CS-1004 Object Oriented programming Week 1

Instructor:
ATIYA
LECTURER
(COMPUTER SCIENCE DEPARTMENT)
NATIONAL UNIVERSITY- FAST (KHI CAMPUS)
EMAIL: ATIYA.JOKHIO@NU.EDU.PK
ROOM: (LECTURER ROOM#19, CS DEPT BASEMENT-2)

What you have done so far !

- Programming Fundamentals
  - How to think a program.
  - How to write a program.
  - Basic Programming structure.
  - Procedural paradigm.
  - Group of functions that interact with each other.

- Already have knowledge about the offered course (Repeater’s Course).
- Need to strengthen our concepts.
- Try to implement what we know.

Computer Programming as a course

What we will study:

- Object Oriented Programming.
- How to think in a OOP way.
- How to map real world into a program
- Or, how to program a real world scenario.
- Aim :
  - Our aim is to lean the concepts of Object Oriented programming.
  - Try to digest them.
  - Implement in a program.
  - Tool: C++.

Contents of the Course

- Object Oriented Programming.

- Classes & Objects.

- Overloading.

- Inheritance.

- Polymorphism.

- Generic Programming.

- Exception Handling.

Books

Text Book:

1- C++ How to program By Deitel & Deitel.

Reference Books:

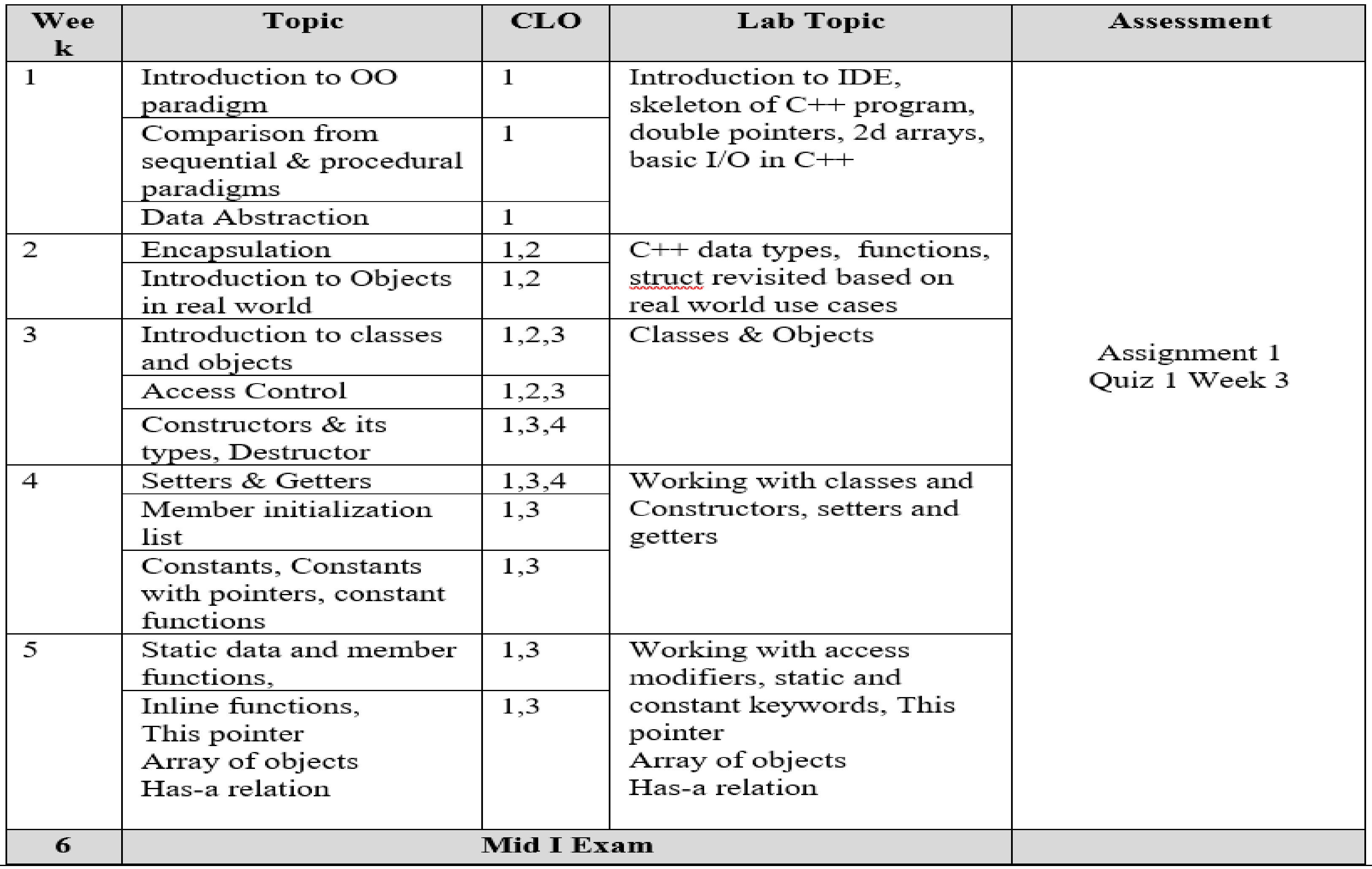
1- The C++ Programming Language By Bjarne Stroustrup.

2- Object Oriented Software Engineering By Jacobson.

Grading policy

Assignments	08	at least three
Quizzes	12	at least three
Midterm’s	30	two (15 marks each)
Final	50	
	_____	
Total	100	

Course Outline [Till Mid-One]



Week	Topic	CLO	Lab Topic	Assessment
1	Introduction to OO paradigm	1	Introduction to IDE, skeleton of C++ program, double pointers, 2d arrays, basic I/O in C++	Assignment 1 Quiz 1 Week 3
	Comparison from sequential & procedural paradigms	1		
	Data Abstraction	1		
2	Encapsulation	1,2	C++ data types, functions, struct revisited based on real world use cases	
	Introduction to Objects in real world	1,2		
3	Introduction to classes and objects	1,2,3	Classes & Objects	
	Access Control	1,2,3		
	Constructors & its types, Destructor	1,3,4		
4	Setters & Getters	1,3,4	Working with classes and Constructors, setters and getters	
	Member initialization list	1,3		
	Constants, Constants with pointers, constant functions	1,3		
5	Static data and member functions,	1,3	Working with access modifiers, static and constant keywords, This pointer Array of objects Has-a relation	
	Inline functions, This pointer Array of objects Has-a relation	1,3		
6	Mid I Exam			

What Are the Types of Programming Languages

- Programming languages allow programmers to code software.
  - First Generation Languages
  - Second Generation Languages
  - Third Generation Languages
  - Fourth Generation Languages
  - Fifth Generation Languages

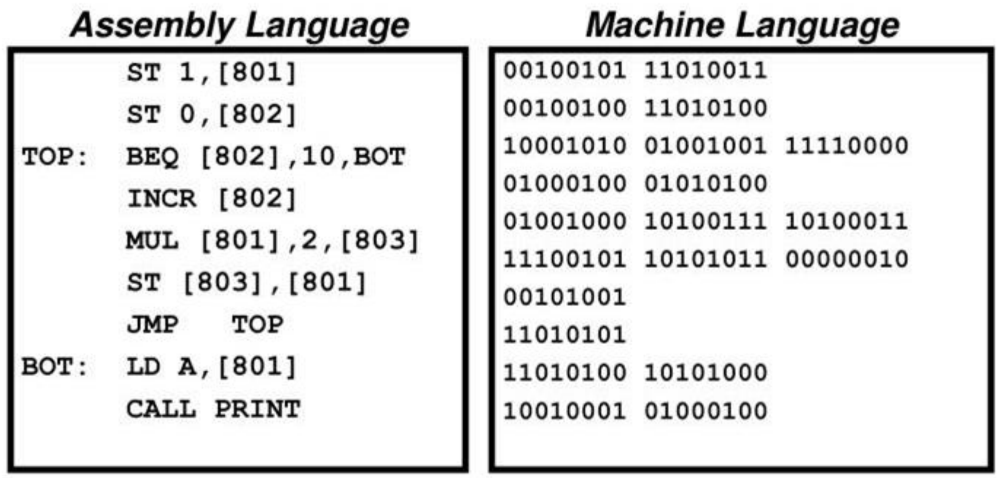
First Generation Languages

- These are low-level languages like machine language.
- Machine language
  - Operation code – such as addition or subtraction.
  - Operands – that identify the data to be processed.
  - Machine language is machine dependent as it is the only language the computer can understand.
  - Very efficient code but very difficult to write.

Second Generation Languages

- Assembly languages
  - Symbolic operation codes replaced binary operation codes.
  - Assembly language programs needed to be “assembled” for execution by the computer. Each assembly language instruction is translated into one machine language instruction.
  - Very efficient code and easier to write.
  - These are low-level assembly languages

    used in kernels and hardware drives.



Assembly Language

```
      ST 1,[801]
      ST 0,[802]
TOP:  BEQ [802],10,BOT
      INCR [802]
      MUL [801],2,[803]
      ST [803],[801]
      JMP   TOP
BOT:  LD A,[801]
      CALL PRINT
```

Machine Language

```
00100101 11010011
00100100 11010100
10001010 01001001 11110000
01000100 01010100
01001000 10100111 10100011
11100101 10101011 00000010
00101001
11010101
11010100 10101000
10010001 01000100
```

Third Generation Languages

- High Level Languages
- Closer to English but included simple mathematical notation.
  - Programs written in source code which must be translated into machine language programs called object code.
  - The translation of source code to object code is accomplished by a machine language system program called a compiler.

Third Generation Languages (cont'd.)

- Alternative to compilation is interpretation which is accomplished by a system program called an interpreter.
- Common third generation languages
  - FORTRAN
  - COBOL
  - C and C++
  - Visual Basic

Fourth Generation Languages

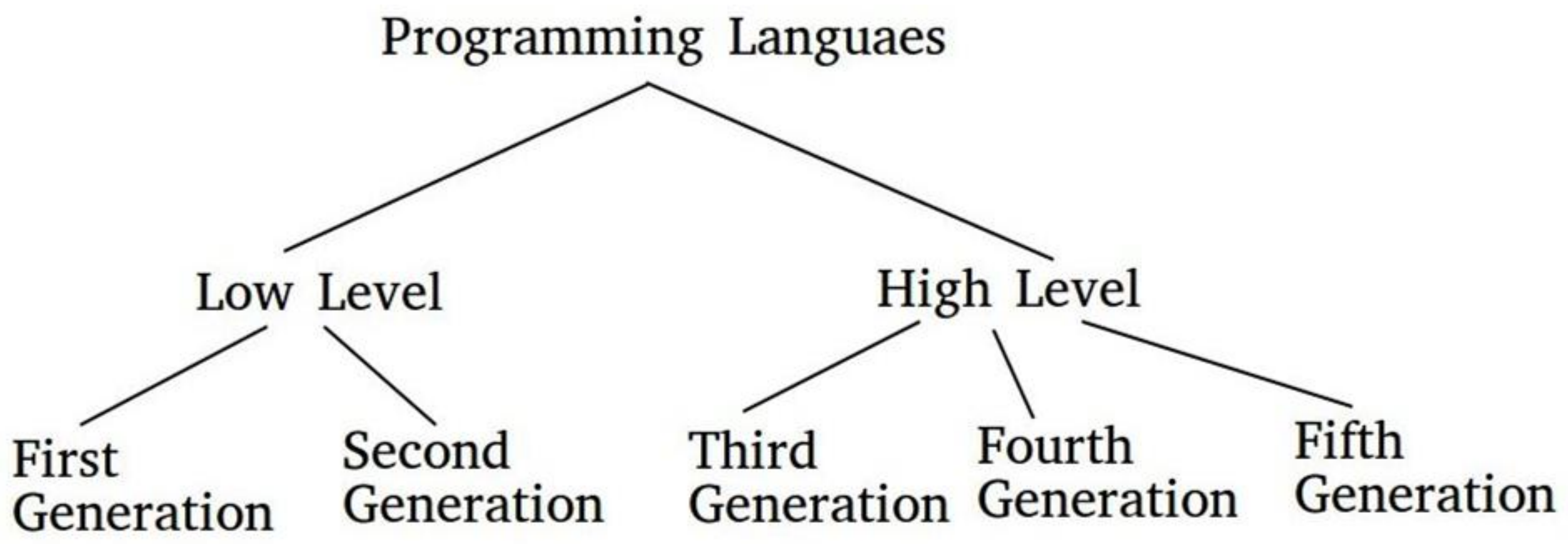
- Event based programming
- A high level language (4GL) that requires fewer instructions to accomplish a task than a third generation language.
- Used with databases
  - Query languages
  - Report generators
  - Forms designers
  - Application generators

Fifth Generation Languages

- Artificial Intelligence
- Machines that can think like humans
- Machines that takes its own decisions
  - These languages can be used to query the database in a fast and efficient manner.
  - In this generation of language, the user can communicate with the computer system in a simple and an easy manner.

Pictorial View

- The first two generations are called low level languages. The next three generations are called high level languages.



The C++ language features most directly support four programming styles

Procedural programming

PP can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions

Data abstraction

This is programming focused on the design of interfaces, hiding implementation details in general and representations

OOP programming

OOP can be defined as a programming model which is based upon the concept of objects

Generic programming

This is programming focused on the design, implementation, and use of general algorithms.

Agenda

What is programing Paradigm?

What is POP [Procedure Oriented Programming]

What is OOP [Object Oriented Programming]

Languages for OOP & POP

POP vs OOP

Programming Paradigms

The term **paradigm** = method to solve a problem

The term **programming paradigm** is used to specify an overall approach to writing program code

Many programming languages support multiple paradigms. Python, which is a popular general-purpose programming language, allows you to code using procedural, object-oriented.

In the early days of computer programming, most programs comprised a single procedure with many lines of code.

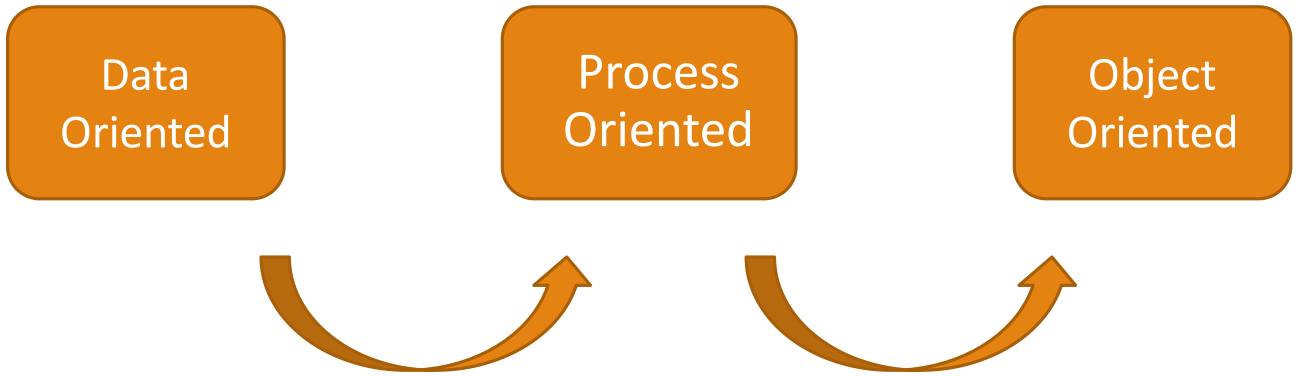
Programming Paradigms

1- Sequential

2- Procedural

3- Object Oriented

Programming Paradigms



Think in term of data that we needed

Think in term of Process that we needed

Think in term of Object that is involved

What is POP

A procedural language is a sort of computer programming language that has a set of functions, instructions, and statements that must be executed in a certain order to accomplish a job or program.

Procedural languages include BASIC, C, FORTRAN, Java, and Pascal, to name a few.

features:

a. Modularity
b. Predefined Function
c. Scoping

Advantages

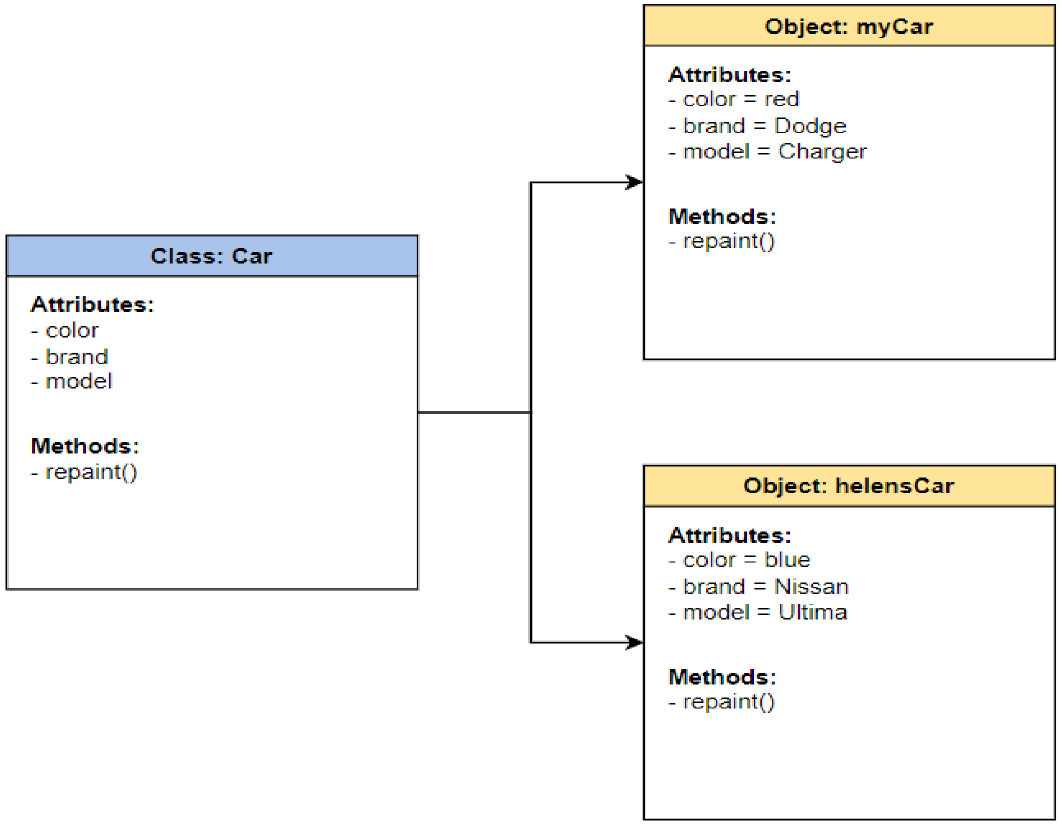
1. portable
2. Function
3. Flow Tracking

Disadvantages

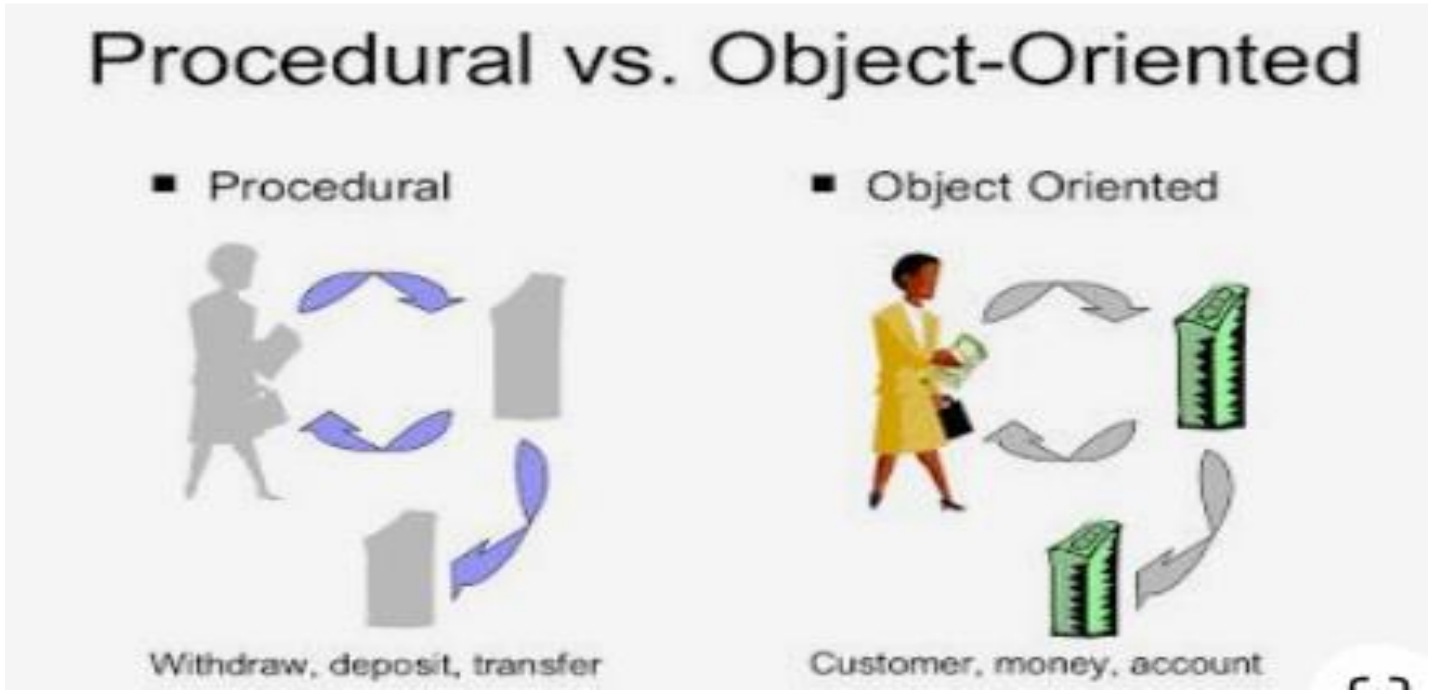
- large scale
- not work for real world problem
- Data Security

What is OOP

- Object-Oriented Programming (OOP) is a programming paradigm in computer science that relies on the concept of classes and objects.
- It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes),
- which are used to create individual instances of objects.



procedural vs object oriented



Procedural vs. Object-Oriented

- Procedural
- Object Oriented

Withdraw, deposit, transfer

Customer, money, account

Programming Paradigms

The **object-oriented programming paradigm** (**OOP**) introduces a fundamentally different approach to problem-solving. Rather than focusing on the problem that needs to be solved, OOP focuses on the **objects** that make up the system.

What is object orientation

- A technique for system modeling.

- OO model consists of several interacting objects.

What is a Model?

- abstraction of something.

- Purpose is to understand the product before developing it.

Example – OO Model

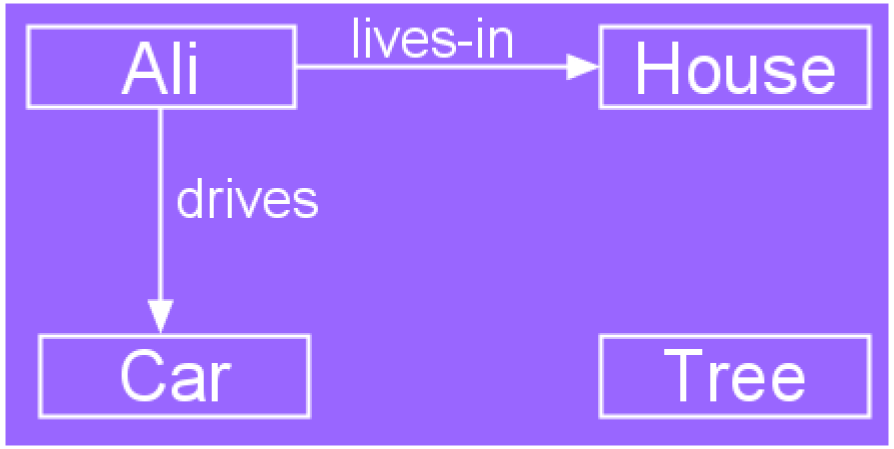
Example – OO Model

Objects

- Person i.e Name: Ali
- House
- Car
- Tree

Interactions

- Ali lives in the house
- Ali drives the car



Object-Orientation - Advantages

- People think in terms of objects

- OO models map to reality

Therefore, OO models are:

- easy to develop

- easy to understand

Five principles of OO paradigm

1- Abstraction: To have the relevant information.

2- Encapsulation: To hide information inside the object.

3- Polymorphism: To have many shapes / behaviors.

4- Inheritance: To create a new object with an existing one (To adopt features from others)

5- Reusability: Ability to use an object again and again if needed.

Abstraction

Abstraction concept only allowing to display the important information and hide the implementation details from user.

Achieve by:

- Abstract Class
- Abstract Method

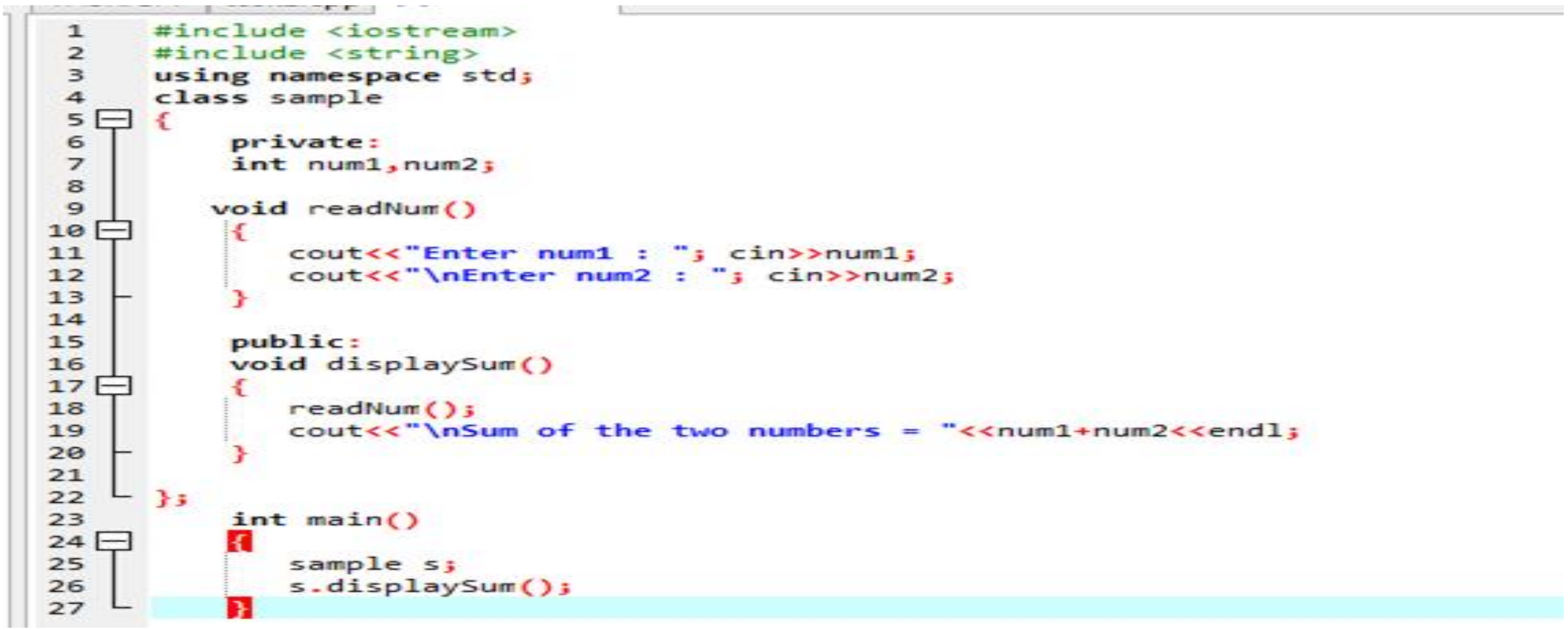
Real-life Example

- We can take a real-life example of an Air conditioner (AC).
- We have a remote control in order to control the various AC functions like start, stop, increase/decrease the temperature, control humidity, etc.
- We can control these functions just with a clock of a button but internally there is a complex logic that is implemented to perform these functions.

Data Abstraction (Example)

Real Life Example of Abstraction

Example-Data Abstraction



```
#include <iostream>
#include <string>
using namespace std;
class sample
{
	private:
	int num1,num2;

      void readNum()
	{
	    cout<<"Enter num1 : "; cin>>num1;
	    cout<<"\nEnter num2 : "; cin>>num2;
	}

	public:
	void displaySum()
	{
	    readNum();
	    cout<<"\nSum of the two numbers = "<<num1+num2<<endl;
	}

};
	int main()
	{
	    sample s;
	    s.displaySum();
	}
```

Advantages of Information Hiding

- Simplifies the model by hiding implementation details
- Prevents accidental access
- Prevents illegal access or manipulation

Encapsulation

Encapsulation refers to binding the data and code that works on together in single unit.

For example: we have a class named as car and we have all the member functions and member variables wrapped inside this single unit and this binding of data is known as encapsulation.

Access Modifier:

- Public
- Private
- Protected

https://www.phptutorial.net/php-oop/php-polymorphism/

Polymorphism

Polymorphism refers to ability to exist in multiple form. Multiple interface can be given to a single interface

For example: First, define an abstract class named Person that has an abstract method called greet().

Type of Polymorphism:

a. Static / Compile time (Overloading)
b. Dynamic/ Run time (Overriding)

Inheritance

Inheritance is a features of OOP which allow the user to inherit the common properties from other class

For example, Class Vehicle and class bike and Class car

Types of inheritance:

1. Single
2. Multiple
3. Multivel
4. Hierarchical
5. Hybrid

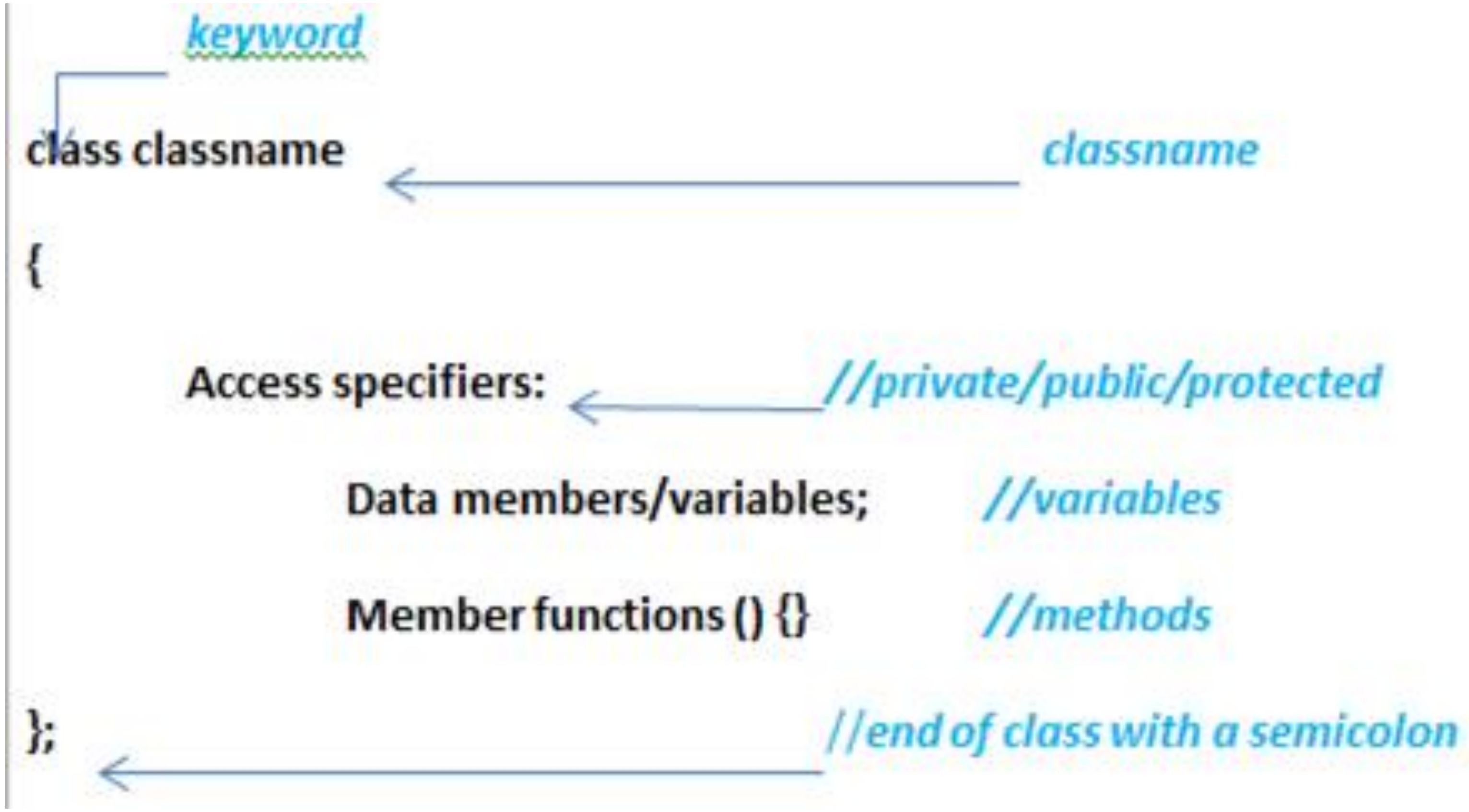
OOP Principles

Encapsulation	Abstraction	Inheritance	Polymorphism
When an object only exposes the selected information.	Hides complex details to reduce complexity.	Entities can inherit attributes from other entities.	Entities can have more than one form.

Class

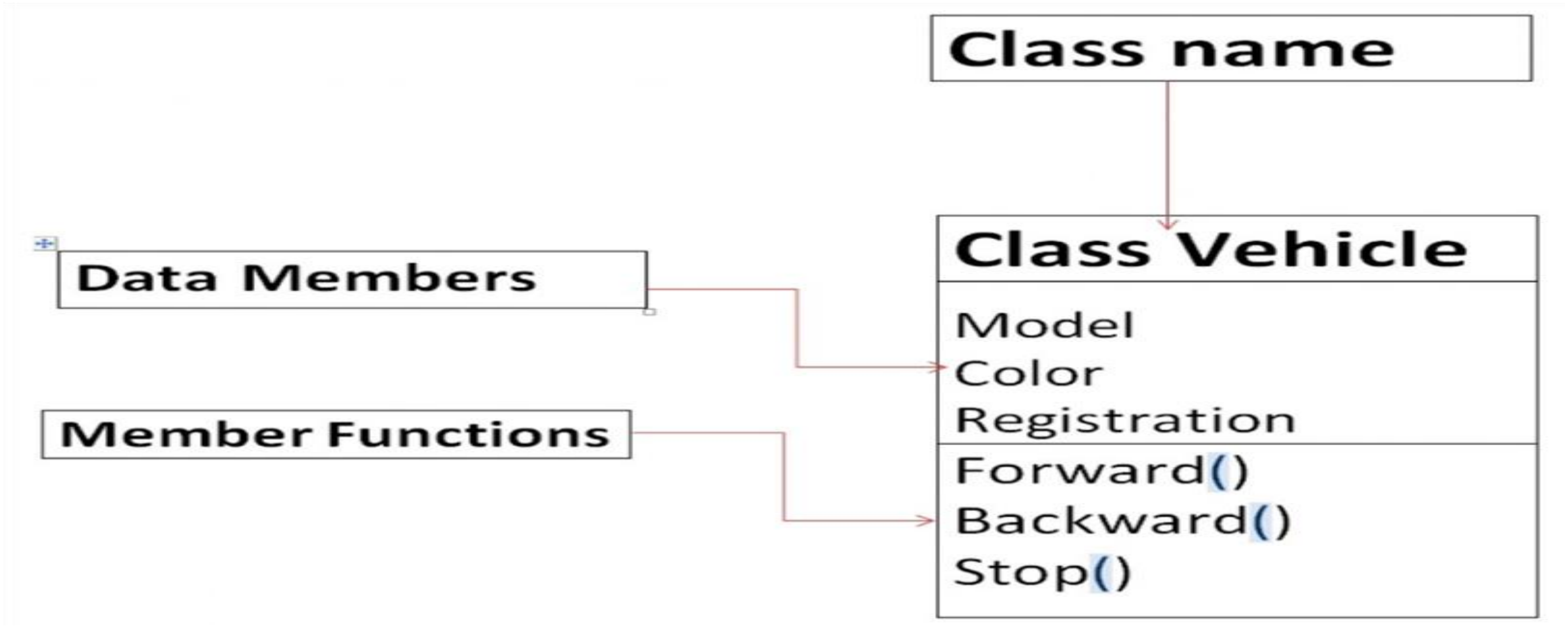
- A class in C++ can be viewed as a blueprint or a skeleton of a particular entity. Class is a user-defined data type. It contains the general information or data for that particular entity and the functions that operate on that entity.
- In C++ syntax, we define a class with keyword “class” followed by the name of the class.
- The class name is followed by the details of the class enclosed in curly braces and is terminated by a semicolon.

Syntax of Class



```
class classname
{
        Access specifiers:              //private/public/protected
                Data members/variables;        //variables
                Member functions () {}          //methods
};                                      //end of class with a semicolon
```

Example-Class



What is an Object?

An object is:

- It can be anything for which we want to save Information

- Something tangible (Ali, Car)

- Something that can be captured intellectually (Time, date)

Technical Definition:

- “An Object is an Instance of a class”

- “An Object is the implementation of a class”

An object has:

- State / attributes / properties / data
- Well-defined behavior / methods / functions
- Unique identity

Ali as an object

Attributes:

- Name
- age

Behavior (operations)

- Walks
- Eats

Identity

- His name

Car as an Object

State (attributes)
- Color
- Model

Behavior (operations)
- Accelerate
- Start Car
- Change Gear

Identity
- Its registration number

Board marker as an object

Attributes?

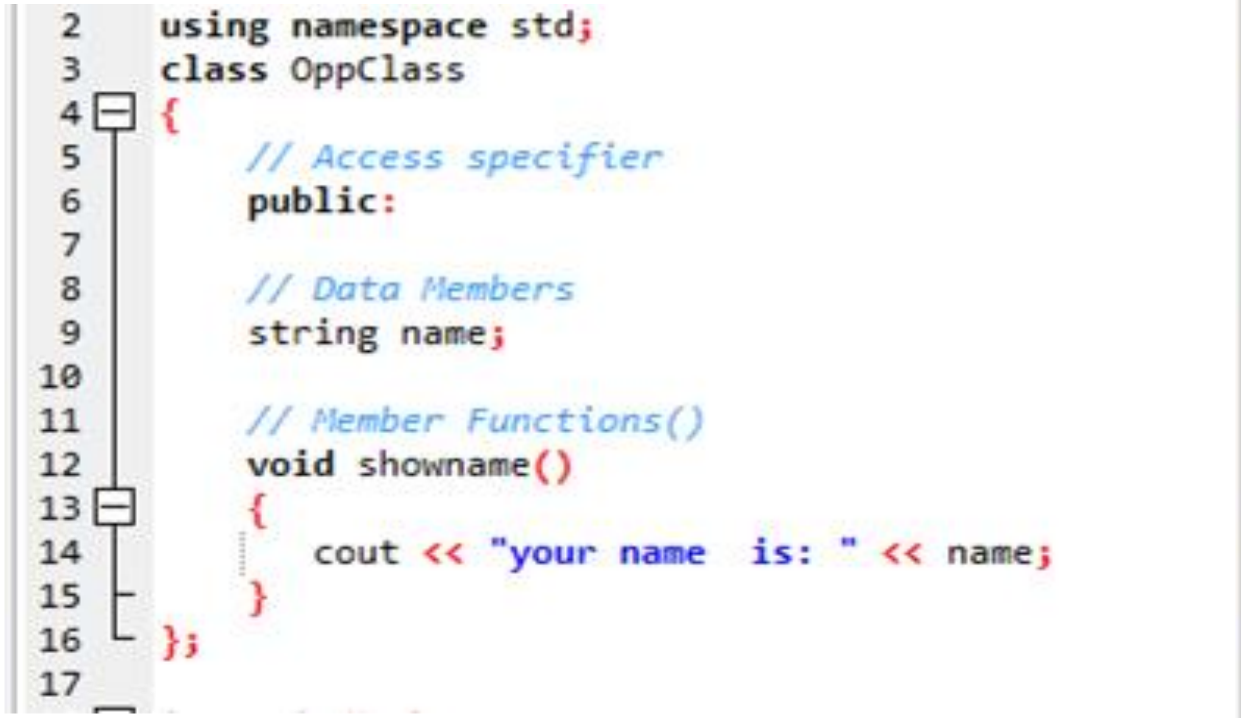
Behavior?

Objects

- In order to use the class functionality, we need to instantiate the class to create an object.
- An object is an instance of a class. In simple words, we can say that **an object is a variable of type class**.
  - **classname object name;**

- Once the object is created, it can be used to access the data members and functions of that class.

Syntax of Class and Object

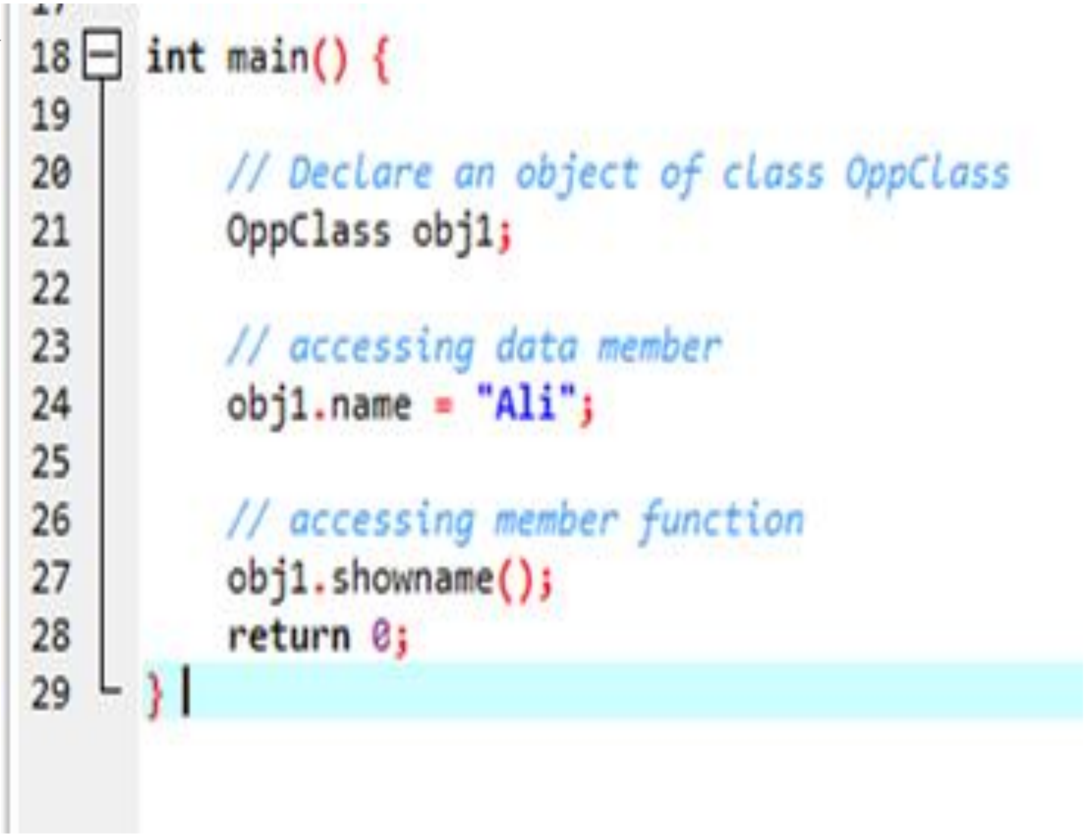


```
using namespace std;
class OppClass
{
    // Access specifier
    public:

    // Data Members
    string name;

    // Member Functions()
    void showname()
    {
        cout << "your name  is: " << name;
    }
};

```



```
int main() {

    // Declare an object of class OppClass
    OppClass obj1;

    // accessing data member
    obj1.name = "Ali";

    // accessing member function
    obj1.showname();
    return 0;
}
```

Some Examples

SHOES

Female sandal

Men sandal

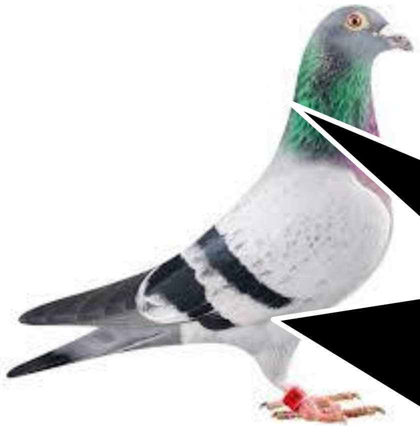
Formal

Pumpies

Loafers

Humans

FISH



BIRD

Animal

Discussion

- Object belongs to a group.
- Which similar.
- Have some common attributes.
- Have some common behaviors.
- We can categorized objects on some basic features …. ?

Exercise

Are there any ‘objects’ in this classroom?

Exercise

Are there any ‘objects’ in this classroom

- **Chair**
- **Multimedia**
- **Student**
- **Teacher**

Exercise

Possible Objects in this university?

Class

- Collection of Similar object.

- The objects that share some common features.

- It is the design of an object.

- It is a detail of an object.

- It tell us what an object contains in it.

Technical Definition:

“ A class is blueprint of an object”

Summarize

A class :

- It’s a blue print .
- It’s a design or template.

An Object:

- Its an instance of a class.
- Implementation of a class.

NOTE: Classes are invisible, object are visible

Class

Accelerator Pedal

Brake Pedal

Steering Wheel

User-friendly “interfaces” to control the car

Their mechanism is *housed* inside the engineering drawings or blueprints

Class

Suppose you want to drive a car and make it go faster by pressing down on its accelerator pedal.

What must happen before you can do this?

Class

before you can drive a car, someone has to *design* it and *build* it.

A car typically begins as engineering drawings

Unfortunately, you cannot drive the engineering drawings of a car—before you can

drive a car, it must be built from the engineering drawings that describe it

A completed car will have an actual accelerator pedal to make the car go faster. But even that's not

enough—the car will not accelerate on its own, so the driver must press the accelerator

pedal to tell the car to go faster.

Now let’s use our car example to introduce the key object-oriented programming concepts

Performing a task in a program requires a function

The function hides from its user the complex tasks that it performs, just as the accelerator

pedal of a car hides from the driver the complex mechanisms of making the car go

Faster

In C++, we begin by creating a program unit called a class to house a function, just

as a car’s engineering drawings house the design of an accelerator pedal.

Class

Just as you cannot drive an engineering drawing of a car, you cannot “drive” a class.

Just as someone has to build a car from its engineering drawings before you can actually drive the car, *you must create an object of a class before you can get a program to perform the tasks the class describes*.

Note also that just as *many* cars can be built from the same engineering drawing, *many* objects can be built from the same class