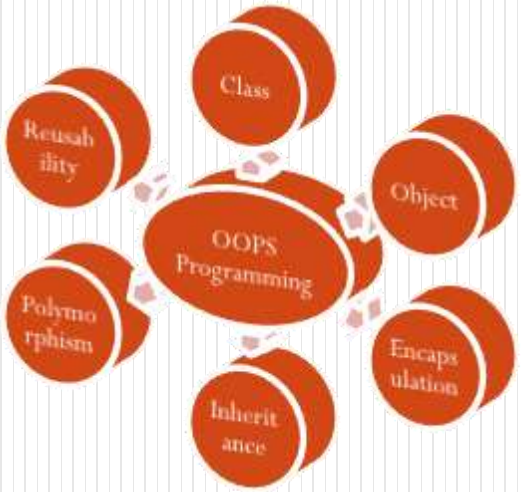
ARDUINO

Object Oriented Programming

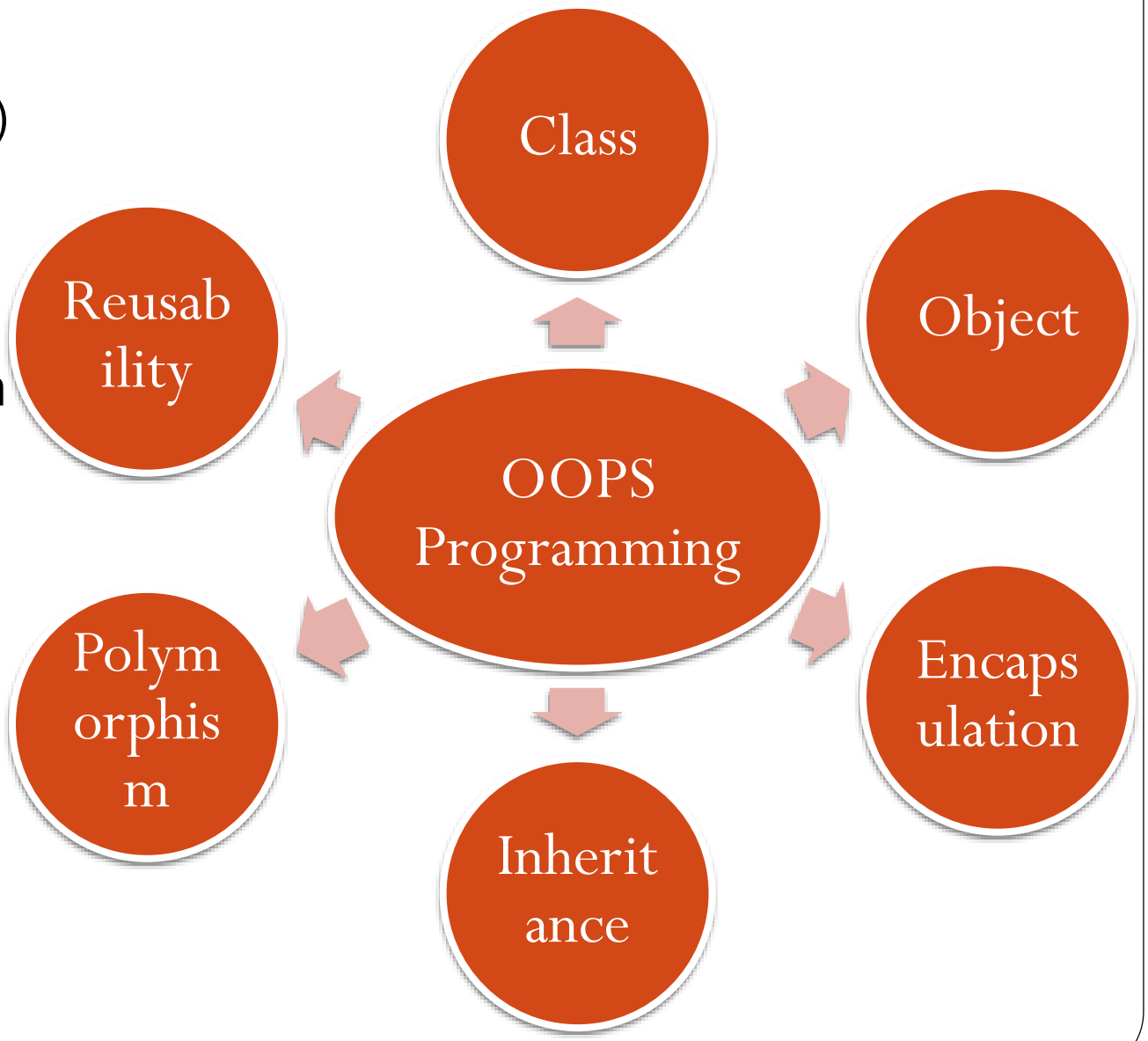
Dr. Sarwan Singh
Deputy Director
NIELIT Chandigarh



Agenda

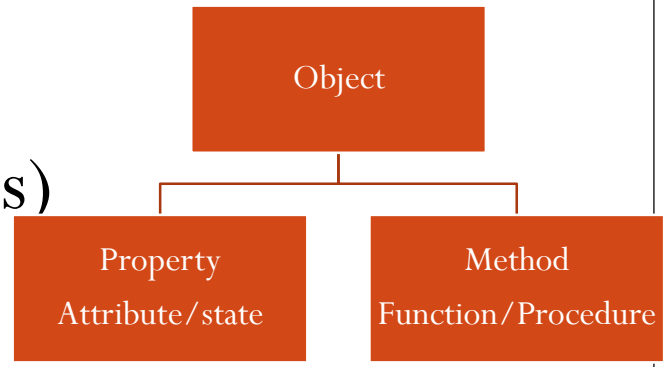
- OOPs-introduction
- Creating class, attributes, methods
- Accessibility – Public, private
- Using objects
- Class in header file
- Creating/accessing Library

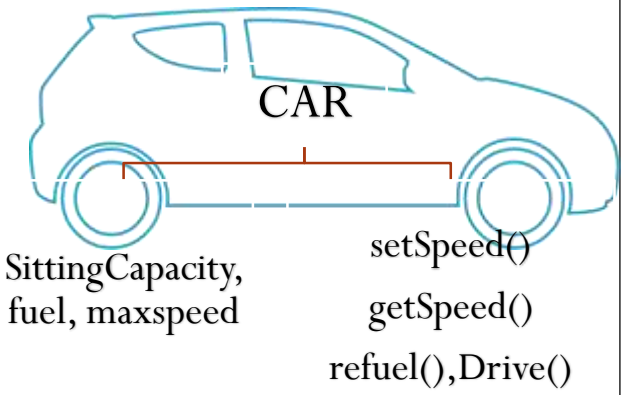
Object-oriented programming (**OOP**) is a **programming language** model organized around **objects** rather than "actions" and data rather than logic.



Terminology

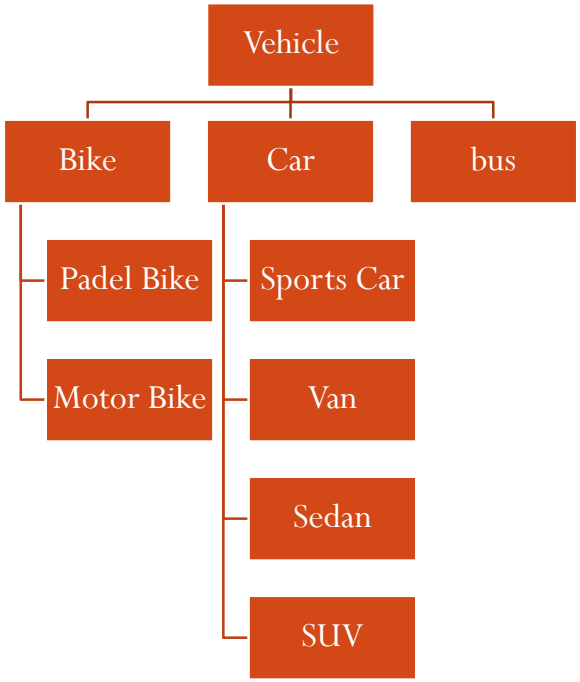
- Object basic unit of object oriented programming.
- Object has both data and function(s) that operate on data
- Class -template for making objects/ blueprint for an object.





Terminology

- Encapsulation is keeping data and functions that work on that data in the same place.
- Inheritance (for code reusability) process of forming a new class(derived clas) from an existing class (base class)
- Polymorphism - ability to use an operator or function in different ways
  - Overloading - existing operator or function is made to operate on new data type



OOPs in Arduino

Creating Flasher class for flashing LEDs

Adding attributes, function, constructor…

Defining a class

```
class Flasher
{  // Class Member Variables
   // These are initialized at startup
   int ledPin; // the number of the LED pin
   long OnTime; // milliseconds of on-time
   long OffTime; // milliseconds of off-time

   // These maintain the current state
   int ledState; // ledState used to set the LED
   unsigned long previousMillis;
   // will store last time LED was updated
};
```


Source : learn.adafruit.com

Adding constructor

```
class Flasher
{  ….Existing code
   // constructor initializes the member variables & state
   public:
   Flasher(int pin, long on, long off)
   {
      ledPin = pin;
      pinMode(ledPin, OUTPUT);
      OnTime = on;
      OffTime = off;
      ledState = LOW;
      previousMillis = 0;
   }
};
```

```
class Flasher
{
// Class Member Variables
// These are initialized at startup
int ledPin; // the number of the LED pin
long OnTime; // milliseconds of on-time
long OffTime; // milliseconds of off-time

// These maintain the current state
int ledState; // ledState used to set the LED
unsigned long previousMillis; // will store last time LED was updated

};
```

Adding member function called Update()

```
void Flasher::Update()
{// check to see if it's time to change the state of the LED
  unsigned long currentMillis = millis();
  if((ledState == HIGH) && (currentMillis - previousMillis >= OnTime))
  {      ledState = LOW; // Turn it off
         previousMillis = currentMillis; // Remember the time
         digitalWrite(ledPin, ledState); // Update the actual LED
  }
  else if ((ledState == LOW) && (currentMillis - previousMillis >= OffTime))
  {      ledState = HIGH; // turn it on
         previousMillis = currentMillis; // Remember the time
         digitalWrite(ledPin, ledState); // Update the actual LED
  }
}
```

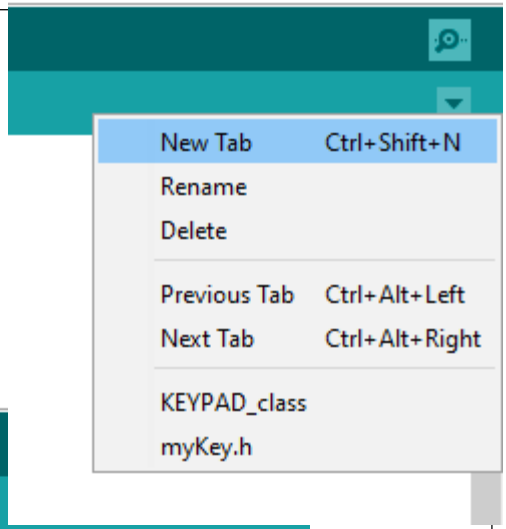
Using Class- creating object

```
Flasher led1(12, 100, 400);
Flasher led2(13, 350, 350);  //Flasher(int pin, long on, long off)
void setup()
{
}

void loop()
{
      led1.Update();
      led2.Update();
}
```

Keypad class

Creating/Using Header file



KEYPAD_class | Arduino 1.6.7

File Edit Sketch Tools Help

KEYPAD_class § myKey.h

```
#include "myKey.h" // include header file
const int rows=4;
const int cols=3;
char keys[rows][cols] = {
  {'1','2','3' },
  {'4','5','6' },
  {'7','8','9' },
  {'*','0','#' }
};
int rowPins[rows] = {6, 7, 8, 9}; //connect to the row pinouts of the keypad
int colPins[cols] = {A0,A1,A2}; //connect to the column pinouts of the keypad

myKeypad mkp(rows,cols,rowPins,colPins, (char *)keys);
```

Using objects

```
//creating mkp object of myKeypad class
myKeypad mkp(rows,cols,rowPins,colPins, (char *)keys);

void setup() {
  Serial.begin(9600);
  Serial.print("Keypad");
  mkp.serialPrint();
  mkp.begin();
}

void loop() {
    char c = mkp.getKeyPressed();
    if (c != NOKEY) Serial.println(c);
```

Library folder of Arduino IDE

This PC › Windows (C:) › Program Files (x86) › Arduino › libraries › myKeypad › src

Name	Date modified	Type	Size
myKeypad.h	02-06-2018 21:13	C/C++ Header	2 KB

This PC › Windows (C:) › Program Files (x86) › Arduino › libraries ›

Name	Date modified	Type
Bridge	17-01-2018 20:46	File folder
Esplora	17-01-2018 20:46	File folder
Ethernet	17-01-2018 20:46	File folder
Firmata	17-01-2018 20:46	File folder
GSM	17-01-2018 20:46	File folder
Keyboard	17-01-2018 20:46	File folder
LiquidCrystal	17-01-2018 20:46	File folder
Mouse	17-01-2018 20:46	File folder
myKeypad	02-06-2018 21:18	File folder
Robot_Control	17-01-2018 20:46	File folder
Robot_Motor	17-01-2018 20:46	File folder
RobotIRremote	17-01-2018 20:46	File folder
SD	17-01-2018 20:46	File folder
Servo	17-01-2018 20:46	File folder
SpacebrewYun	17-01-2018 20:46	File folder
Stepper	17-01-2018 20:46	File folder
Temboo	17-01-2018 20:46	File folder
TFT	17-01-2018 20:46	File folder
WiFi	17-01-2018 20:46	File folder

KEYPAD_UsingLibrary §

```
// folder named: myKeypad  is created inside  C:\Program Files (x86)\Arduino\libraries\
// myKepad.h file is placed inside C:\Program Files (x86)\Arduino\libraries\myKeypad\src

#include <myKeypad.h>
const int rows=4;
const int cols=4; //3;
char keys[rows][cols] = {
  {'7','8','9','d' },
  {'4','5','6','x' },
  {'1','2','3','-' },
  {'c','0','=','+' }
};
int rowPins[rows] = {6, 7, 8, 9}; //connect to the row pinouts of the keypad
int colPins[cols] = {A0,A1,A2,A3}; //connect to the column pinouts of the keypad
//int colPins[cols] = {A0,A1,A2}; //connect to the column pinouts of the keypad

myKeypad mkp(rows,cols,rowPins,colPins, (char *)keys);
```

```
myKeypad mkp(rows,cols,rowPins,colPins, (char *)keys);

void setup() {
  Serial.begin(9600);
  Serial.print("Keypad");
  //mkp.serialPrint();
  mkp.begin();
}
void loop() {
    char c = mkp.getKeyPressed();
    if (c != NOKEY) Serial.println(c);
}
```

Happy Coding

Journey begins from here……