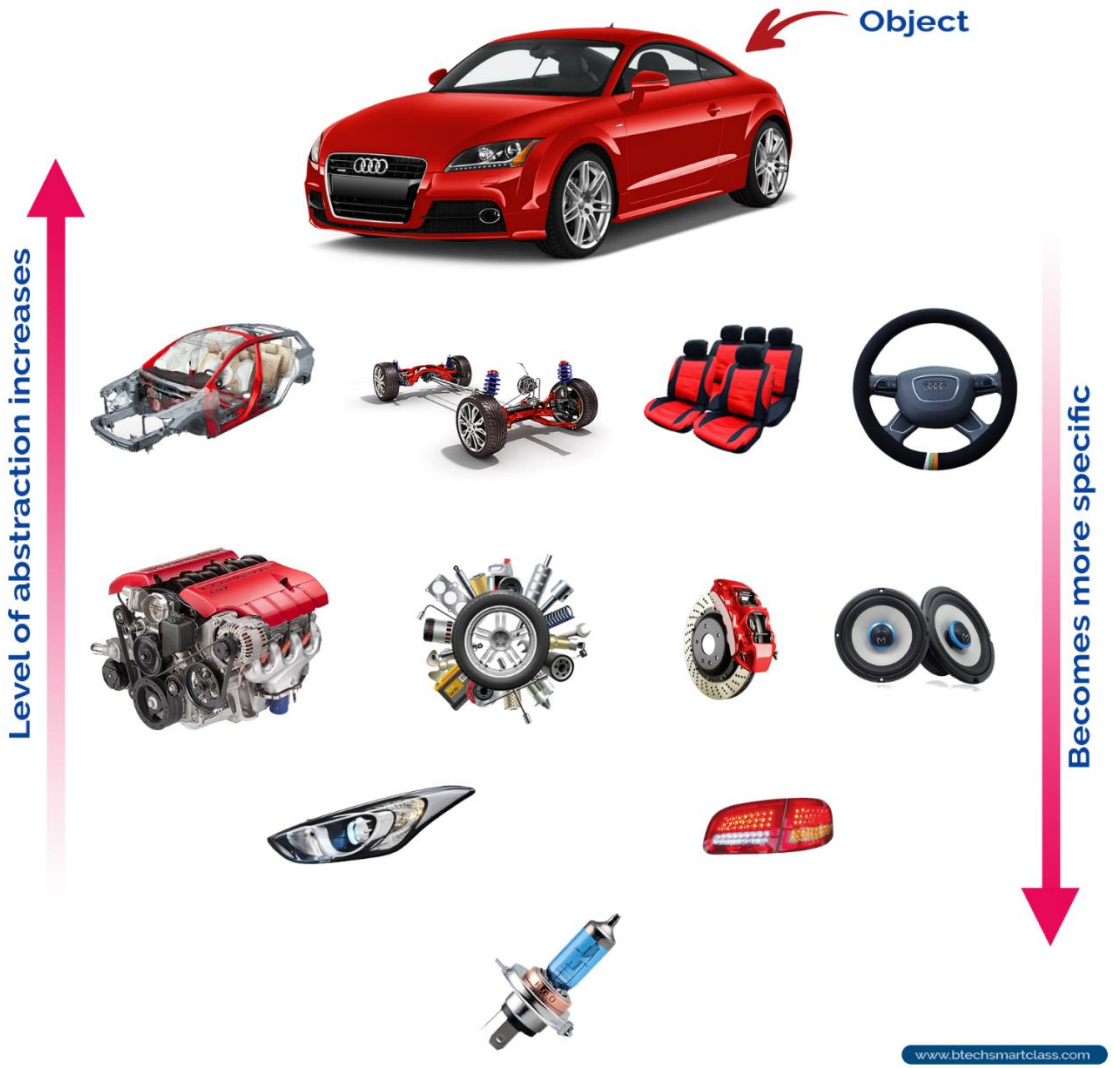
Object Oriented Programming

Abstraction

Abstraction is hiding the internal details and showing only essential functionality.

In the abstraction concept, we do not show the actual implementation to the end user, instead we provide only essential things.

For example, if we want to drive a car, we do not need to know about the internal functionality like how wheel system works? how brake system works? how music system works? etc.



Abstraction

- Abstraction is the "process of representing only essential features". That means Abstraction doesn't show the complexity behind features. Abstraction is used for "Making things more general, simple".
- Example : ATM Machine , Car

Real Life Example of Abstraction

http://theabhinavbaba.blogspot.in

Abstraction

- Abstraction is a design principle.
- Is the process of removing characteristics from something in order to reduce it to a set of essential characteristics.
- Through the process of abstraction, a programmer hides all but the relevant data about a class in order to reduce complexity and increase reusability.

Abstraction

- Abstraction allows programmers to represent complex real world in the simplest manner.
- It is a process of identifying the relevant qualities and behaviors an object should possess, in other word represent the necessary features without representing the back ground details
- You should always use abstraction to ease reusability, and understanding for the design and enable extension.
- When we design the abstract classes, we define the *framework* for later extensions.

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and cost, and function operate on these attributes.
They encapsulate all the essential properties of the object that are to be created.

The **attributes** are some time called ***data members*** because they hold information. The **functions** that operate on these data are sometimes called ***methods or member function***.

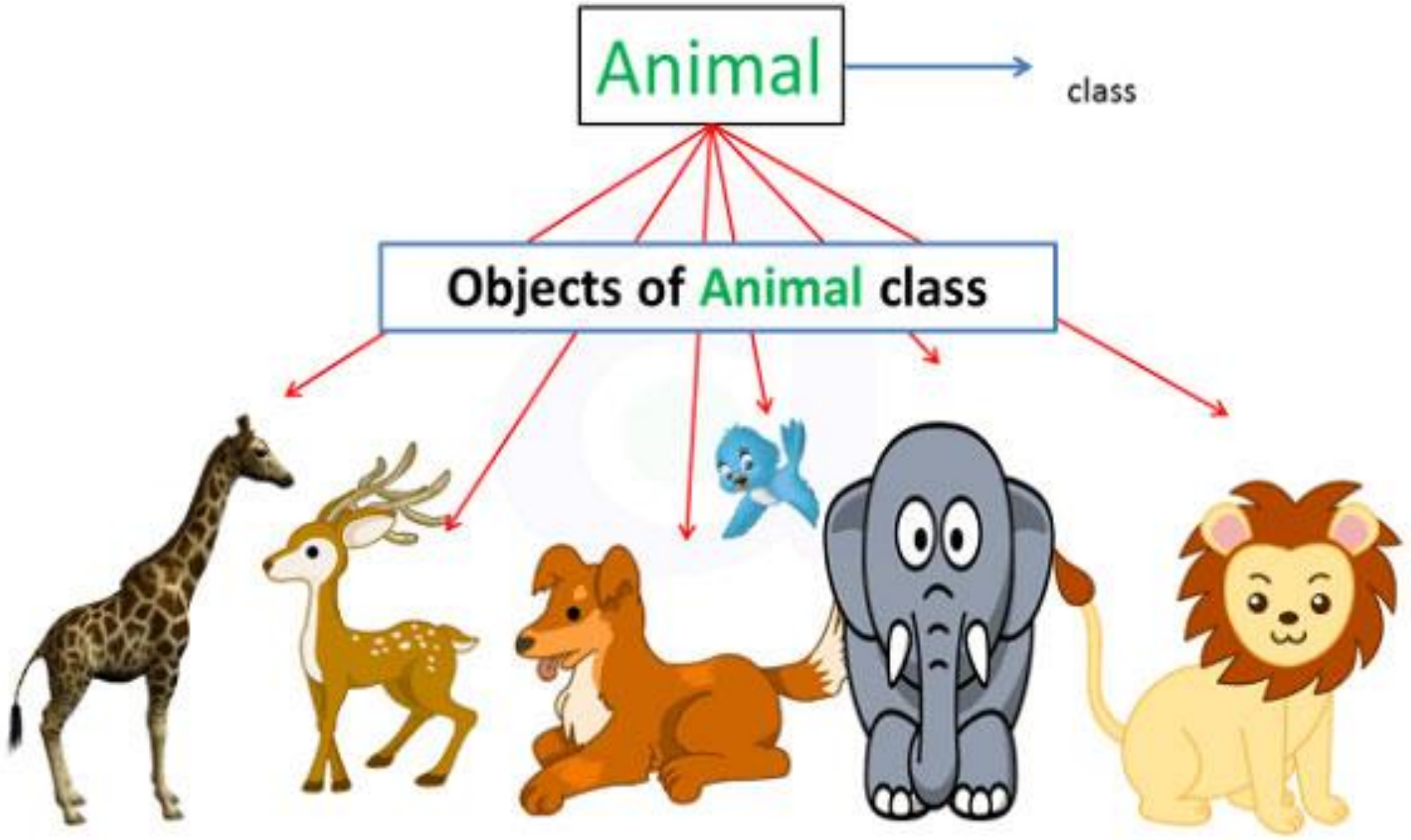
Characteristics/Attributes

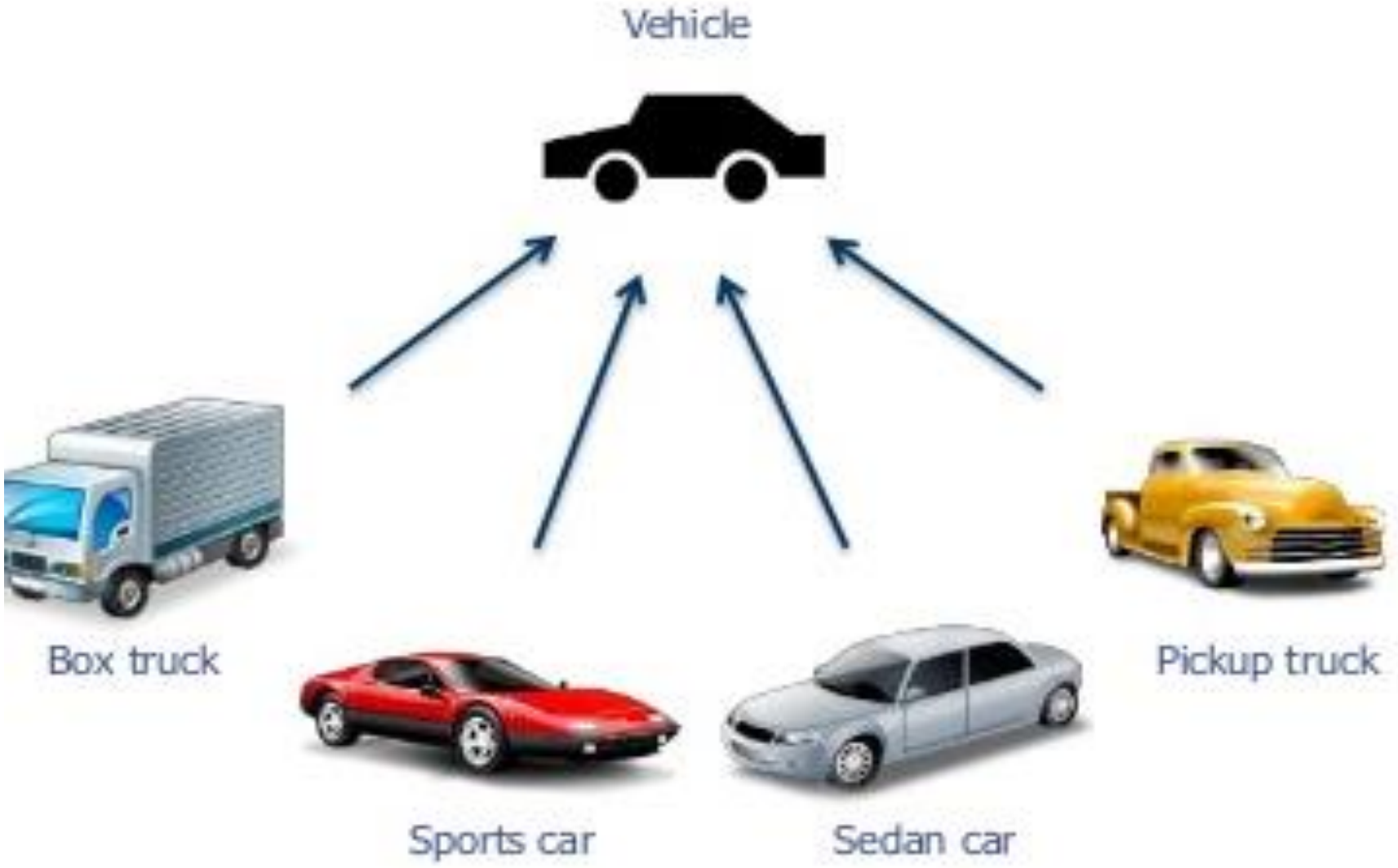
- Make
- Model
- Number of doors
- Engine size

Actions it can perform/Methods

- Accelerate
- Stop
- Brake
- Turn

The wrapping up of data and function into a single unit (called class)


Animal
class
Objects of Animal class


Vehicle
Box truck
Sports car
Sedan car
Pickup truck

Uses of UML to represent Phone
what properties and methods

OOP Lab 1

File - New - Project – Console App

```
using System;

namespace Lab1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Class Object

```
Using system;
class Program1
{
  static void Main(string[] args)
  {
  string.model = "Mustang";
  string.color = "red";
  int year = 1969;

  Console.WriteLine(model);
  Console.WriteLine(color);
  Console.WriteLine(year);
  //----------------------------------------
  Console.WriteLine(“MODEL = ” + model);
  Console.WriteLine(“COLOR = “ + color);
  Console.WriteLine(“YEAR   = “ + year);
  }
}
```

```
Using system;
class Program2
{
   void printinfo()
  {
     string.model = "Mustang";
     string.color = "red";
     int year = 1969;
     Console.WriteLine(“MODEL = ” + model);
     Console.WriteLine(“COLOR = “ + color);
     Console.WriteLine(“YEAR   = “ + year);
   }

static void Main(string[] args)
{
   printinfo();
}
}
```

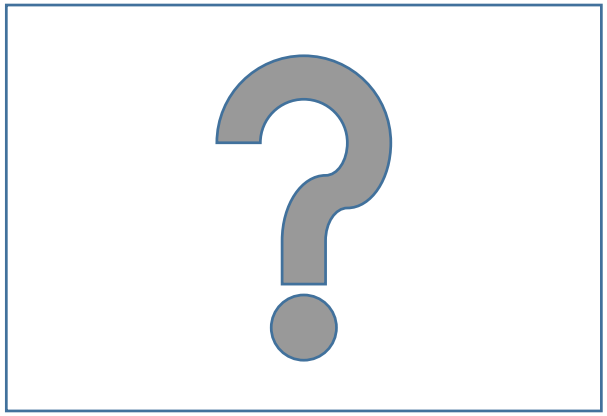
```
Using system;
class Program3
{
   void sum2()
  {
     int n1,n2,s;
      n1=10;
      n2=20;
      s= n1 + n2;
     Console.WriteLine(“SUM = ” + s);
   }

static void Main(string[] args)
{
   printinfo();
}
}
```

```
Using system;
class Program4
{
   int sum2()
  {
     int n1,n2,s;
      n1=10;
      n2=20;
      s= n1 + n2;
     return s;
   }

static void Main(string[] args)
{
   int ss;
   ss=printinfo();
   Console.WriteLine("SUM = " + ss);
}
}
```

QUESTION



Google Classroom :