Sir Syed University of Engineering & Technology, Karachi

Introduction to Object Oriented Programming

Course Code : CS-127

Course Title : Object Oriented Programming

Semester : 2nd

C# - What is OOP?

- OOP stands for Object-Oriented Programming.

- Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

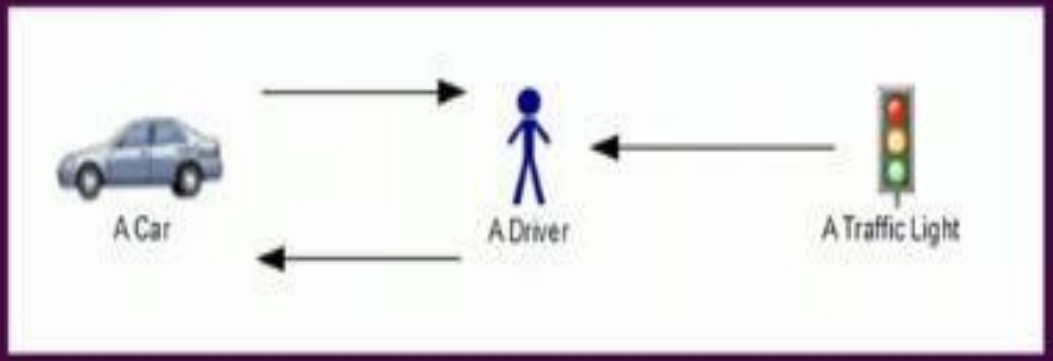
C# - What is OOP?

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute.
- OOP provides a clear structure for the programs.
- OOP makes it possible to create full reusable applications with less code and shorter development time.

Object-oriented programming

An object-oriented program may be considered a collection of interacting objects. Each object is capable of sending and receiving messages, and processing data.



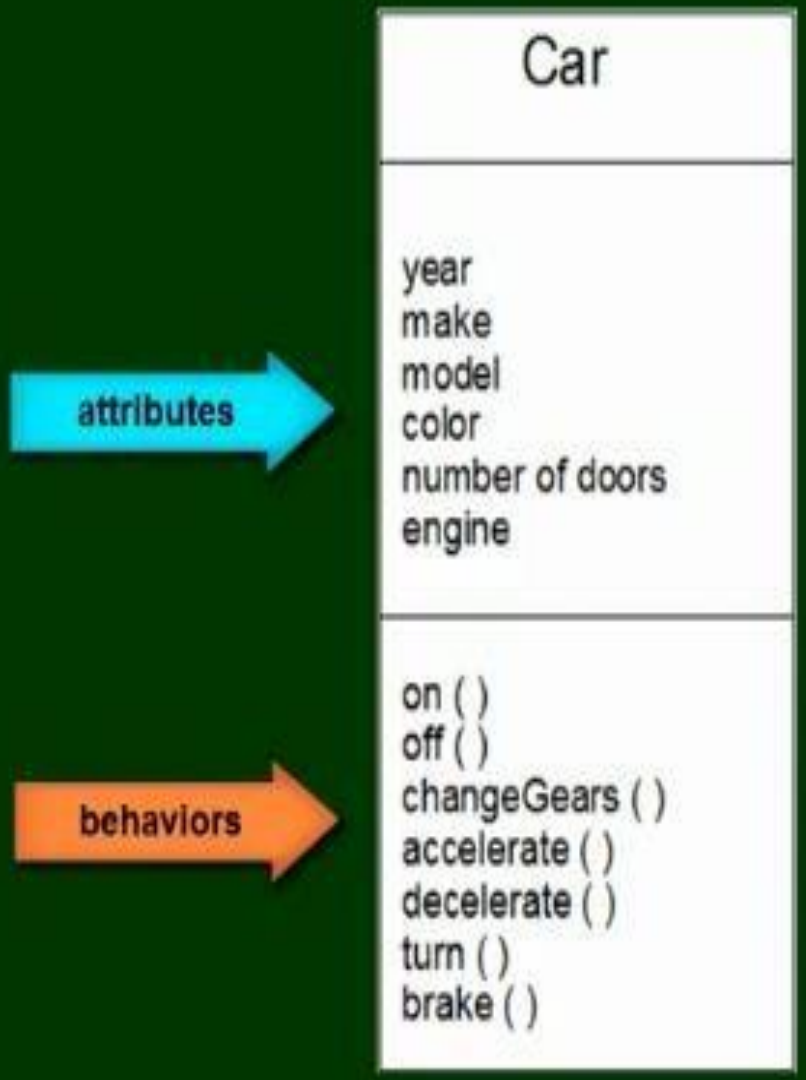
Object-oriented programming concepts

- Class
- Object
- Method
- Message passing
- Inheritance
- Abstraction
- Encapsulation
- Polymorphism

Class

A class defines the characteristics of an object.

Characteristics include:
- attributes
  (fields or properties)
- behaviors
  (methods or operations).



C# Classes and Objects

Classes and objects are the two main aspects of object , oriented programming.
Look at the following illustration to see the difference between class and objects:

class	objects
Fruit	Apple Banana Mango

Another example:

class	objects
Car	Volvo Audi Toyota

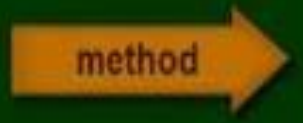
So, a class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the variables and methods from the class.

Method

A method is a behavior of an object.

Within the program, a method usually affects only one particular object.



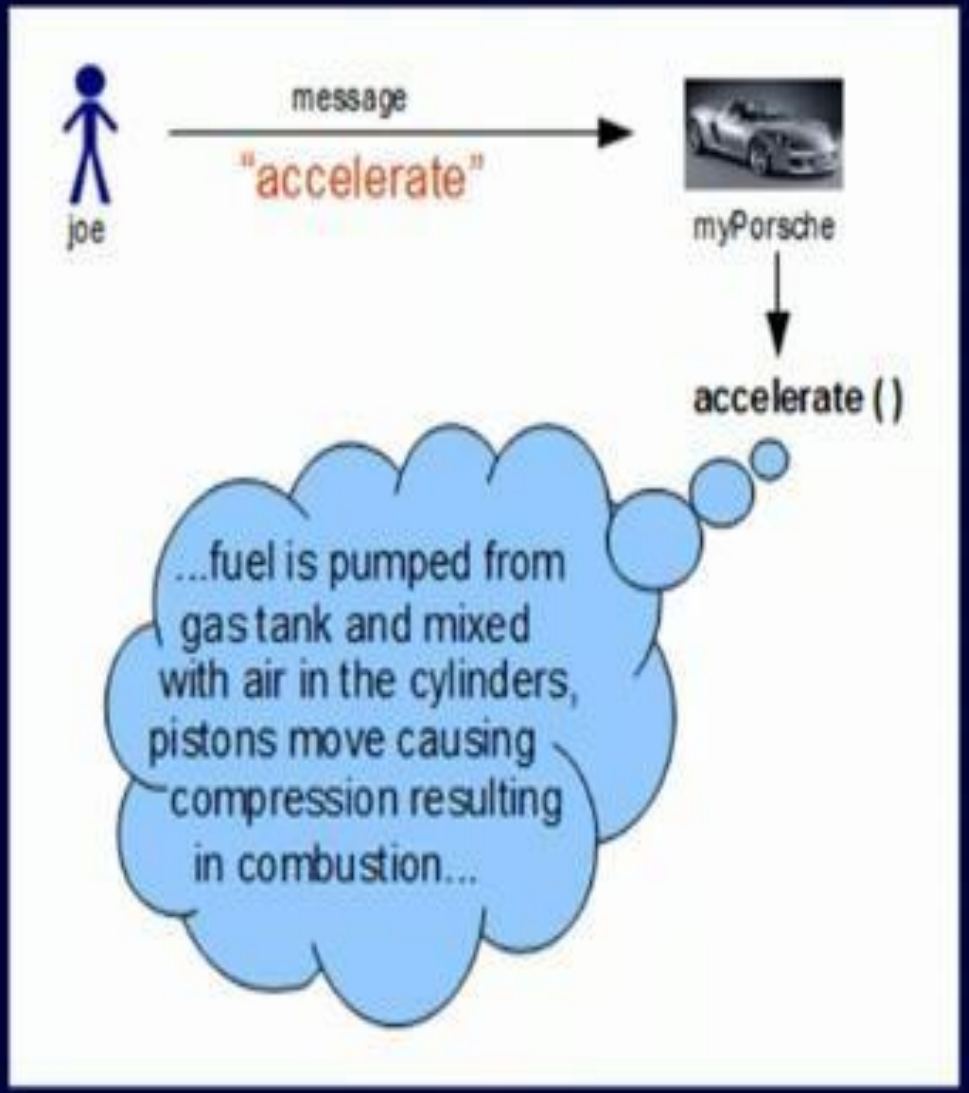
myPorsche : Car

year = 2007
make = "Porsche"
model = "Carrera GT"
color = "Silver"
number of doors = 2
engine = "5.7 liter V10"

on ( )
off ( )
changeGears ( )
accelerate ( )
decelerate ( )
turn ( )
brake ( )

Encapsulation

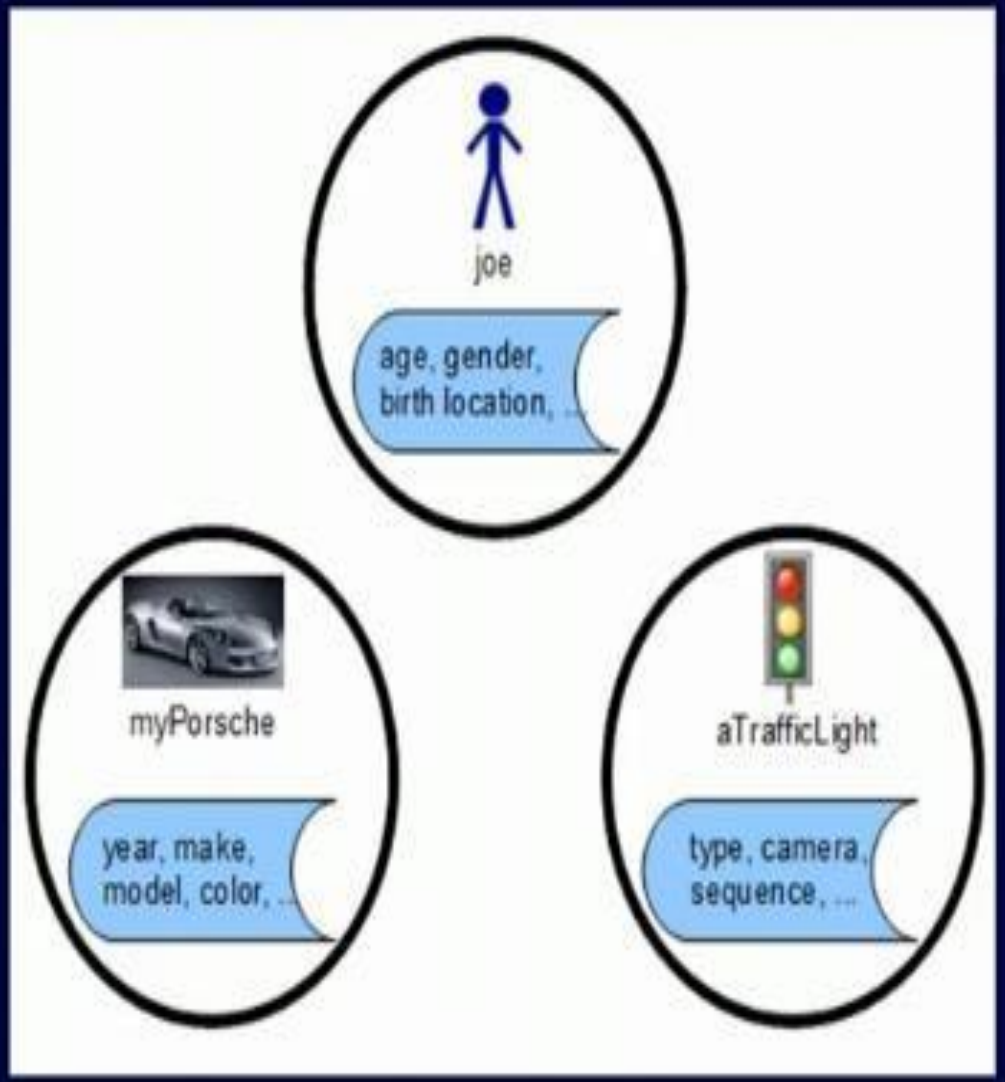
Encapsulation conceals the functional details of a class from objects that send messages to it.



Encapsulation

Encapsulation protects the integrity of an object by preventing users from changing internal data of an object into something invalid.

Encapsulation also reduces system complexity and thus increases robustness, by limiting inter-dependencies between software components.



C# | Encapsulation

Encapsulation is defined as the wrapping up of data under a single unit.

_Technically in encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of own class in which they are declared.

_As in encapsulation, the data in a class is hidden from other classes, so it is also known as data hiding.

Abstraction

In c#, Abstraction is a principle of object ، oriented programming language (OOP) and it is used to hide the implementation details and display only essential features of the object.

In Abstraction, by using access modifiers we can hide the required details of the object and expose only necessary methods and properties through the reference of an object.

In real ، time, the laptop is a perfect example of abstraction in c#. A laptop that consists of many things such as processor, RAM, motherboard, LCD screen, camera, USB ports, battery, speakers, etc. To use it, we just need to know how to operate the laptop by switching it on, we don، t need to know how internally all the parts are working. Here, the laptop is an object which is designed to expose only required features by hiding its implementation details.

Abstraction

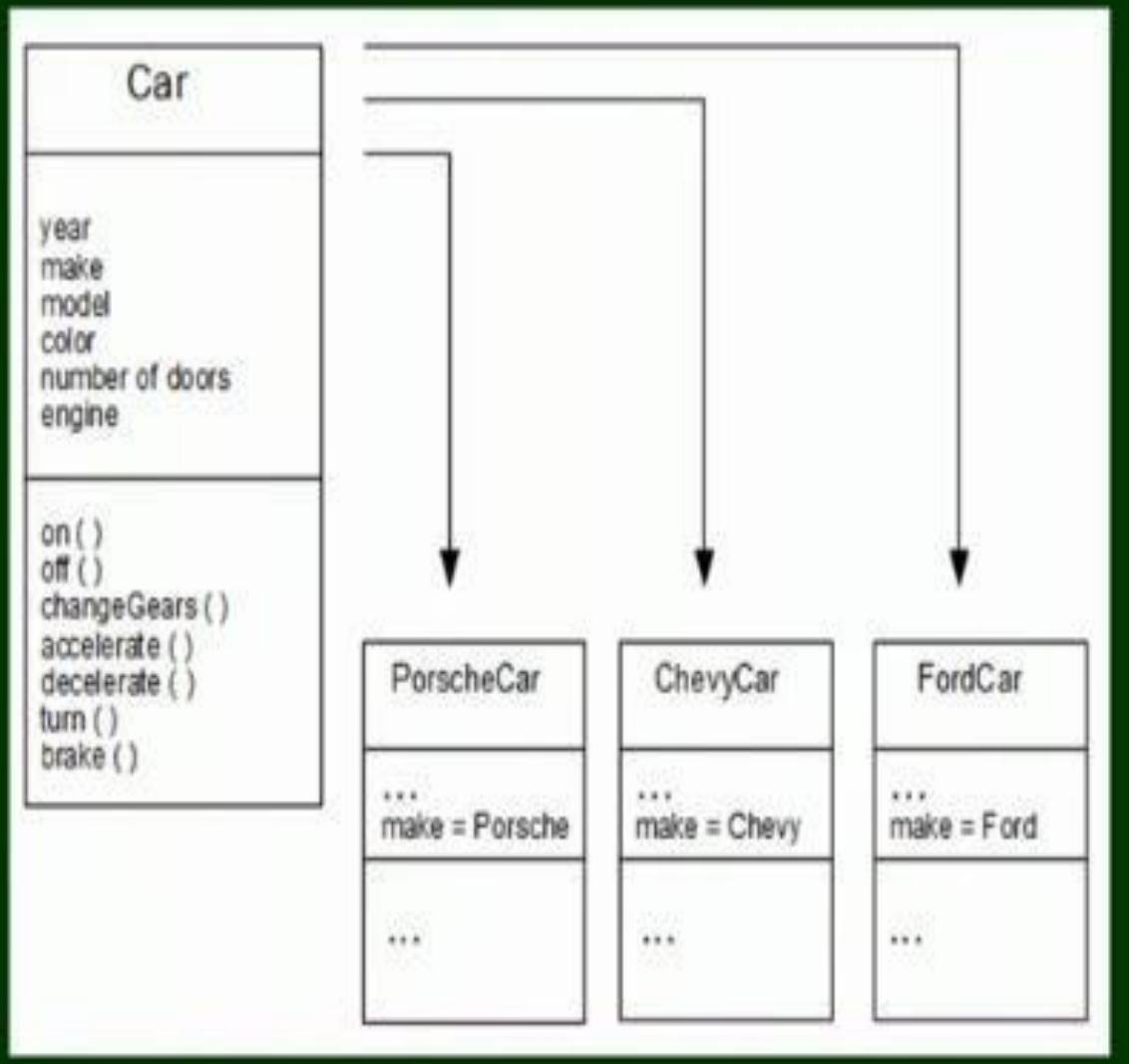
Abstraction and encapsulation are related features in object , oriented programming. Abstraction allows making relevant information visible and encapsulation enables a programmer to implement the desired level of abstraction.

Inheritance

(also known as subclasses)

A subclass is a specialized version of a class, which inherits attributes and behaviors from the parent class.

A subclass can alter its inherited attributes or methods.



Inheritance

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in C# by which one class is allowed to inherit the features(fields and methods) of another class.

Important terminology:

Super Class: The class whose features are inherited is known as super class(or a base class or a parent class).

Sub Class: The class that inherits the other class is known as subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

Reusability: Inheritance supports the concept of reusability, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

How to use inheritance

The symbol used for inheritance is :.

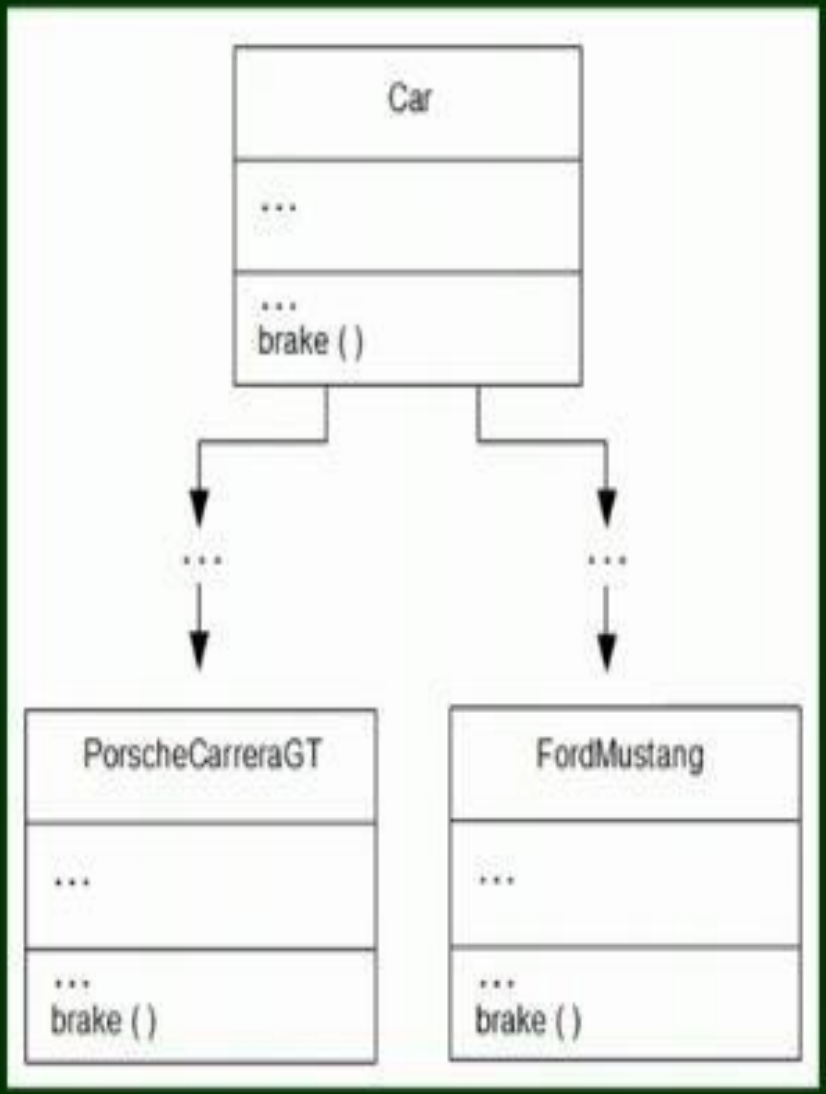
Syntax:

```
class derived-class : base-class
{
   // methods and fields
   .
   .
}
```

Polymorphism

Polymorphism is the ability of one type to appear as, and be used like another type.

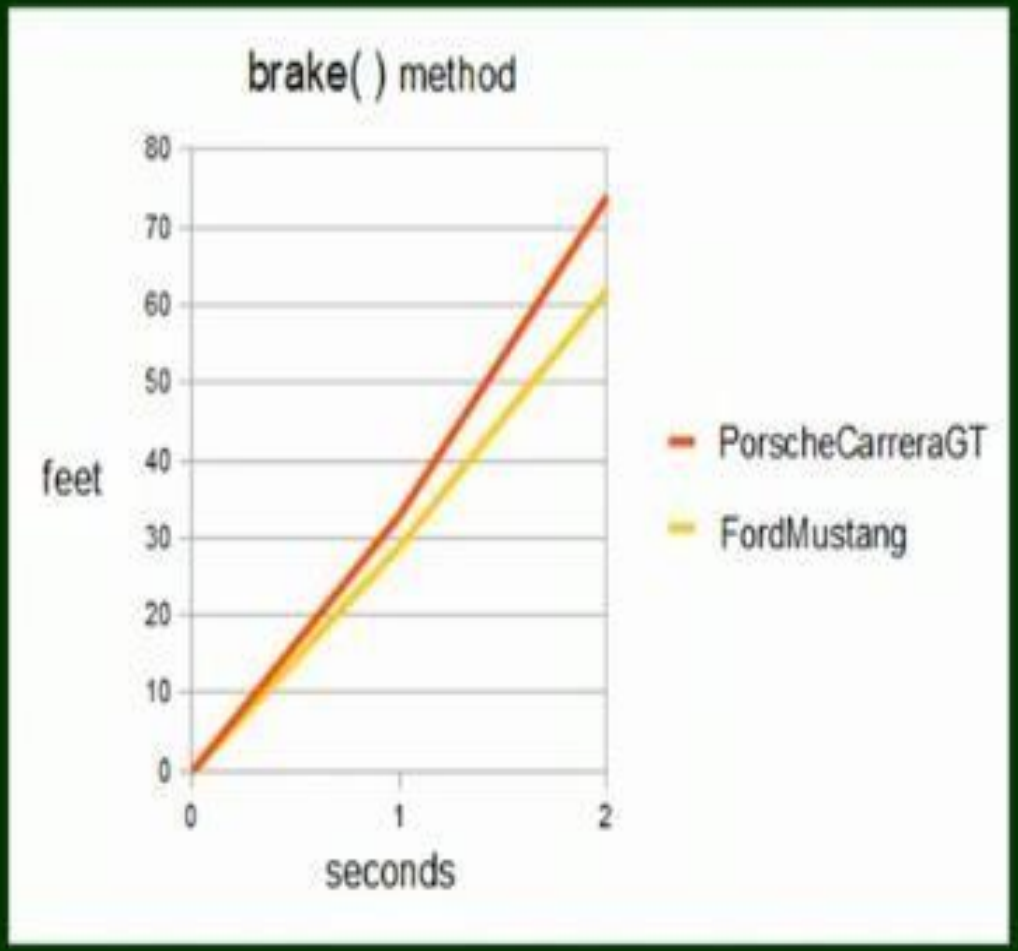
Classes PorscheCarreraGT and FordMustang both inherited a method called brake from a similar parent class.



Polymorphism

Yet the results of executing method "brake" for the two types produces different results.

myPorsche may brake at a rate of 33 feet per second, whereas myMustang may brake at a rate of 29 feet per second.



C# Polymorphism

The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms. It is a greek word. In object , oriented programming, we use 3 main concepts: inheritance, encapsulation and polymorphism.

There are two types of polymorphism in C#: compile time polymorphism and runtime polymorphism.

_Compile time polymorphism is achieved by method overloading and operator overloading in C#. It is also known as static binding or early binding.

_Runtime polymorphism in achieved by method overriding which is also known as dynamic binding or late