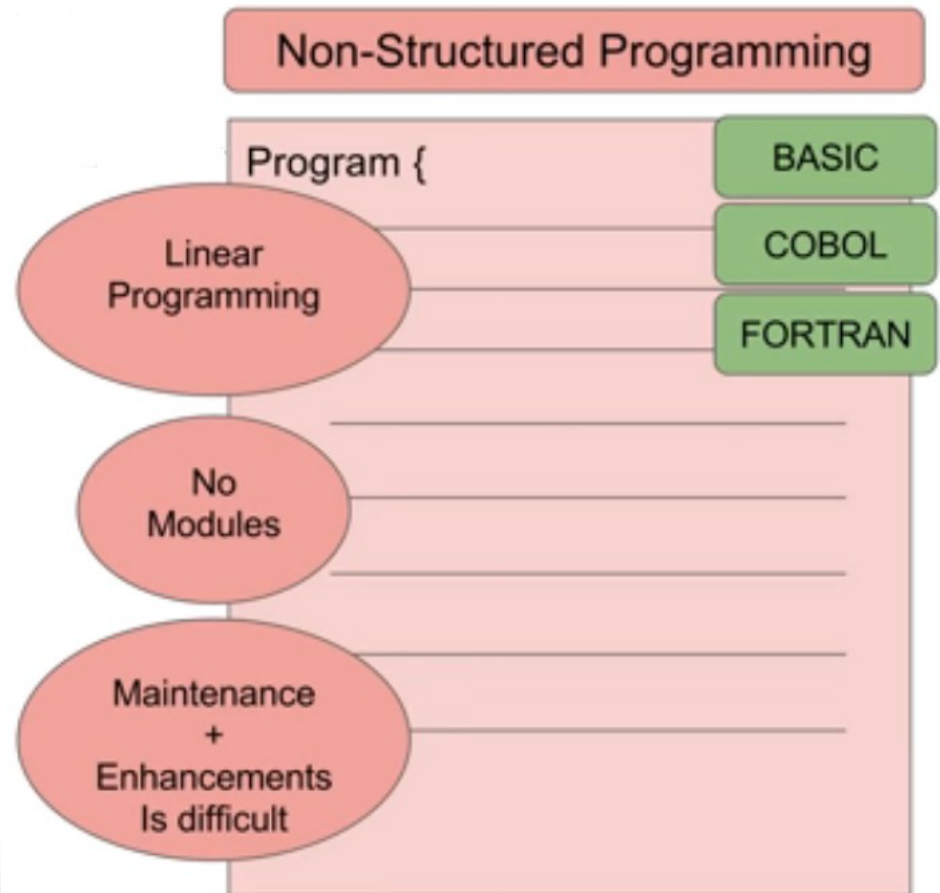
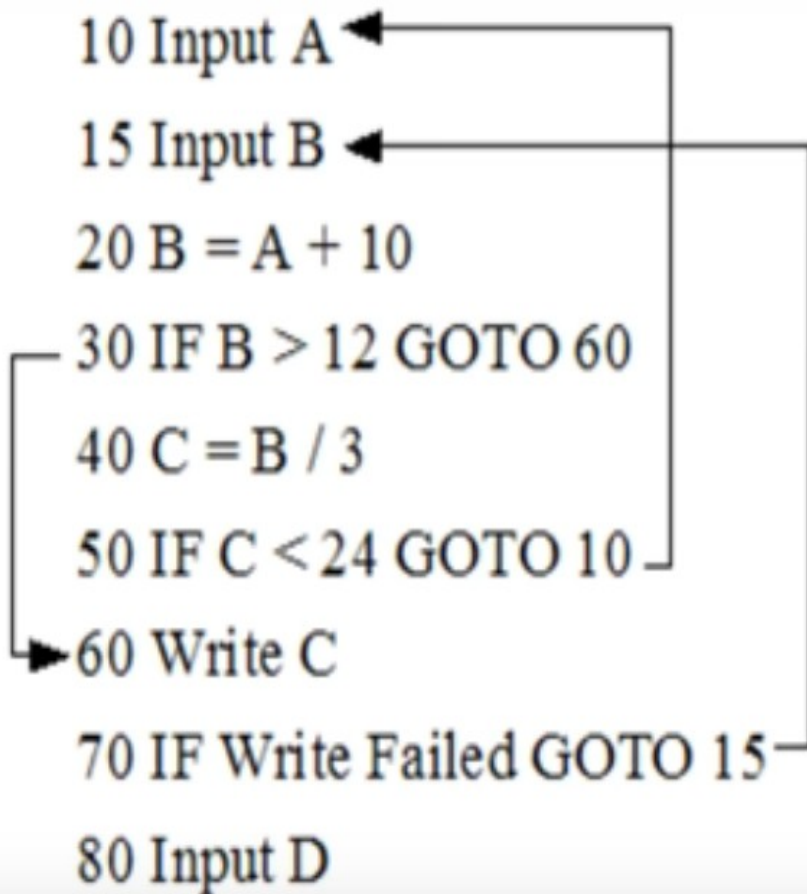
Step 3: Enter the amount.

Step 4: Withdraw Cash.

Step 5: Withdraw Card.

Programming Example using Non structured methodology

```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```



Programming Example using Structured methodology

Structured Programming

```
withdrawMoney() {  
    _____  
    _____  
    _____  
}
```

```
depositMoney() {  
    _____  
    _____  
    _____  
}
```

```
openAccount() {  
    _____  
    _____  
    _____  
}
```

A person **opens account** with a bank and **deposits money** and uses his Credit Card to **withdraw money** from ATM.

Structured Programming

Works on **ACTIONS**

openAccount()

depositMoney()

withdrawMoney()

Programming Example using Object Oriented methodology

OOP - Object Oriented Programming

Does not work on **ACTIONS**

Works With

Classes

Objects

OOP - Object Oriented Programming

An Account Holder
withdraws money from
his Bank Account using
Credit Card.

```
class AccountHolder{  
}
```

```
class BankAccount{  
}
```

```
class CreditCard{  
}
```

Programming Example using Object Oriented methodology

OOP - Object Oriented Programming

An **Account Holder** withdraws money from his **Bank Account** using **Credit Card**.

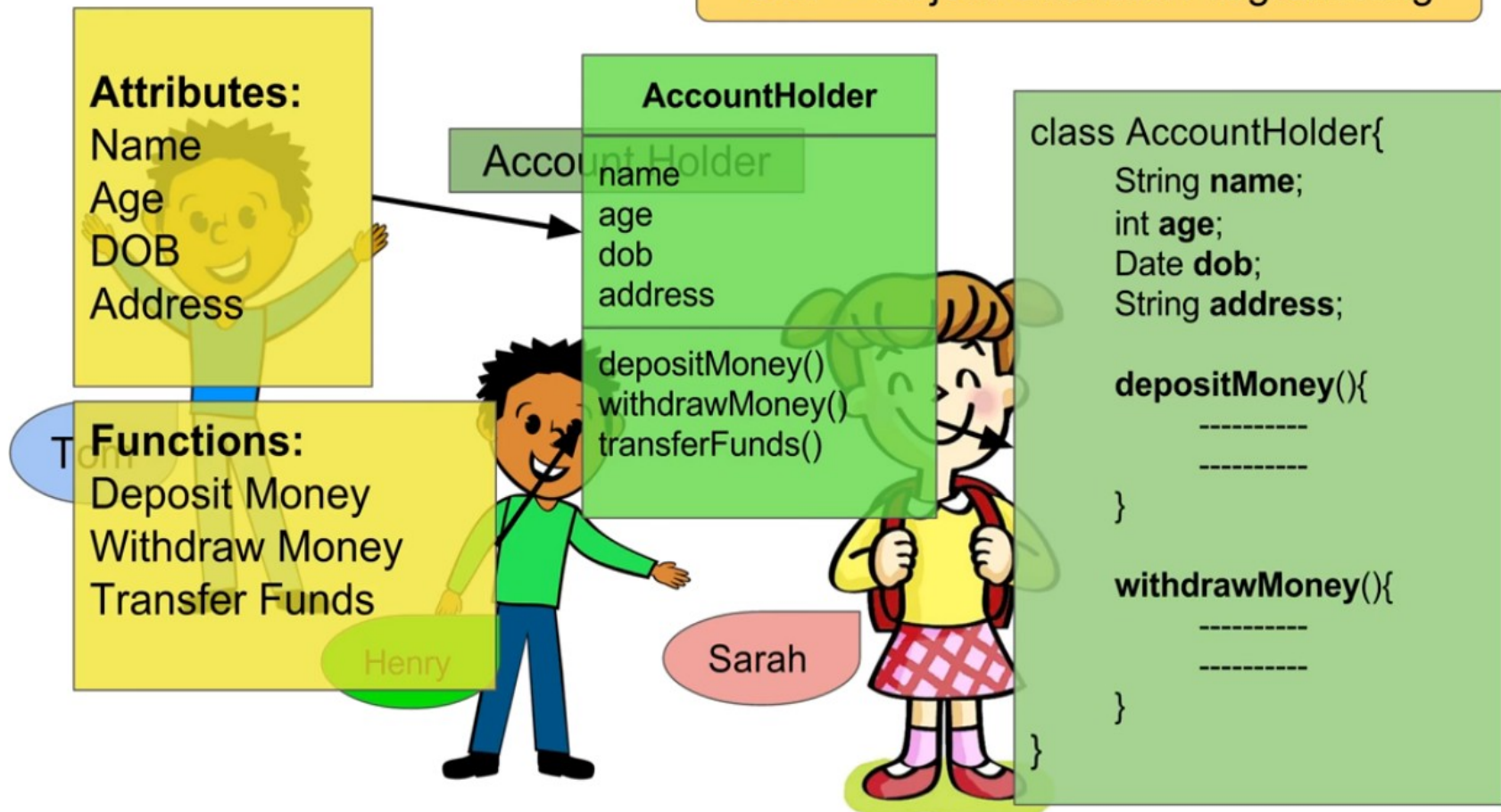
```
class AccountHolder{  
}
```

```
class BankAccount{  
}
```

```
class CreditCard{  
}
```

Programming Example using Object Oriented methodology

OOP - Object Oriented Programming



Programming Example using Object Oriented methodology

OOP - Object Oriented Programming

Class is a template to define OBJECTS

Account Holder

Using a class (template) multiple objects can be defined (created)

Tom

Henry

Sarah

```
class AccountHolder{  
    String name;  
    int age;  
    Date dob;  
    String address;  
  
    depositMoney(){  
        -----  
        -----  
    }  
  
    withdrawMoney(){  
        -----  
        -----  
    }  
}
```


Programming Example using Object Oriented methodology

OOP - Object Oriented Programming

Classes

```
class AccountHolder{  
}
```

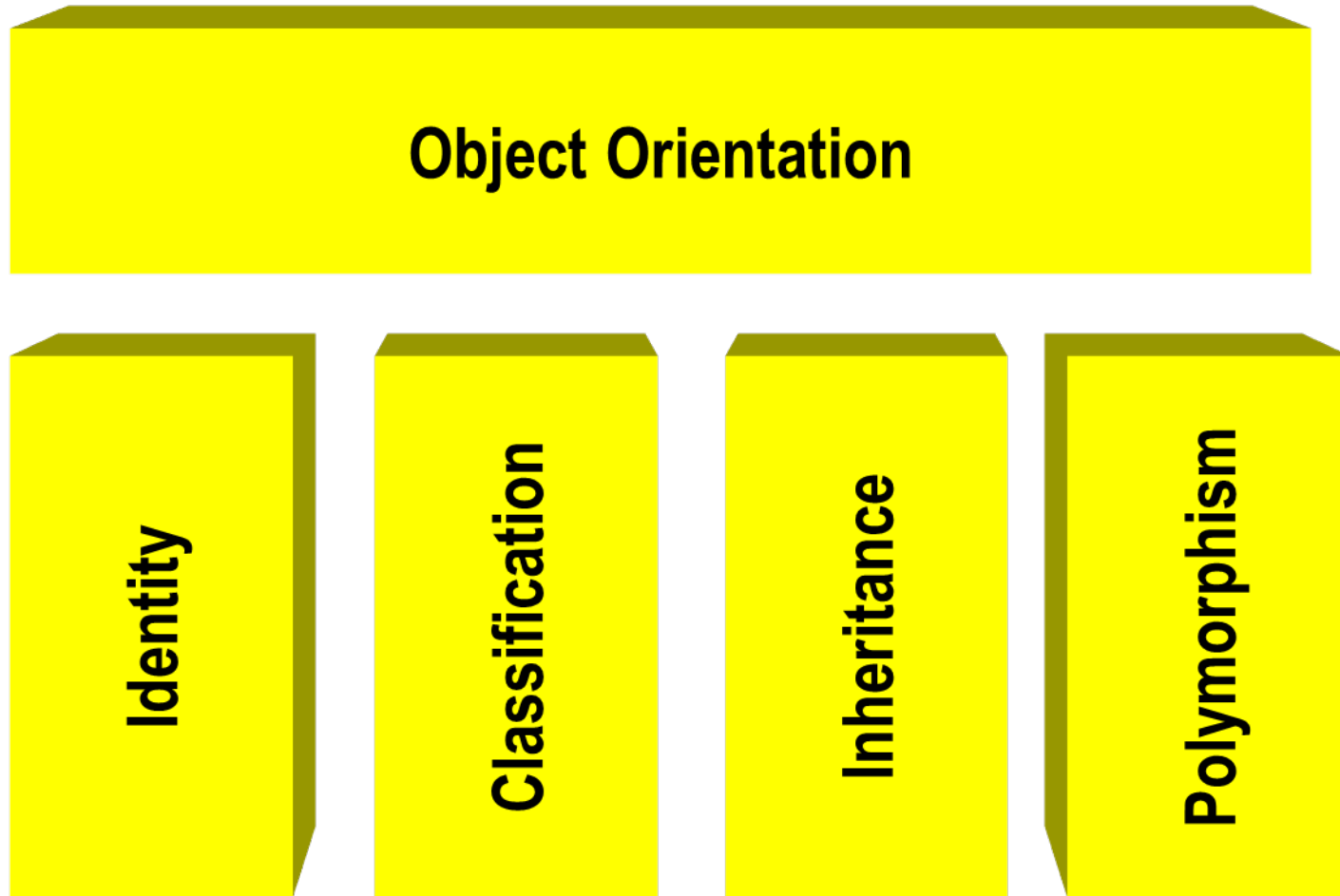
```
class BankAccount{  
}
```

```
class CreditCard{  
}
```

Objects



Basic Principles of object Orientation



Basic Principles of object Orientation

- **Identity:** Quantized data into discrete , distinguishable entities called objects
- **Classification:** Objects with the same data structure (attributes) and behavior (operations) are grouped into a class.
- **Inheritance:** Sharing of attributes and operations– among classes based on hierarchical relationship.
- **Polymorphism:** Same operation may behave differently for different classes.

Object- Oriented Methodology

- Object-Oriented Methodology is a set of
 - Methods
 - Models
 - Rulesfor Developing the system.
- It consists of Building a model of an application and then adding details to it during design.

Object- Oriented Methodology

The Methodology has the following stages.

1.System conception

2.Analysis

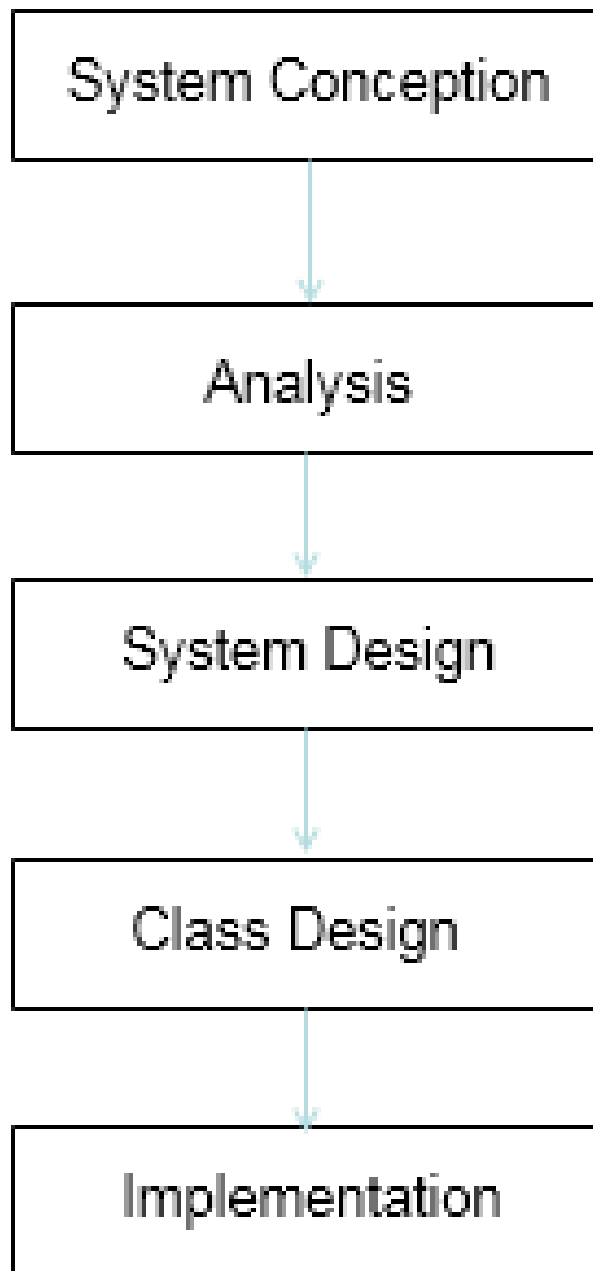
3.System Design

4.Class Design

5.Implementation

Object- Oriented Methodology

- **System conception:** Software development begins with analysts.
- **Analysis:** The problem is formulated, user requirements are identified, and then a model is built based upon real-world objects.
- **System Design:** The complete architecture of the desired system is designed.
- **Class Design:** The class designer adds details to the analysis model in accordance with the system design strategy.
- **Implementation:** Implementers implements into a programming language.



Concepts of OOP:

Object

Abstraction

Inheritance

Class

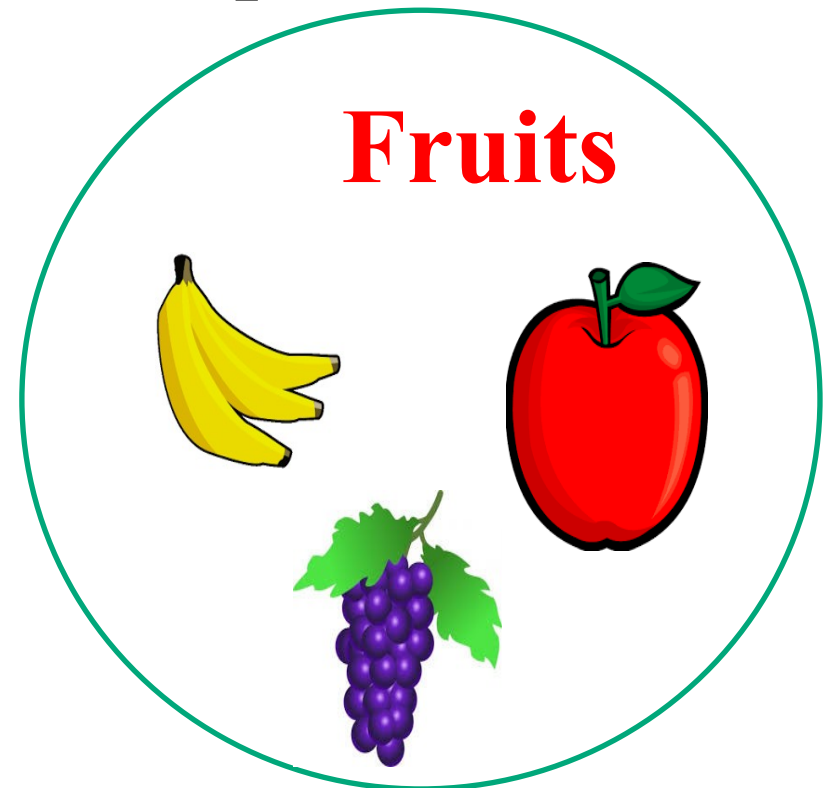
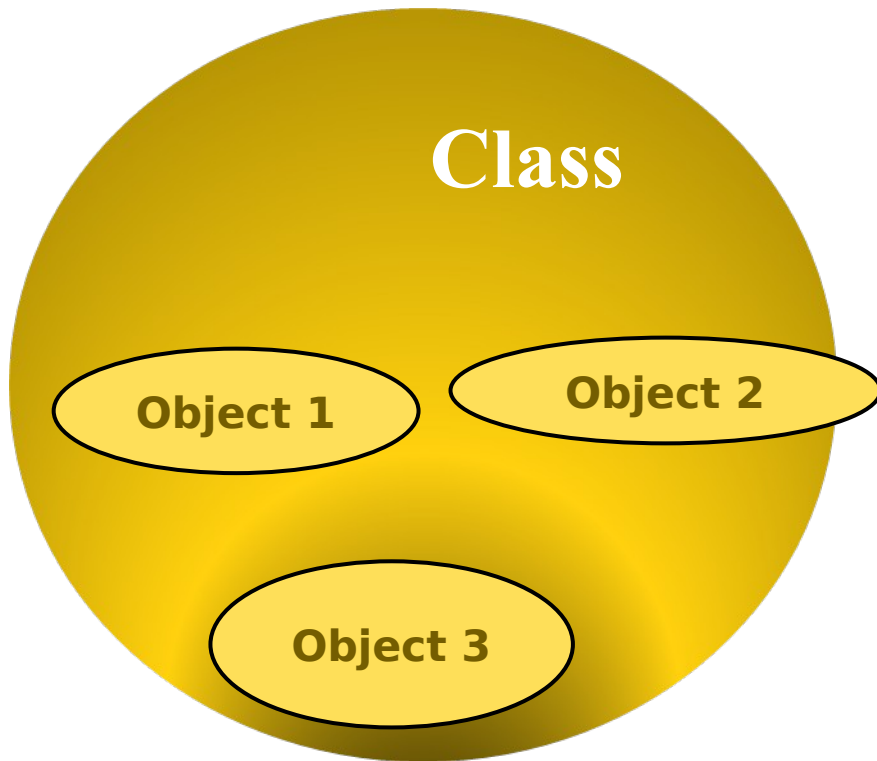
Encapsulation

Polymorphism

Concepts of OOP:

- **Class:** Collection of Objects.
- **Blueprint** of any functional entity which defines its **properties and functions**

Example:



What is a Class?

- software “blueprints” for objects are called **classes**
- Definition:

A **class** is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind

Class Example

Television



Properties

Screen Height

Screen Width

Screen Shape

Functionalities

On Off Switch

Volume Control

Tuner

Class and Object with Example

Television

CLASS



Object



Silver
Magic

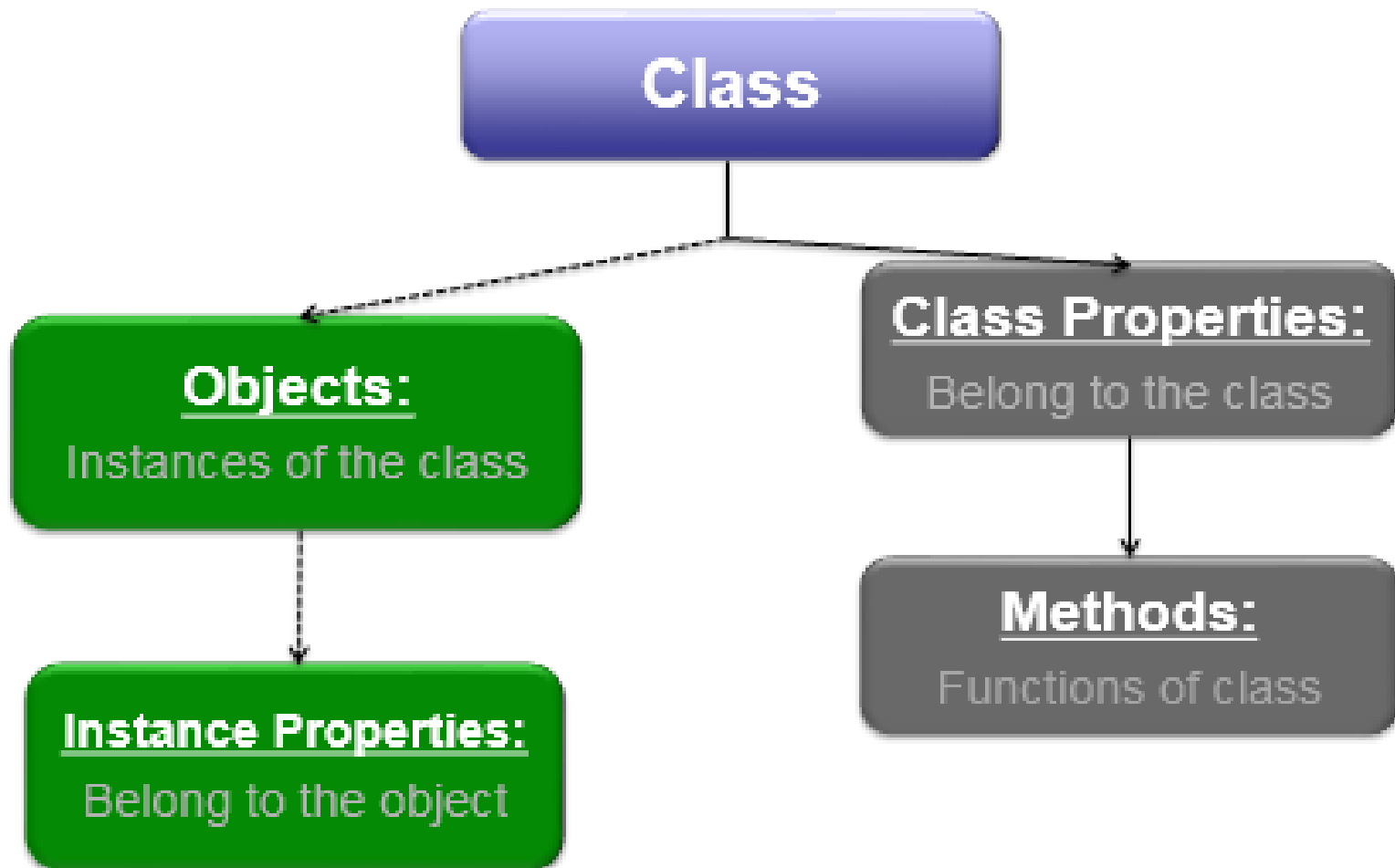
Instance of

Instance of

Wooden
Classic



Classes



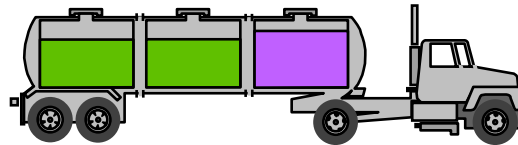
Almost everything in the world can be represented as an object

- A flower, a tree, an animal
- A student, a professor
- A desk, a chair, a classroom, a building
- A university, a city, a country
- The world, the universe
- A subject such as CS, IS, Math, History, ...

More about objects

- Informally, an object represents an entity, either physical, conceptual, or software.

- Physical entity



Truck

- Conceptual entity



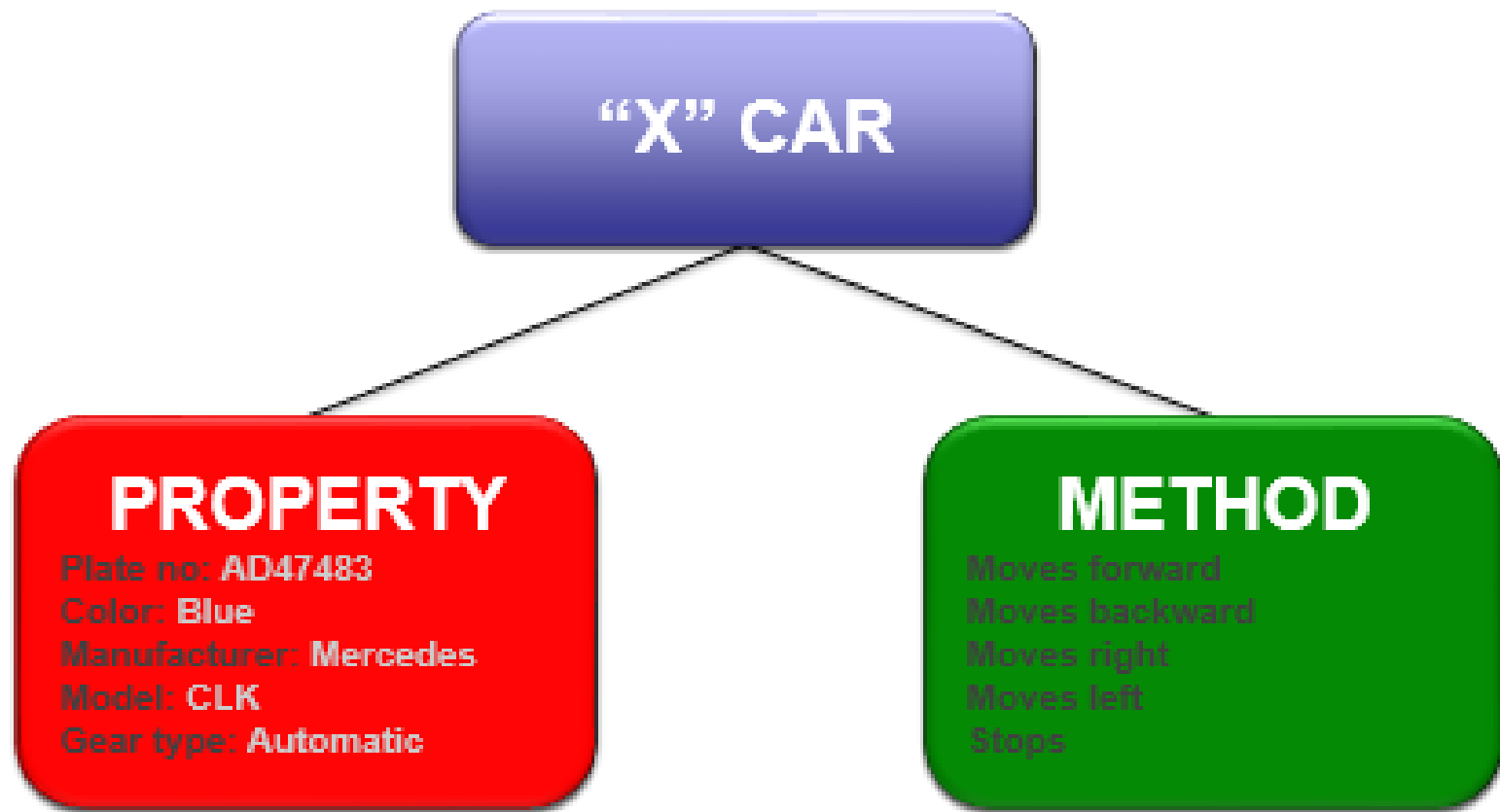
Chemical
Process

- Software entity



Linked
List

Classes & Objects



Class/Object

Each copy of an object from a particular class is called an ***instance*** of the class.



Class/Object

The act of creating a new instance of an class is called **instantiation**.



In short...

- An Object is a Class when it comes alive!
- Student is a class, Suresh and Ramesh are objects
- Animal is a class, the cat is an object
- Vehicle is a class, My neighbor's BMW is an object
- Galaxy is a class, the MilkyWay is an object

Technical contrast between Objects & Classes

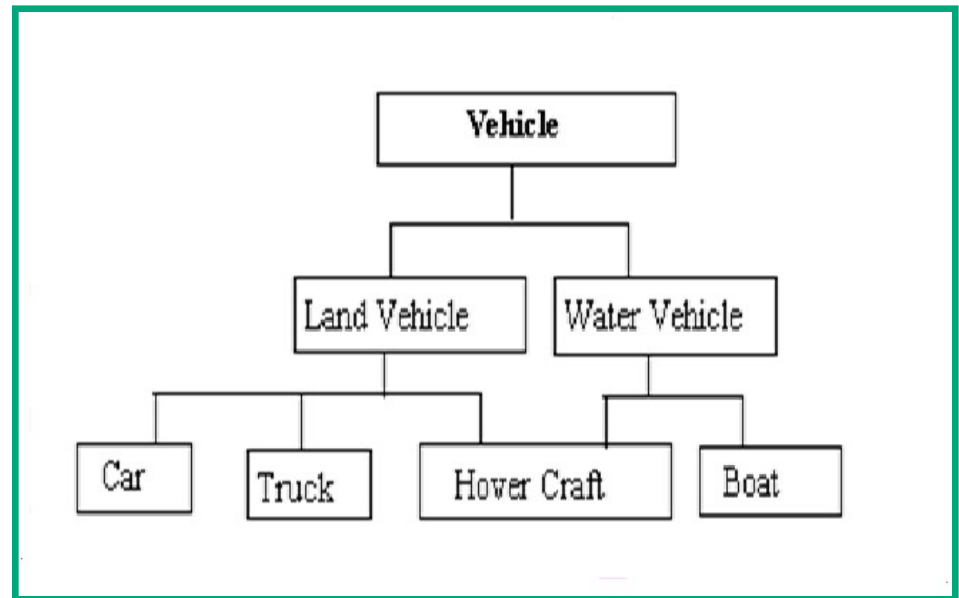
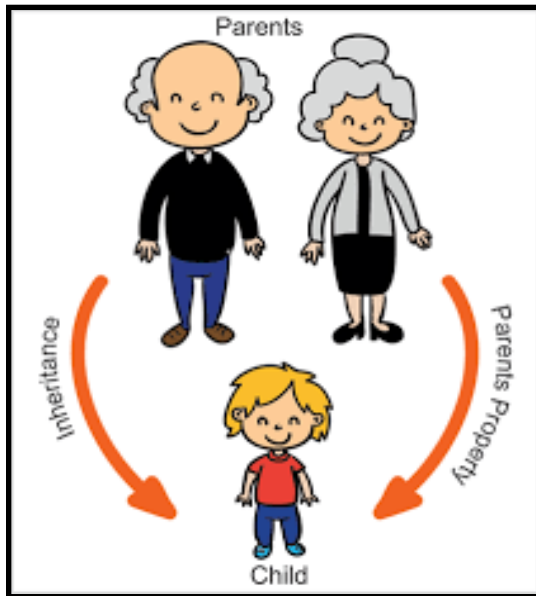
CLASS	OBJECT
Class is a data type	Object is an instance of Class.
It generates OBJECTS	It gives life to CLASS
Does not occupy memory location	It occupies memory location.
It cannot be manipulated because it is not available in memory (<i>except static class</i>)	It can be manipulated.

Object is a class in “runtime”

Concepts of OOP:

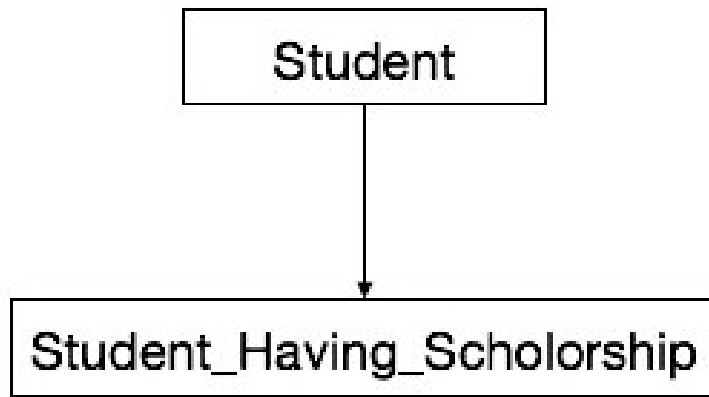
Inheritance:

- When one object acquires all the properties and behaviours of parent object i.e. known as inheritance.
- It provides code reusability.

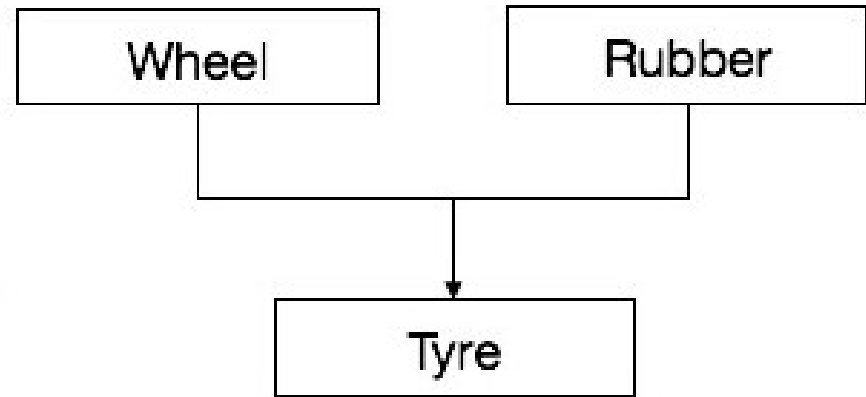


Types of Inheritance

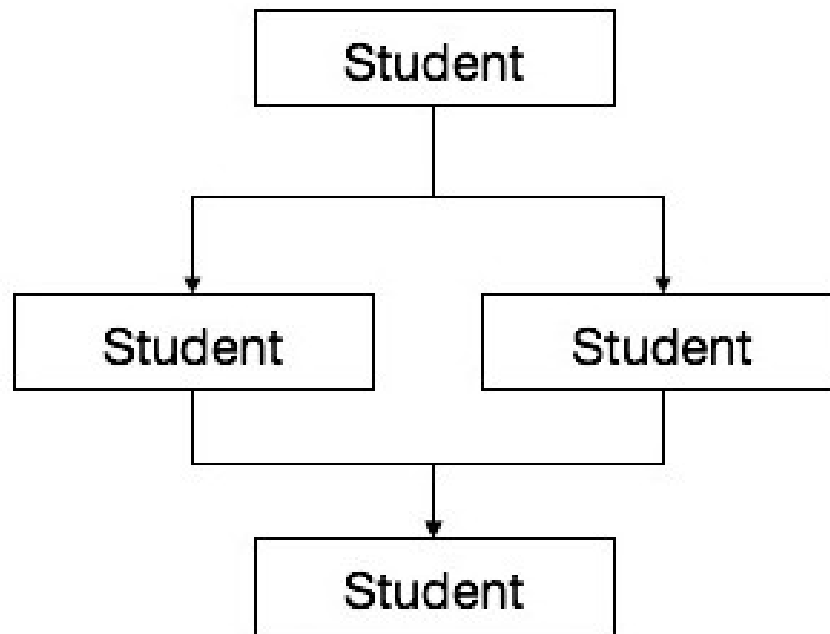
- **Single Inheritance** – A subclass derives from a single super-class.
- **Multiple Inheritance** – A subclass derives from more than one super-classes.
- **Multilevel Inheritance** – A subclass derives from a super-class which in turn is derived from another class and so on.
- **Hierarchical Inheritance** – A class has a number of subclasses each of which may have subsequent subclasses, continuing for a number of levels, so as to form a tree structure.
- **Hybrid Inheritance** – A combination of multiple and multilevel inheritance so as to form a lattice structure.



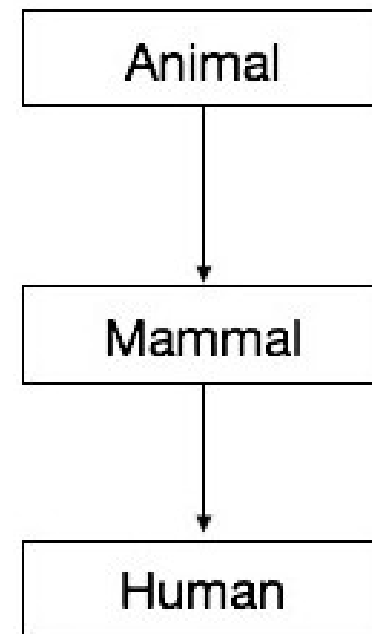
Single Inheritance



Multiple Inheritance

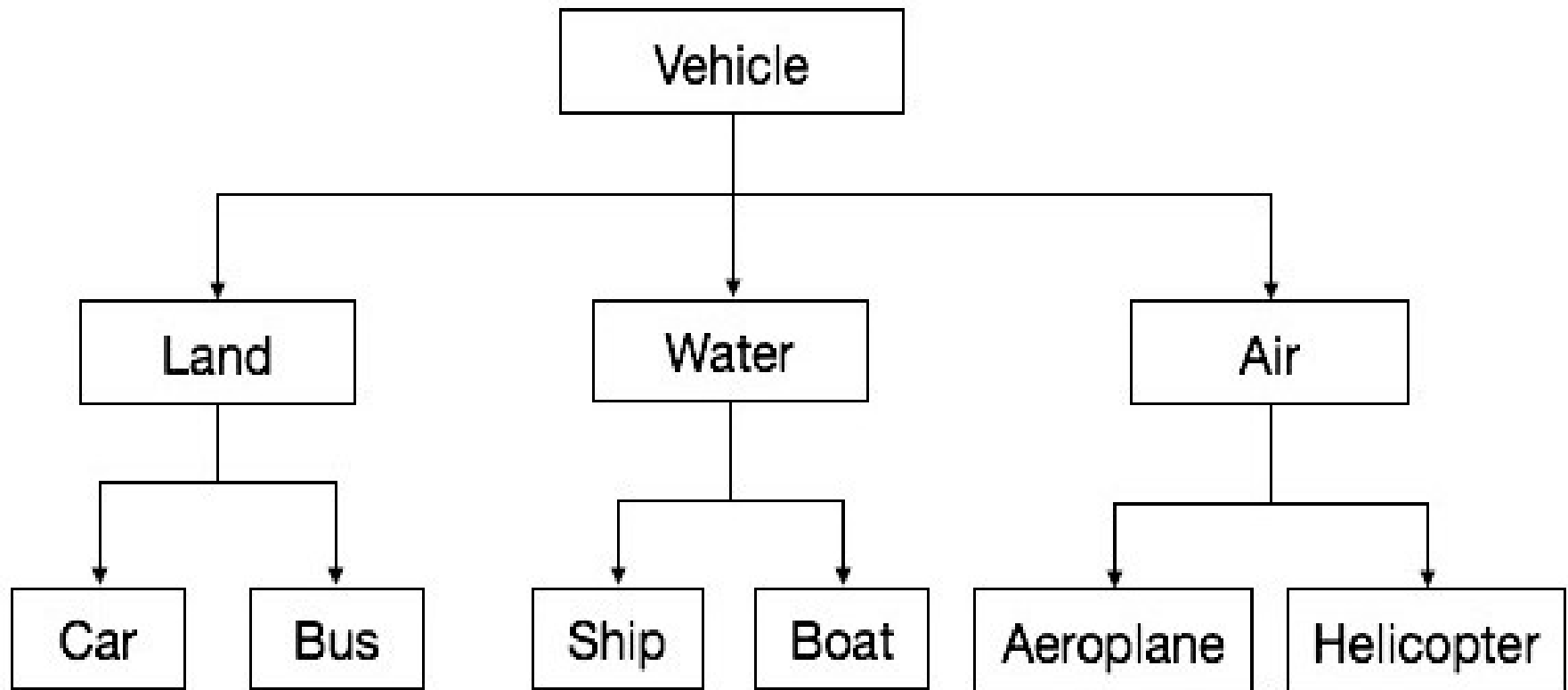


Hybrid Inheritance



Multi-level Inheritance

Contd...



Hierarchical Inheritance

Concepts of OOP:

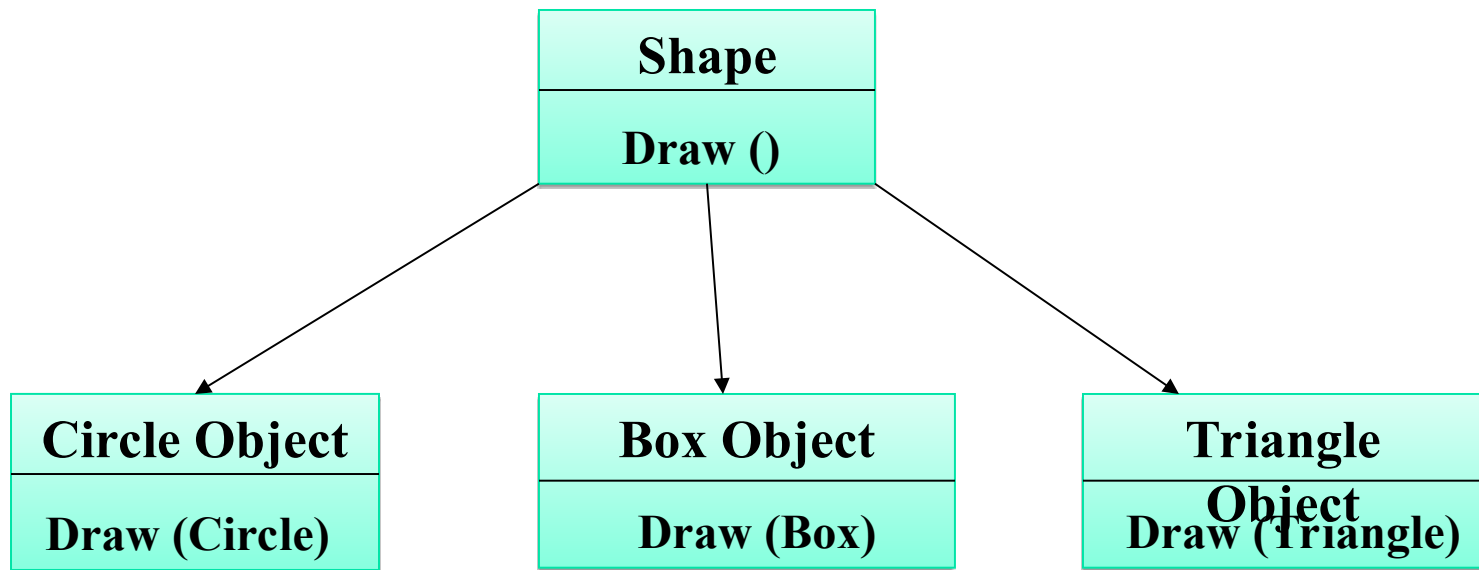
Polymorphism:

- When **one task is performed by different ways** i.e. known as polymorphism.
- For example:
 - To draw something e.g. shape or rectangle etc.
 - To speak something e.g. cat speaks meow, dog barks woof etc.



Polymorphism

- Single function name can be used to handle different number and different kind of arguments. This is something similar to a particular word having different meaning.
- Using a single function name to perform different types of tasks is known as function overloading.



Polymorphism

Concepts of OOP:

Encapsulation:

- Mechanism of **wrapping the data (variables) and code acting on the data (methods) together as a single unit.**



- The variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.
- Also known as data hiding.
- For example: capsule, it is wrapped with different medicines.

Concepts of OOP:

- **Abstraction:**
- **Hiding internal details and showing functionality** is known as abstraction.
- For example: phone call, we don't know the internal processing.



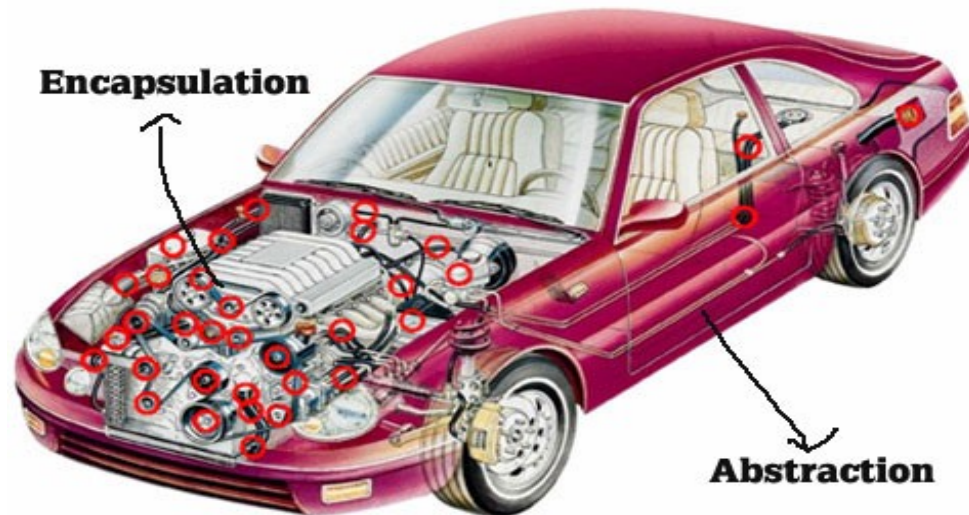
Real Life Example of Abstraction

Data Abstraction

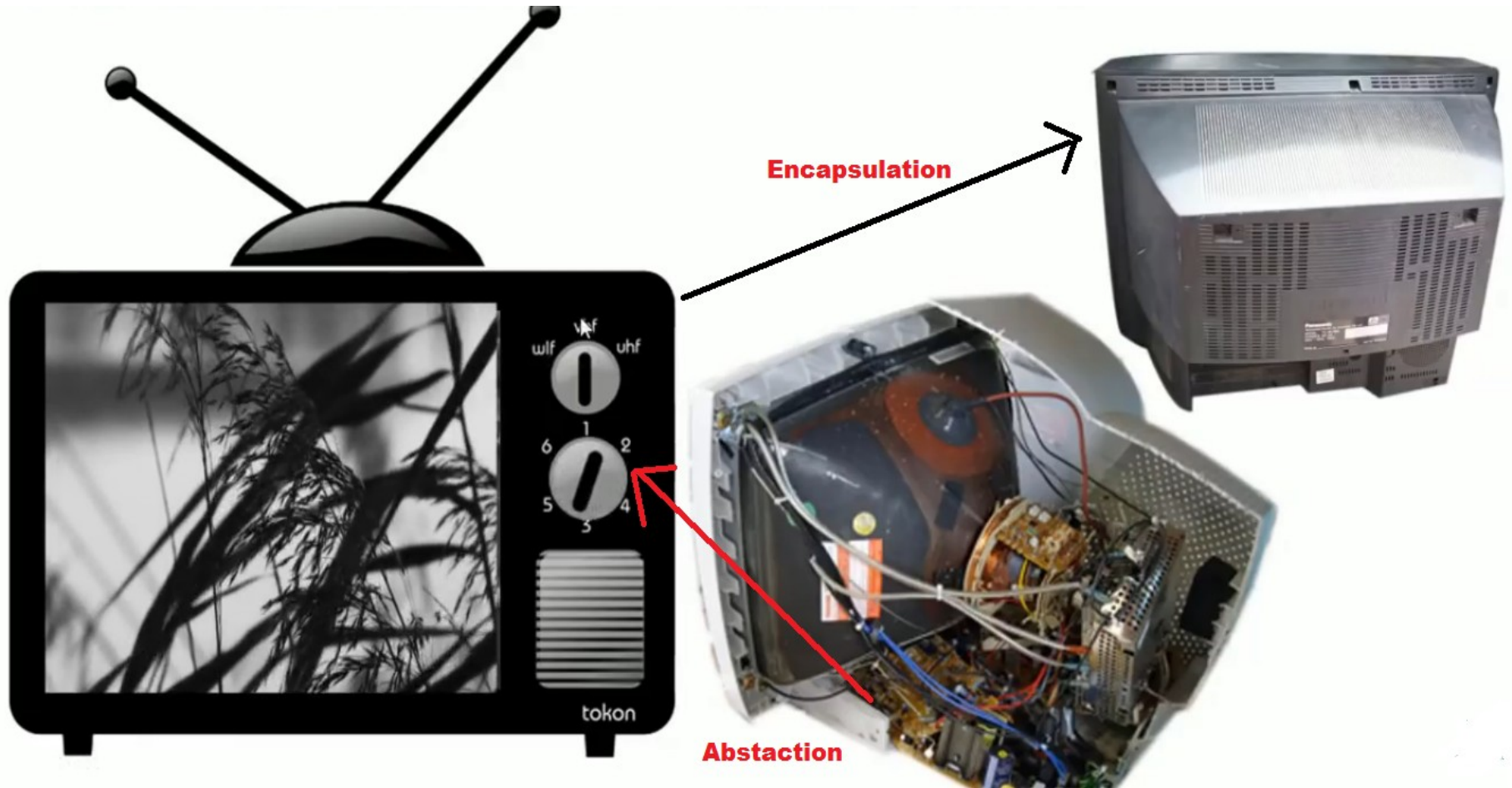
- **Data Abstraction:** Abstraction refers to the act of representing essential features without including the background details or explanation.
- Classes encapsulate all the essential properties of the objects that are to be created.
- The attributes are sometimes called data members, because they hold information.
- The functions that operate on these data are sometimes called methods.

Whether the Encapsulation and Abstraction are same or different?

- **Encapsulation:** Hiding unnecessary things from outside world is known as encapsulation
- **Abstraction:** Displaying only necessary things to the outside world is known as abstraction

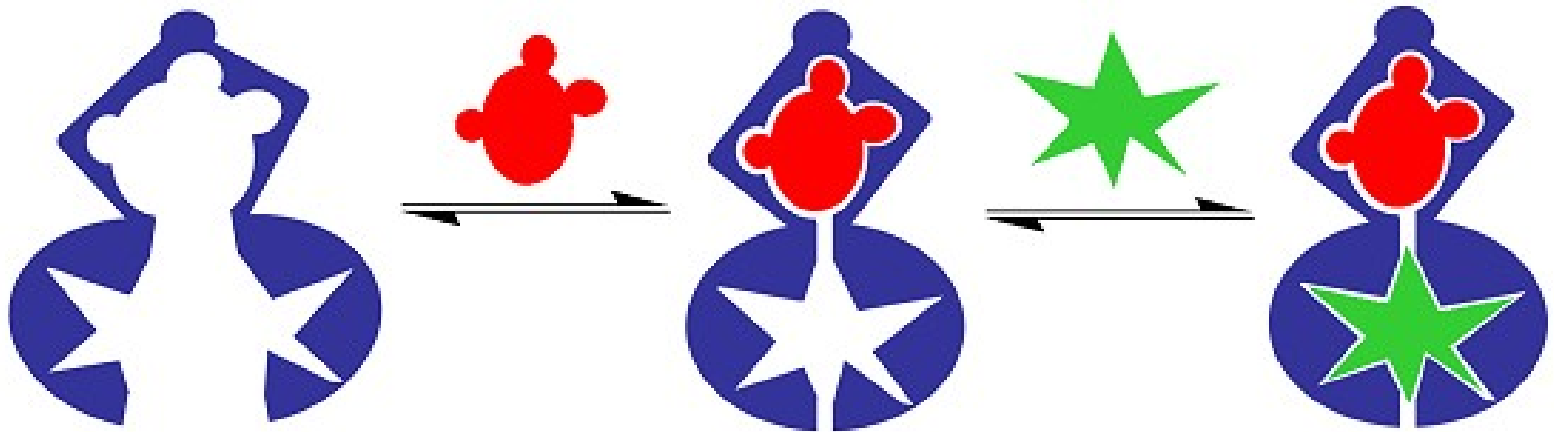


Example of Encapsulation and Abstraction



Dynamic Binding

- Compiler should match function calls with the correct definition at the run time



Dynamic Binding

- Binding refers to the linking of a procedure call to the code to be executed in response to the call.
- Dynamic binding means that the code associated with a given procedure call is known until the time of the call at run-time.
- It is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depend on the dynamic type that reference.

Message Passing



Message Passing

- An object oriented programming consists of a set of objects that communicate with each other.
- Object oriented programming involves the following steps:
 1. Creating classes that define objects and their behaviour.
 2. Creating object from class definition
 3. Establishing communication among objects.
- Objects communicate with one another by sending and receiving information.
- A message for an object is a request for execution of a procedure and therefor will invoke a function in the receiving object that generate the desired result.

OO Themes

Following are the OO Themes

- Abstraction
- Encapsulation
- Combining Data and Behavior
- Sharing
- Emphasis on the Essence of an Object
- Synergy

OO Themes

- **Combining Data and Behavior:** Data Structure matches the operation inheritance hierarchy
- **Sharing:** No redundancy, reusability.
- **Emphasis on the Essence of an Object :** Focus on what an object is rather than how it is used.
- **Synergy:** Identity, classification, polymorphism, inheritance.

Characteristics of OOPs

1. Objects
2. Classes
3. Data abstraction
4. Data encapsulation
5. Information Hiding
6. Message passing
7. Inheritance
8. Dynamic binding
9. Polymorphism
10. Overloading

Overloading

- It allows an object to have different meanings depending on the context.
- 2 types:
 1. Operator Overloading
 2. Function Overloading

Overloading is one type of polymorphism

Introduction to OO Models

- There are three models to describe a system .

1)Class Model

2)State Model

3)Interaction Model

Class Model

- The Class Model describes the static structure of objects in a system and their relationships.
- The Class Model contains class diagrams. A **class diagrams** is a graph whose nodes are classes and whose arcs are relationships among classes.

State Model

- The State Model describes the aspects of an object that change over time.
- The State Model contains State diagrams. A **state diagrams** is a graph whose nodes are state and whose arcs are transitions between states caused by events.
- The state models specifies and implements control with state diagrams.

Interaction Model

- The Interaction Model describes how the objects in a system cooperate to achieve the results.
- The Interaction Model starts with the use cases that are then elaborated with sequence and activity diagrams.
- A **use case** focuses on the functionality of a system – that is what a system does for users.
- A **sequence diagram** shows the objects that interact and the time sequence of their interactions.
- An **activity diagram** elaborates important processing steps.

Advantages of OOP

Modularity:

- The source code for a class can be written and maintained independently of the source code for other classes.
- Once created, an object can be easily passed around inside the system.

Information -hiding:

- By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.

Code re-use:

- If a class already exists, you can use objects from that class in your program. This allows programmers to implement/test/debug complex, task-specific objects, which you can then use in your own code.

Easy Debugging :

- If a particular object turns out to be a problem, you can simply remove it from your application and plug in a different object as its replacement.
- This is analogous to fixing mechanical problems in the real world. If a bolt breaks.

Advantages of Object Oriented Programming

1. Inheritance – eliminate redundant code and extend the use of existing classes.
2. We can build programs from the standard working modules that communicate with one another , rather than having to start writing the code from scratch.
3. The principle of Data hiding helps the programmer to build secure programs that can not be invaded by code in other parts of the program.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. It is possible to map objects in the problem domain to those in the program.
6. It is easy to partition the work in a project based on objects.
7. The data centered design approach enables us to capture more details of a model in implementable form.

Contd...

8. Object-oriented system can be easily upgraded from small to large systems.
9. Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
10. Software complexity can be easily managed.

Disadvantages of OOP

1. Steep learning curve
2. Could lead to larger program sizes
3. Could produce slower programs

Task for students:- Detailed description.

Applications of OOP

1. Real-Time System
2. Simulation & Modeling
3. Object Oriented Database
4. Hypertext and hypermedia
5. AI and expert system
6. Neural Network & Parallel Programming
7. Decision Support and Office Automation System
8. CIM/CAM/CAD System

Benefits of Object Model

The benefits of using the object model are –

- 1.It helps in faster development of software.
- 2.It is easy to maintain. Suppose a module develops an error, then a programmer can fix that particular module, while the other parts of the software are still up and running.
- 3.It supports relatively hassle-free upgrades.
- 4.It enables reuse of objects, designs, and functions.
- 5.It reduces development risks, particularly in integration of complex systems.

Structured Programming

Object Oriented Programming

Structured Programming is designed which focuses on **process/** logical structure and then data required for that process.

Structured programming follows **top-down approach**.

Structured Programming is also known as **Modular Programming** and a subset of **procedural programming language**.

In Structured Programming, Programs are divided into small self contained **functions**.

Structured Programming is **less** secure as there is no way of **data hiding**.

Structured Programming can solve **moderately** complex programs.

Structured Programming provides **less reusability**, more function dependency.

Less abstraction and less flexibility.

Object Oriented Programming is designed which focuses on **data**.

Object oriented programming follows **bottom-up approach**.

Object Oriented Programming supports **inheritance, encapsulation, abstraction, polymorphism**, etc.

In Object Oriented Programming, Programs are divided into small entities called **objects**.

Object Oriented Programming is more secure as having data hiding feature.

Object Oriented Programming can solve any **complex** programs.

Object Oriented Programming provides more reusability, less function **dependency**.

More abstraction and more **flexibility**.

Summary

- C++ is a superset of C language
- C++ adds a number of object oriented features such as objects, inheritance, function overloading and operator overloading to C
- These features enables building of programs with clarity, extensibility and ease of maintenance.

MCQ

1. **Object oriented programming employs_____ programming approach.**
A. top-down B. Bottom-up
C. Procedural D.All of these
2. **Classes in CPP are_____ .**
A. Derived Data Types B. User Defined Data types
C. Built –in Data Types D. All o f These
3. **_____ is the OOP feature and mechanism that binds together code and the data it manipulates, and keep both safe from outside world.**
A. Data Binding B. Data Encapsulation
C. Data Storing D. Data Abstraction

Answers

- 1. Bottom –up**
- 2. User Defined Data Types**
- 3. Data Encapsulation**

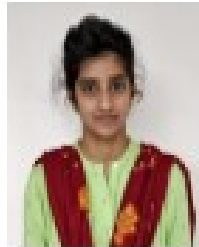
Presentation Prepared By:



Ms. Atufaali Saiyed



Ms. Shraddha Vyas



Ms. Krishna Patel



Ms. Dweepna Garg
Subject Coordinator

Other Lecture Faculties of DEPSTAR:



Mr. Parth Goel



Mr. Hardik Javswal



Ms. Khushi Patel

Contact us:

dweepnagarg.ce@charusat.ac.in
parthgoel.ce@charusat.ac.in
hardikjayswal.it@charusat.ac.in
krishnapatel.ce@charusat.ac.in
khushipatel.ce@charusat.ac.in

Thank you!

