Object Oriented Programing

Lecture 3 class and objects

Arsalan Rahman Mirza

Computer Science Department

Faculty of Science

2022-2023

Outlines

- **OOP definition**
- **Objects and Class**
- **Object attributes**
- **Access modifiers**
- **Constructor**
- **Inheritance**

OOP

- Object Oriented Programming.
- Object → Things, Items, Collection of Data
- Oriented → Toward, aiming,
- Programming → you know what it means !

Object and Class

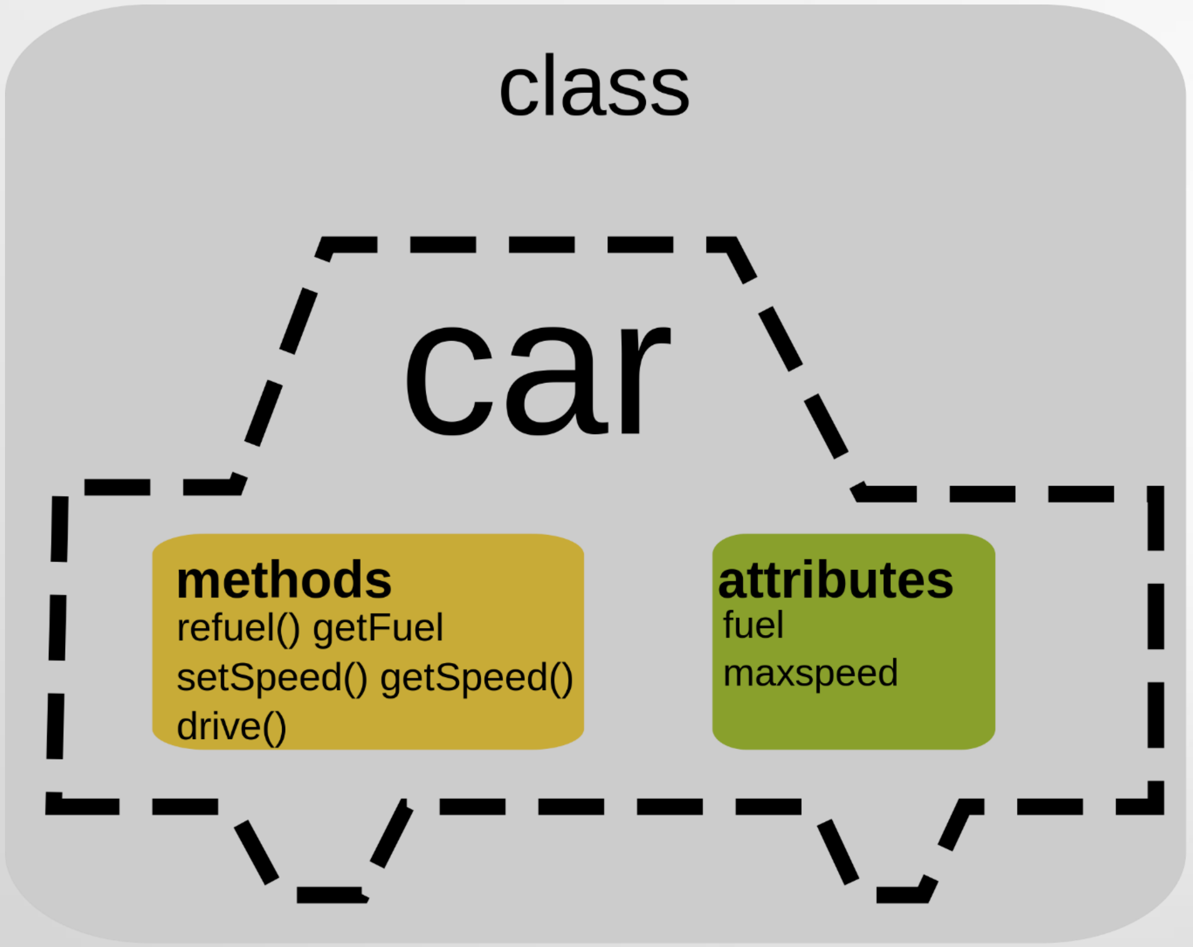
- Objects are basic building blocks of a C# OOP program. An object is a combination of data and methods. The data and the methods are called *members* of an object. In an OOP program, we create objects. These objects communicate together through methods. Each object can receive messages, send messages and process data.
- There are two steps in creating an object. First, we define a class. A *class* is a template for an object. It is a blueprint which describes the state and behavior that the objects of the class all share. A class can be used to create many objects. Objects created at runtime from a class are called *instances* of that particular class.

How to Write Classes

```
class classname
{
   members;
}
```

There are three type of Members:

1. Variables (Memory)
2. Methods (Actions)
3. Properties (Fields)



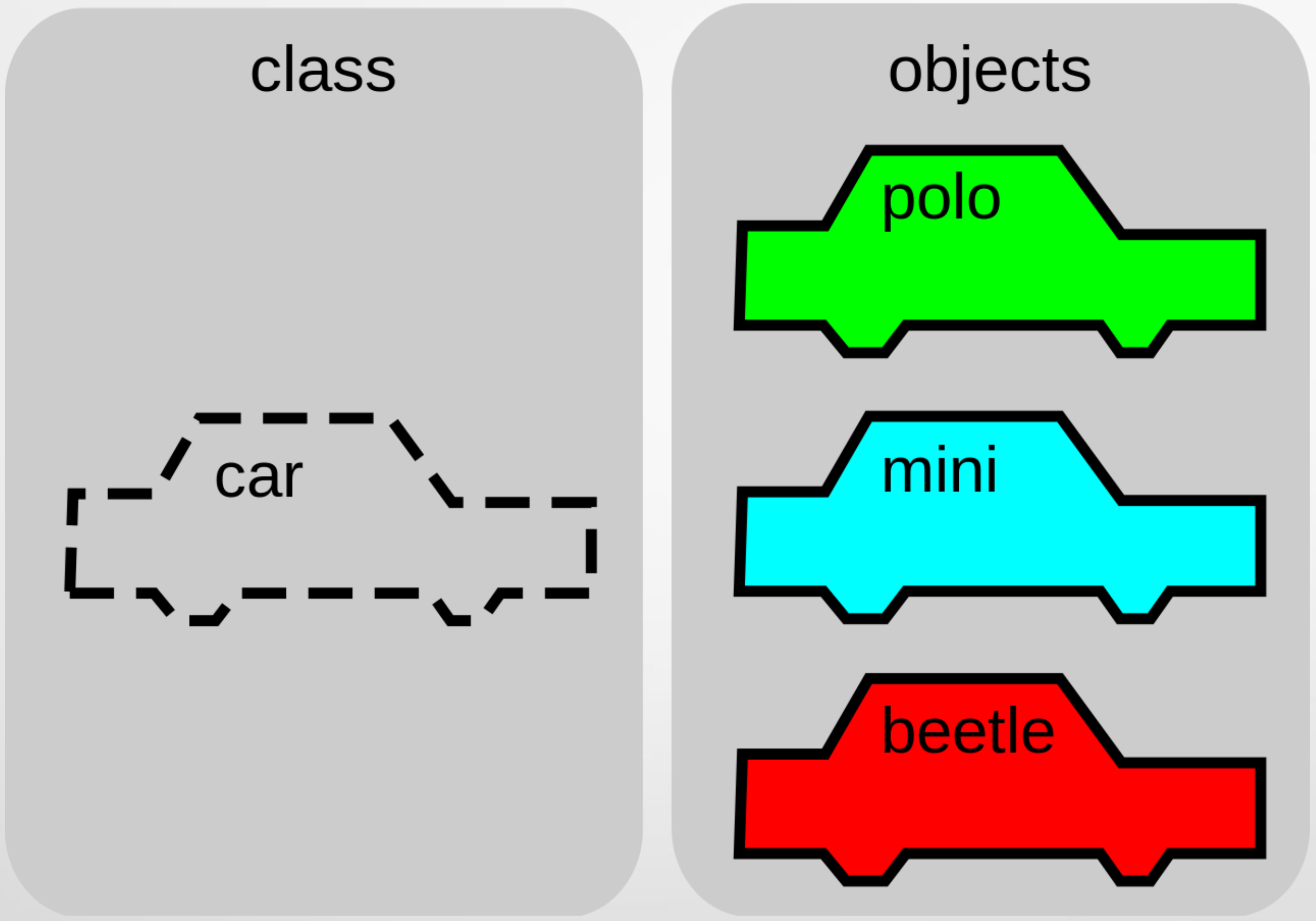
CAR
Fuel maxSpeed

Refule()
getFuel()
setSpeed()
getSpeed()
Drive()

What is an Object ?

- An Object is an instance of a class
- For example if we have class Car
- Then the followings are the objects:

```
car blueCar ;
car hunda = new car();
car [ ] gasCars = new car [ 10 ];
```


class
car
objects
polo
mini
beetle

Object and Class

- Generally, a class declaration contains only keyword **class**, followed by an **identifier(name)** of the class.

```
class testClass
{
        public int a, b;
        public void display()
        {
            Console.WriteLine("Class & Objects in C#");
        }
}
```

Object and Class

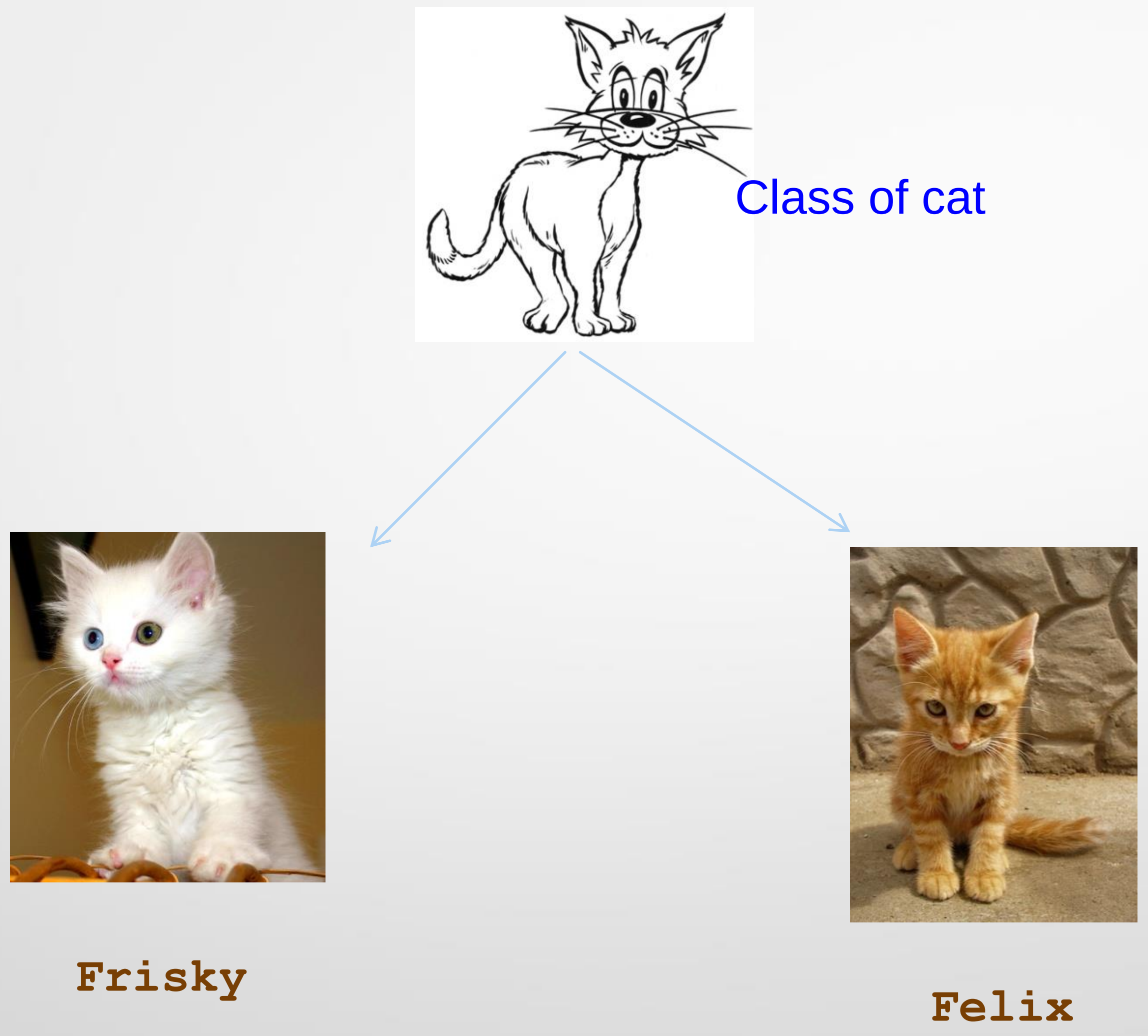
- In c#, **Object** is an instance of a **class** and that can be used to access the data members and member functions of a **class**.
- Generally, we can say that objects are the concrete entities of classes. In c#, we can create an objects by using a **new** keyword followed by the name of the class like as shown below.

```
Users user = new Users();
```

- If you observe above example, we created an instance (**user**) for the class (**Users**). Now the instance “**user**” is a reference to an object that is based on **Users**. By using object name “**user**” we can access all the data members and member functions of **Users** class.

Object and Class

- Declaring an object of a class
  - When a class is defined, we can further define the objects of that class:
    - **Cat Frisky = new cat(); // declare 1 Cat object called Frisky**
    - It states that we are going to handle a **Cat** called **Frisky**
    - It is similar to declaring a number of the type integer
    - **int xyz; // declare 1 integer called xyz**
  - Obviously, if one declares two cats as follows
    - **Cat Frisky = new cat();**
    - **Cat Felix = new cat();**
    - **Frisky** is never the same as **Felix**, although they both belong to the class **Cat**, i.e. they are cats.


Class of cat
Frisky
Felix

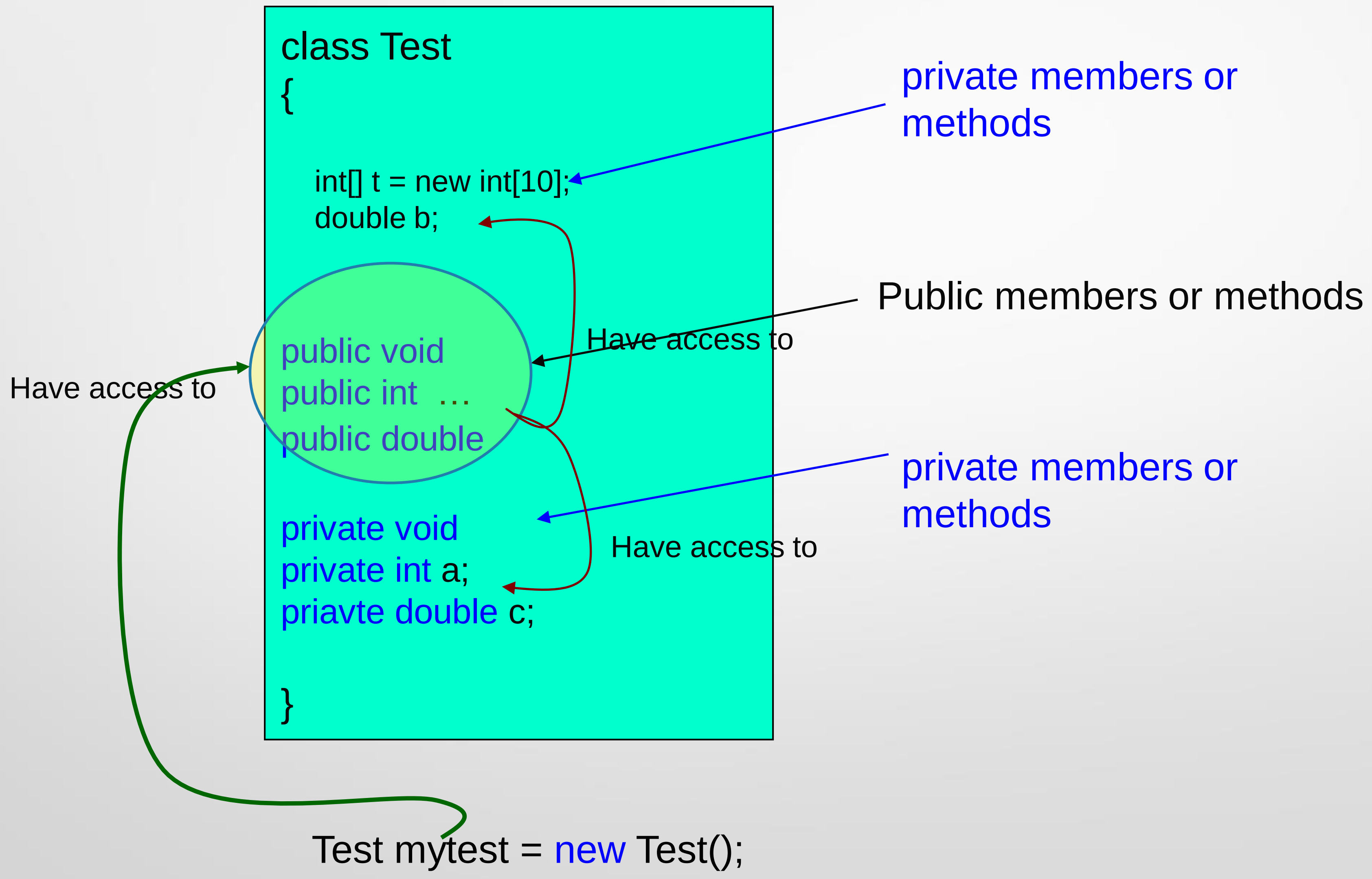
Object and Class

- Accessing Class Members
  - When an object is defined, we can access the **members** of that object based on its class definition
  - For example, if we want to know the weight of **Frisky**:
    - **int weight = Frisky.itsWeight; // Get the weight of Frisky**
    - The operator '**.**' allows us to access the members of the object
  - Similarly, if we want to know the age of **Frisky**:
    - **int age = Frisky.itsAge;   // Get the age of Frisky**
  - If we want to ask **Frisky** to meow:
    - **Frisky.Meow();   // Ask Frisky to execute Meow()**

Private vs. public

- Members can be divided into public members or private members
  - Private members of a class are those members that can only be accessed by methods of that class
  - Public members of a class are those members that can be accessed by other class objects and methods
- Keywords public and private are member access modifiers.
- Instance variables or methods with member access modifier public are accessible wherever the program has a reference to a class object.
- instance variables or methods declared with member access modifier private are accessible only in that class definition.

Private vs. public (Access)



```
class Test
{

   int[] t = new int[10];
   double b;

public void
public int  …
public double

private void
private int a;
priavte double c;

}
```

```
Test mytest = new Test();
```

private members or methods
Public members or methods
Have access to
private members or methods
Have access to
Have access to

Example

```
class Program
{
        static void Main(string[] args)
        {
            Users user = new Users("Ahmad Karim", 30);
            user.GetUserDetails();
            Console.ReadKey();
        }
}
public class Users
{
        public string Name;
        public int Age;
        public Users(string name, int age)
        {
            Name = name;
            Age = age;
        }
        public void GetUserDetails()
        {
            Console.WriteLine("Name: {0}, Age: {1}", Name, Age);
        }
}
```

Example

```
public class customer
{
        public int CustID;
        public string Name;
        public string Address;
        public customer()
        {
            CustID = 1101;
            Name = "Tom";
            Address = "USA";
        }
        public void displayData()
        {
            Console.WriteLine("Customer=" + CustID);
            Console.WriteLine("Name=" + Name);
            Console.WriteLine("Address=" + Address);
        }
}
```

Example

```
class Program
    {
        static void Main(string[] args)
        {
            customer obj = new customer();
            obj.displayData();
            Console.WriteLine(obj.CustID);
            Console.WriteLine(obj.Name);
            Console.WriteLine(obj.Address);
            Console.ReadKey();
        }
    }
```

Multiple Class Declaration

- Sometimes circumstances require multiple classes to be declared in a single namespace. So in that case it is not mandatory to add a separate class to the solution, instead you can attach the new class into the existing program.cs or another one as in the following;

Example

```
class Program
{
        public void MainFunction()
        {
            Console.WriteLine("Main class");
        }
        static void Main(string[] args)
        {
            Program obj = new Program();
            obj.MainFunction();

            demo dObj = new demo();
            dObj.addition();
        }
}
class demo
{
        int x = 10;
        int y = 20;
        int z;
        public void addition()
        {
            z = x + y;
            Console.WriteLine("other class in Namespace");
            Console.WriteLine(z);
        }
}
```