Lab No. 01: Introduction to OOP

12.1 Objectives

- To Know the importance of OOP
- Implementation of OOP Concepts
- Building Blocks of OOP
- Features of OOP

12.2 Software

- Dev ++
- VS Code

12.3 Theory

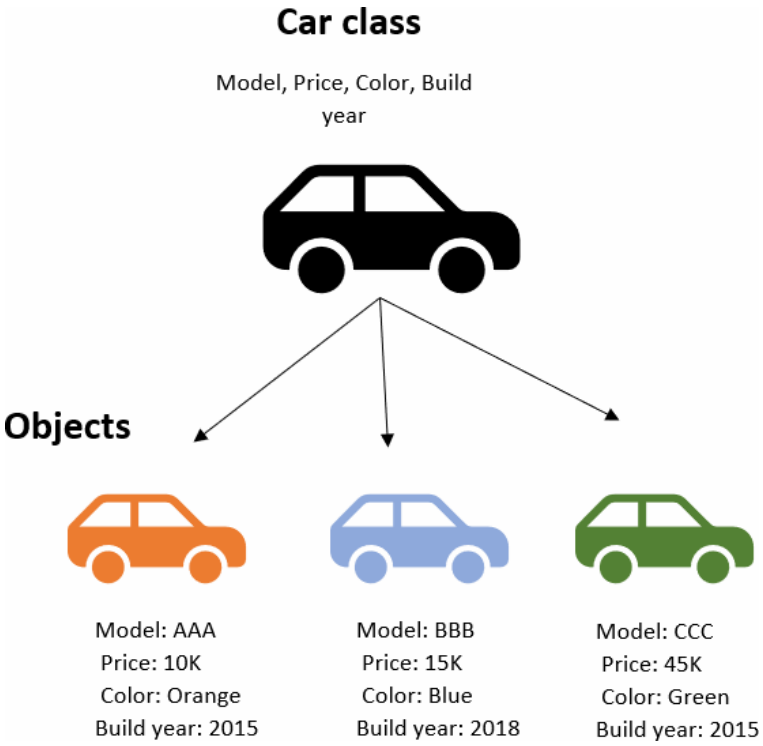
- OOP is a fundamental programming paradigm
- In this type of language, programs are divided into objects.
- Prime focus is on the data that is being operated and not on the functions or procedures.
- Data is hidden and cannot be accessed by external functions.
- Program structure follows "Bottom UP Approach"
- The focus of OOP languages is not on structure, but on modeling data.

The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

- OOP is a programming paradigm 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.
- A programmer designs a software program by organizing related pieces of information and behaviors together into a template called a class
- The entire software program runs by having multiple objects interact with objects to create the larger program
- Examples of OOP languages include C++, Visual Basic.NET and Java.

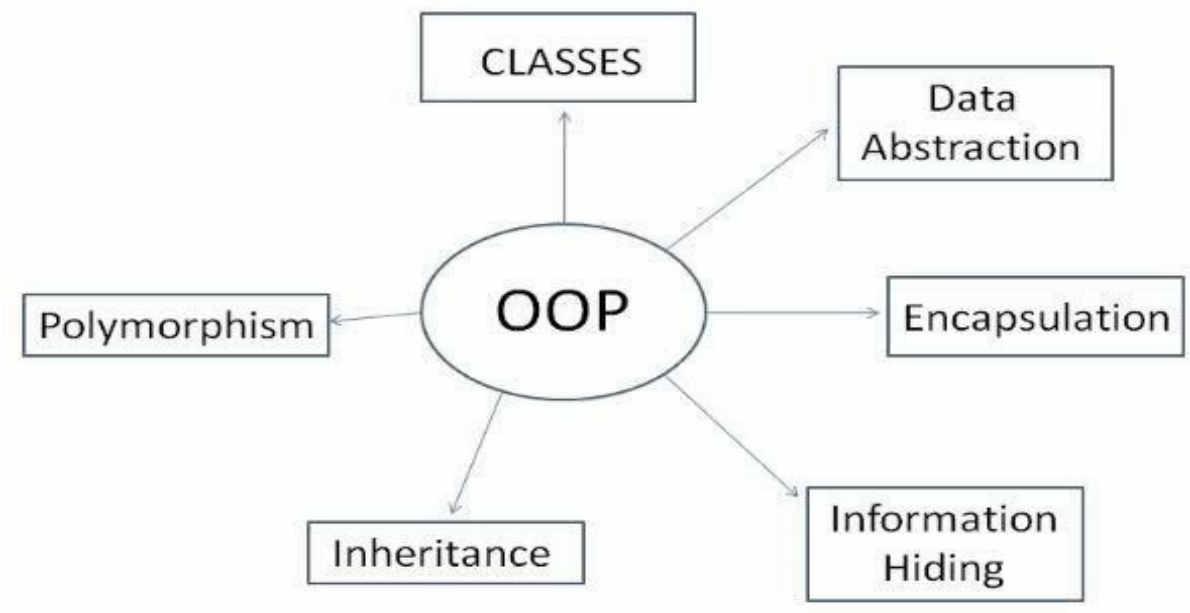

Building Blocks of OOPS
Class
Object
Attributes
Methods

Example:


Car class
Model, Price, Color, Build year
Objects
Model: AAA
Price: 10K
Color: Orange
Build year: 2015
Model: BBB
Price: 15K
Color: Blue
Build year: 2018
Model: CCC
Price: 45K
Color: Green
Build year: 2015

```cpp
#include <iostream>
using namespace std;

class Car {
public:
    string model;
    int price;
    string color;
    long year;

    void Start() {
        cout << "Car Started!" << endl;
    }

    void carModel() {
        cout << this->model << endl;
    }
};

int main() {
    Car c1;
//    c1.model = "2015";
    c1.Start();

    return 0;
}

```

FEATURES OF OOP:



12.4 Tasks

1. Create a Shape class with attributes Color and ShapeName. Include a method called area() within this class to calculate the area of the shape. Then, call this method from the main class using an object.

2-Modify the Car class to include a function named showDetails().

2. This function should print all information about the car (model, year, price, color).
3. Create a car object in main(), assign values to its attributes, and then call showDetails().

3-Create a class Student with attributes: name, rollNo, marks.

1. Write a function showRecord() that displays the student's information.
2. In main(), create two student objects, assign values, and display their records.

Lab # 01 Marks distribution

	ER1	**ER6**	**ER8**
Task	3 points	3 points	4 points

Lab # 01 Rubric Evaluation Guideline:

#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**
ER 1	**Task Completion**	Minimal or no program functionality was achieved.	Some tasks were completed, but the program has errors or incomplete functionalities.	All tasks were completed, and the program runs without errors.
#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**
ER 6	Program Output	Output is inaccurate or poorly presented.	Output is mostly accurate but may lack labels, captions, or formatting.	Output is clear, accurate, and well presented with labels, captions, and proper formatting.
#	**Qualities & Criteria**	**0 < Poor <= 1.5**	**1.5< Satisfactory <= 3**	**3< Excellent <= 4**
ER 8	Question & Answer	Answers some questions but not confidently or based	Answers most questions confidently and based on lab task knowledge.	Answers all questions confidently and demonstrates a deep

		on lab task knowledge.		understanding of the given lab task.