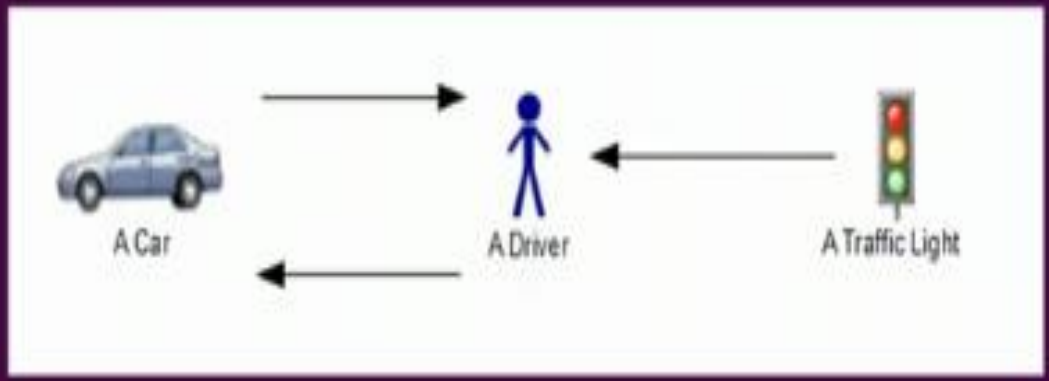
Object-oriented programming

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



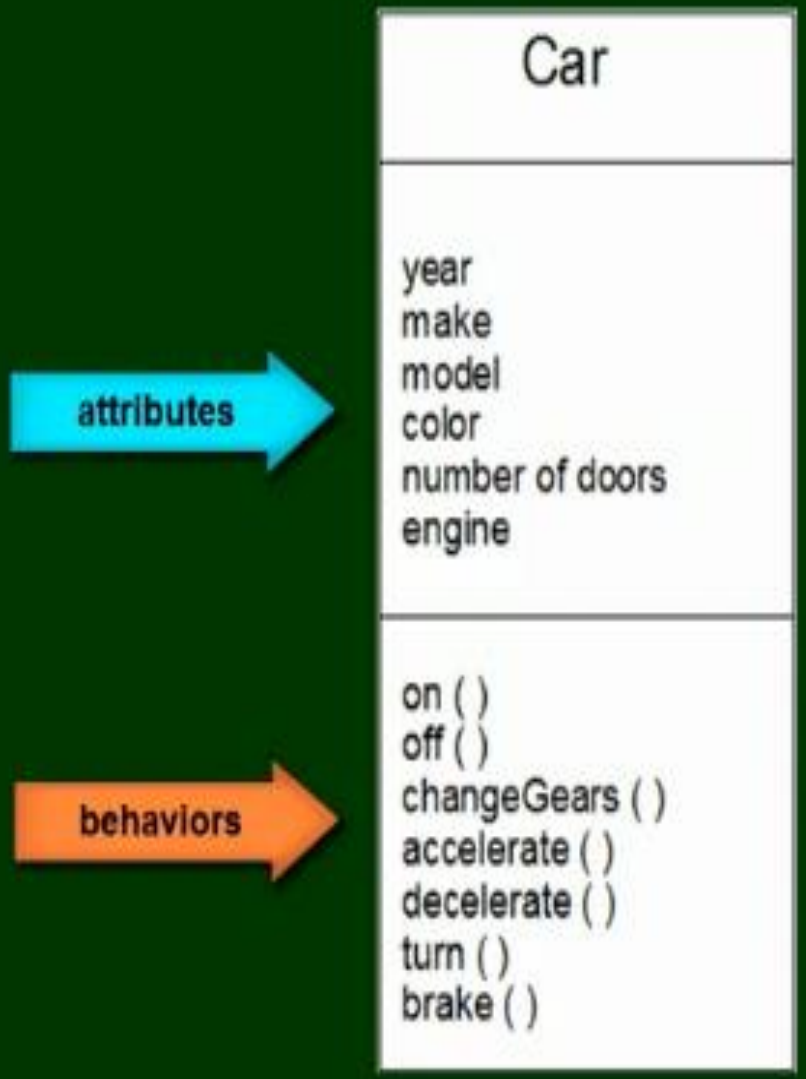
Object-oriented programming concepts

- 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).



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 ( )

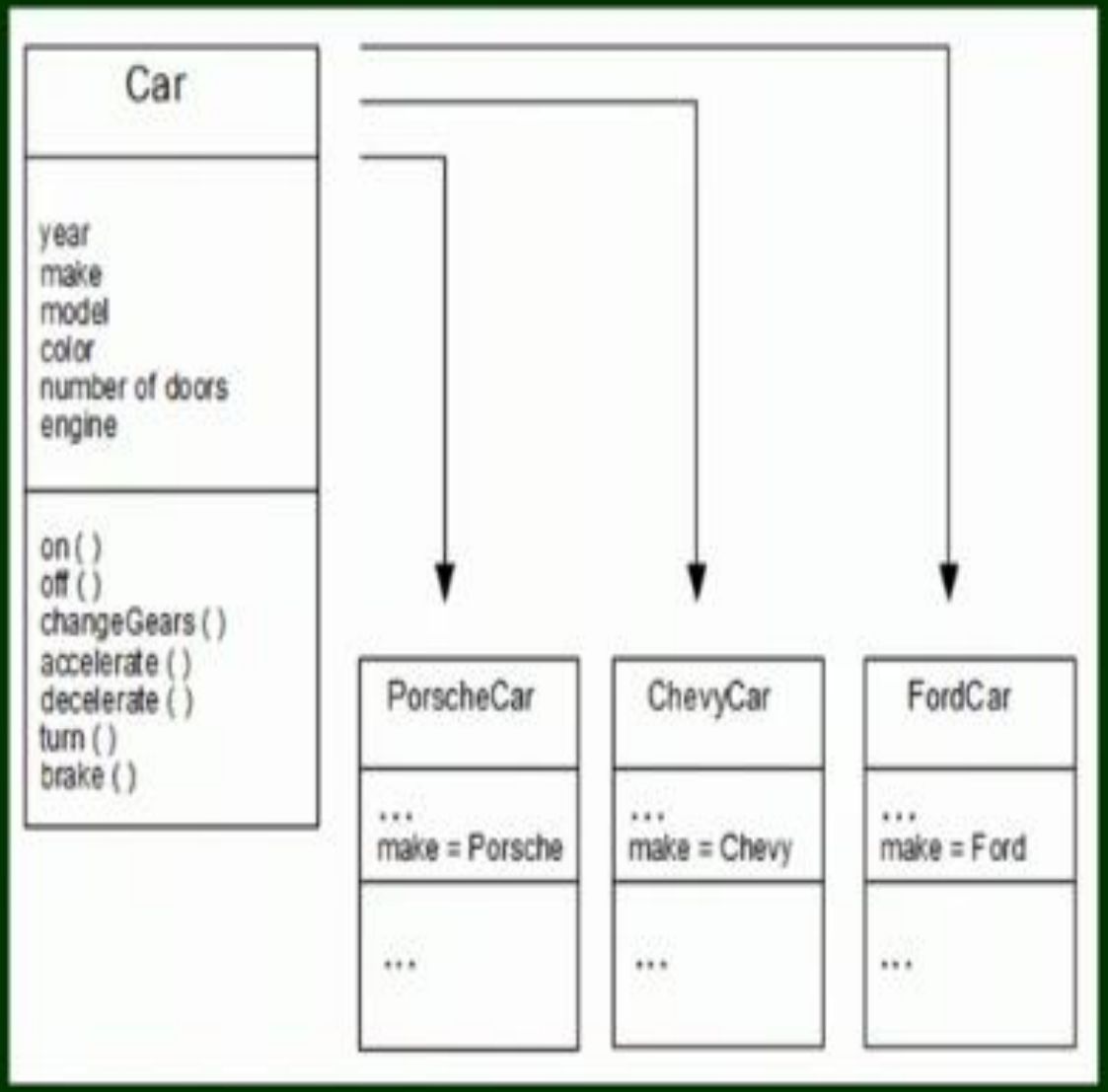
```
//C++ program to demonstrate the use of object and class
#include<iostream>
#include<conio.h>
#include<string>
using namespace std;
class student
{
    private:
      int roll_number;
    public:
      float cgpa;
      int func1()
      {
           roll_number = 2;
           return roll_number;
      }
      float func2()
      {
           cgpa = 3.5;
           return  cgpa;
      }

};
int main()
{
     student s1; //creating object s1 of class type student
     cout << s1.func1();
     cout << s1.func2();
     getch();
     return 0;
}
```

Inheritance

(also known as subclasses)

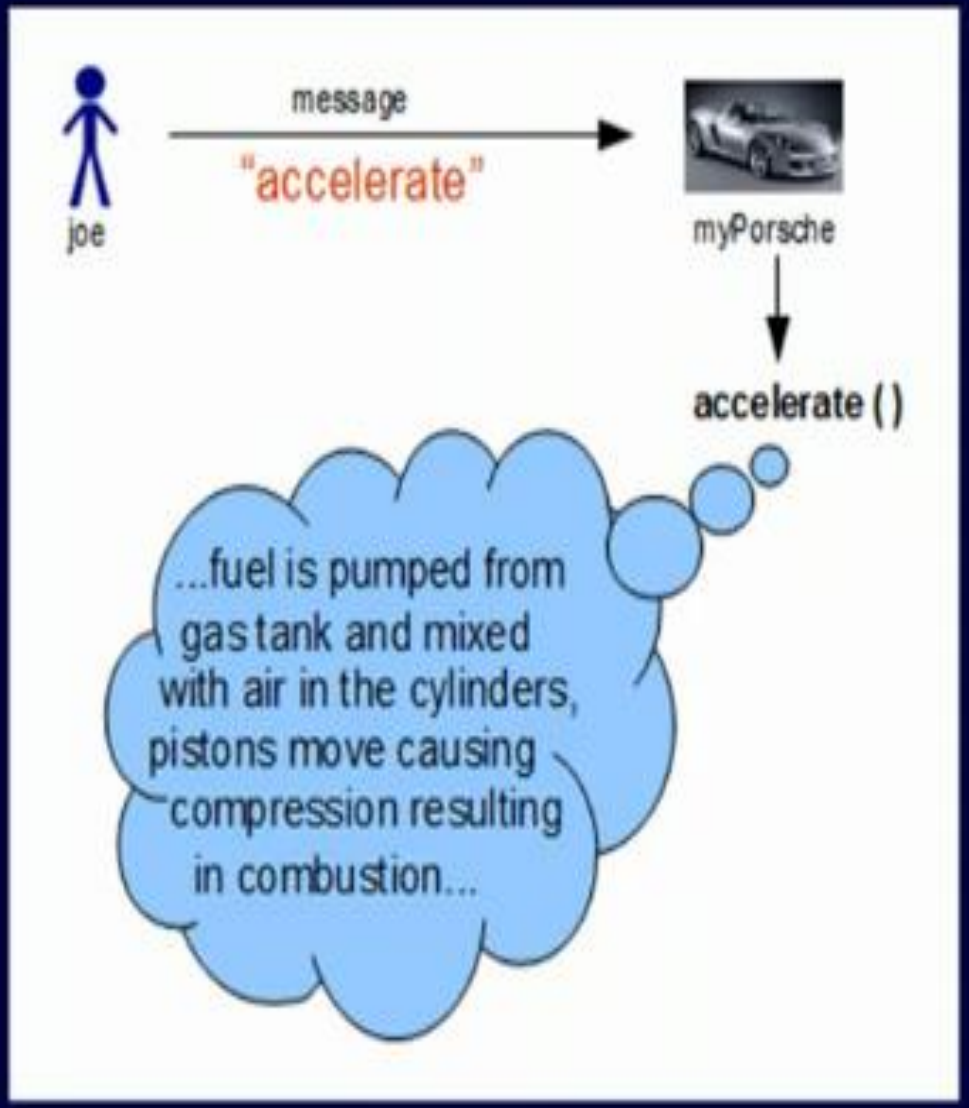
A subclass is a specialized version of a class, which inherits attributes and behaviors from the parent class.

A subclass can alter its inherited attributes or methods.



Encapsulation

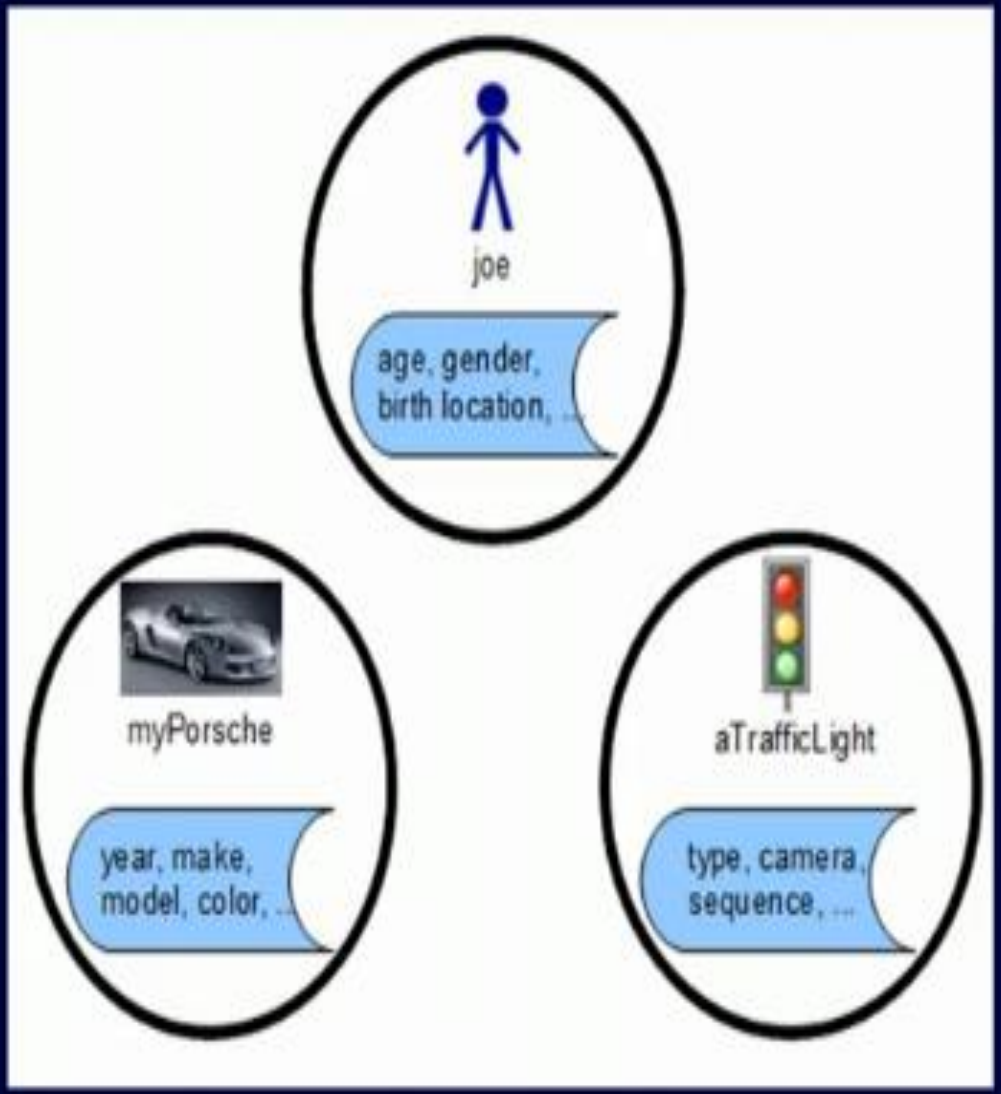
Encapsulation conceals the functional details of a class from objects that send messages to it.



Encapsulation

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

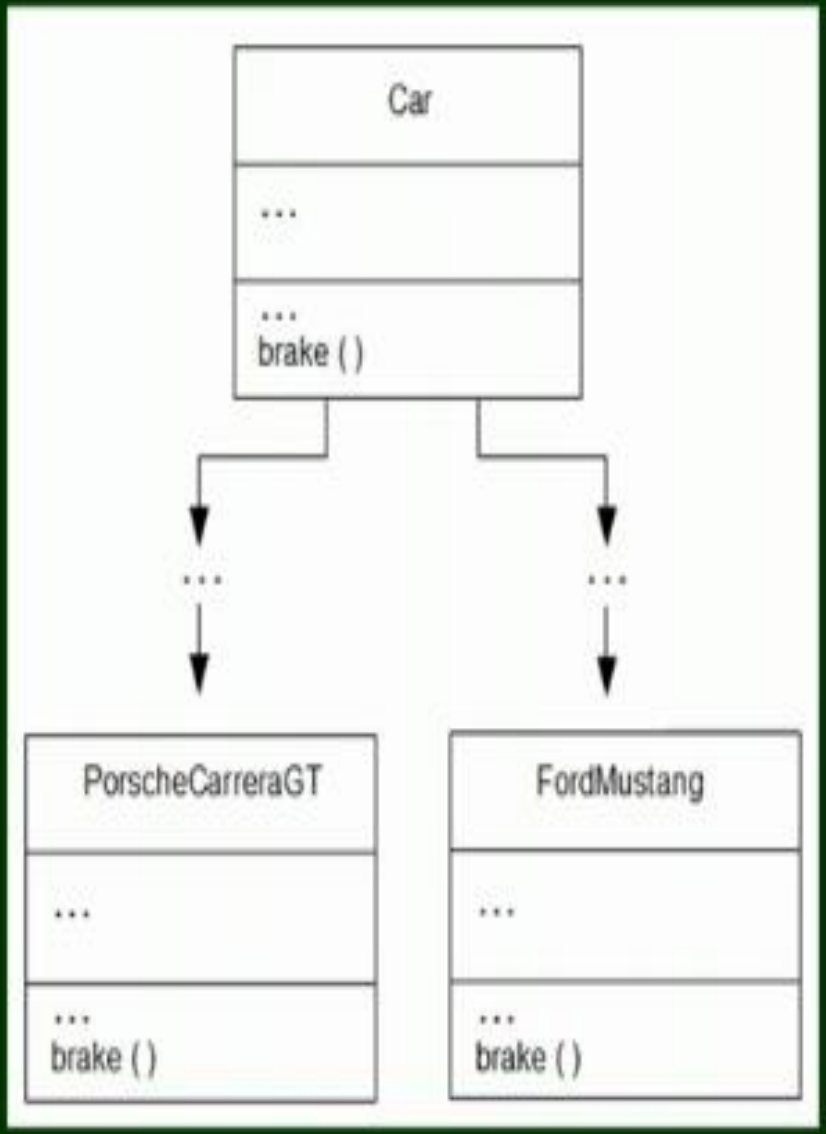
Encapsulation also reduces system complexity and thus increases robustness, by limiting inter-dependencies between software components.



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.