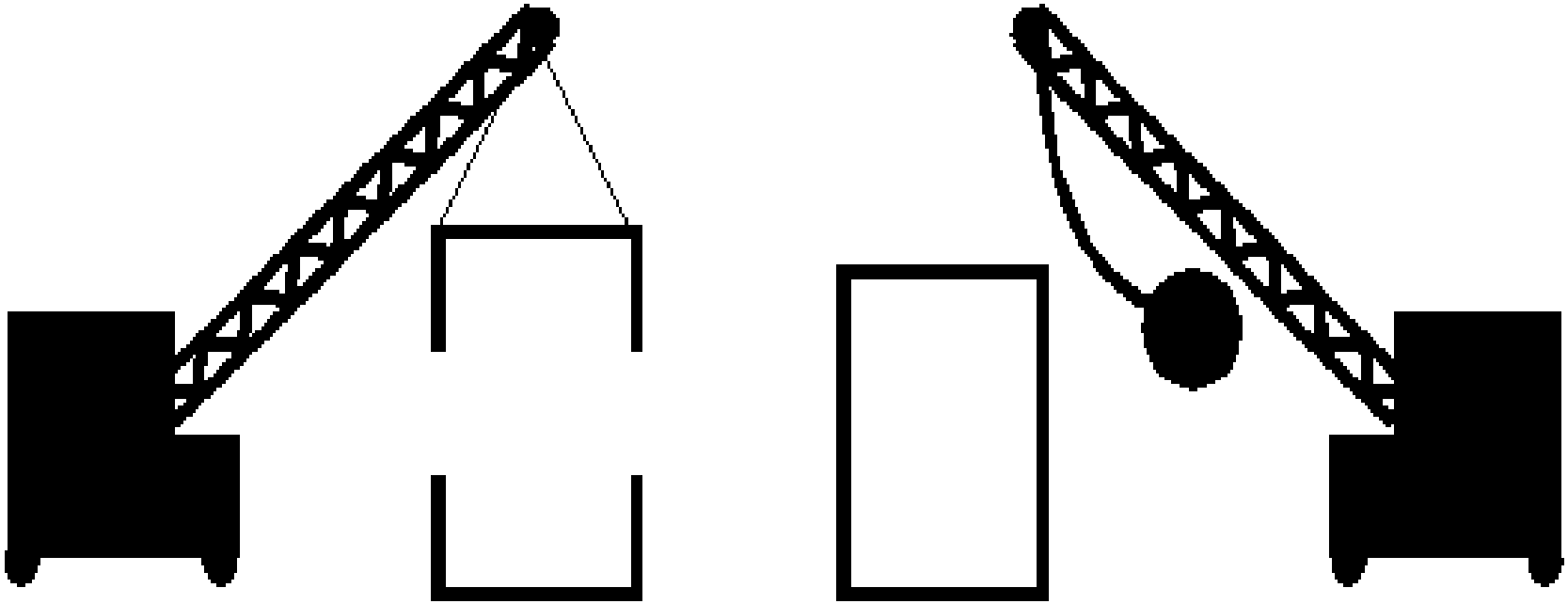
Object Oriented Programming

CONSTRUCTOR AND DESTRUCTOR



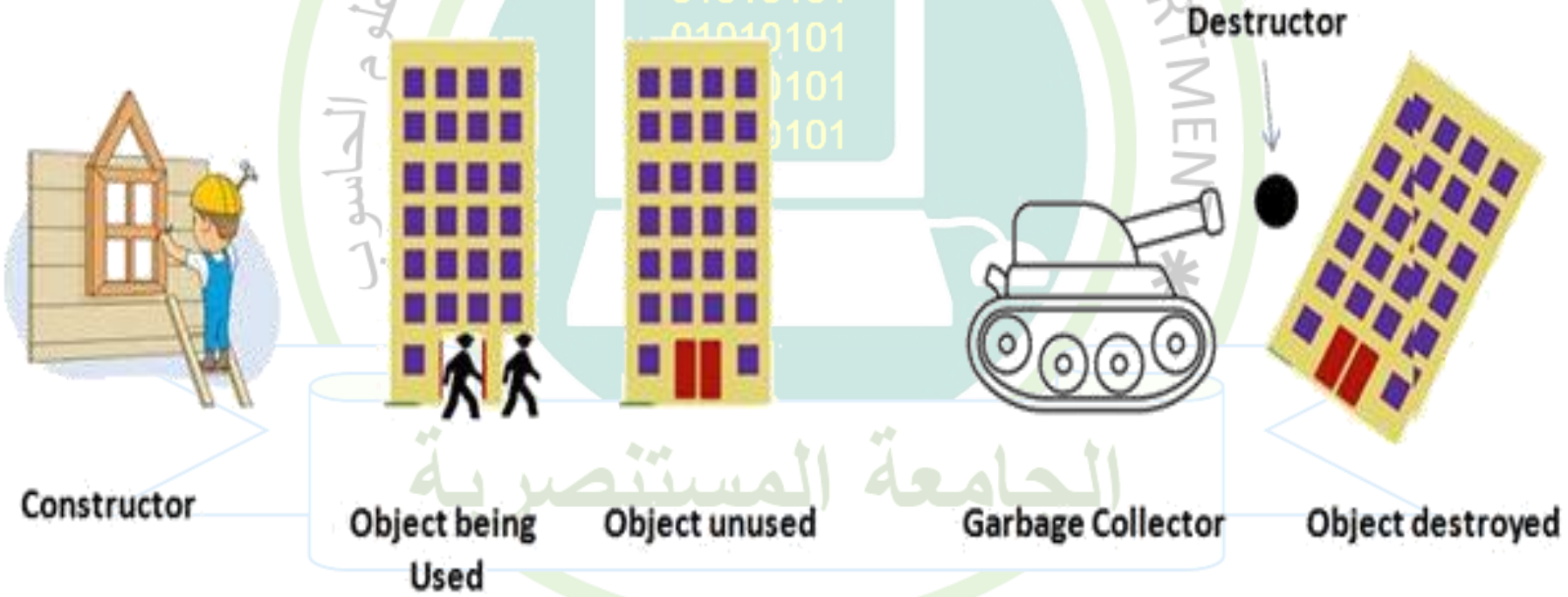
Constructor

Destructor

```
MyClass *MyObjPtr = new MyClass();
```

```
delete MyObjPtr;
```

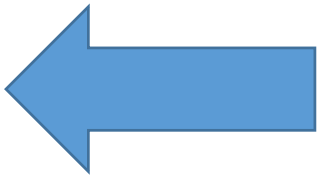
A constructor is a **special method** that is used to initialize objects to initialize fields(data members)
it is called when an object of a class is created.



important notes about constructor :

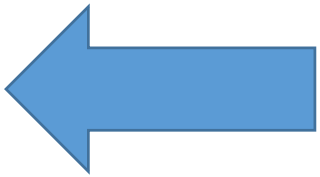
- A constructor has the same name as the class.
- Has no data type - A constructor can never return anything, which is why you don't have to define a return type for it.
- If no constructor defined then the CLR(Common Language Runtime) will provide an implicit constructor which is known as a *Default Constructor*.
- Constructors can be overloaded. - class can have any number of constructors and they vary with the number of arguments that are passed
- We don't use references or pointers on constructors because their addresses cannot be taken.

```
public string bike()
{
}
```



normal method is defined like this:

```
public      bike()
{
}
```



simple constructor(without parameters) can be defined like this:

example of constructor:

```
using System;
program P5Constroctor0
{
    class bike
    {
     private int kilometers;
     private string color;

     public bike()
     {
      Console.WriteLine("Bike's Constracted");
     }

     public bike(int km, string col)
     {
           kilometers = km;
           color = col;
     }

     public void DisplayBikeData()
     {
          Console.WriteLine("Bike's Mileage is " + kilometers + " and color is " + color);
     }
}
```

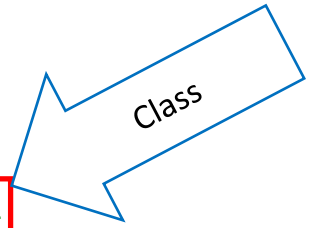
Class

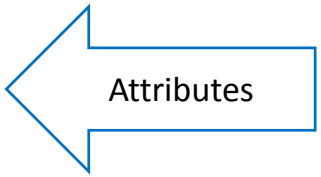
Attributes

//constructor without parameter

//constructor with two parameters "mil" and "col"

example of constructor:






```
using System;
namespace P5Constroctor0
{
    class bike //-------------------------------------
    {
        private int kilometers;
        private string color;

        public bike()
        {
            Console.WriteLine("Bike's Constracted");
        }
    }          //-------------------------------------

    class Program
    {
        static void Main(string[] args)
        {
            bike bicycle = new bike();     // ++++++ create object ++++++++++

            Console.ReadLine();
        }
    }
}
```

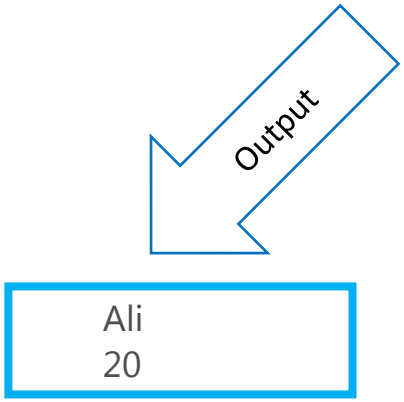
```
using System;
namespace program6
{
    class student
    {
        private string name;
        private int age;

        public student()    // Default Constructor  without parameters
        {
            name = "Ali";
            age  = 20;
        }
    }
```

```
class Program
    {
        static void Main(string[] args)
        {
            student st = new student();

            Console.WriteLine(st.name);
            Console.WriteLine(st.age);
            Console.ReadLine();
        }
    }
}
```

constructor method has called automatically and initialized the parameter values after creating an instance of our class.



```
using System;
namespace program6
{
    class student
    {
        private string name;
        private int d1,d2,d3;
        private double av;
        public student(string a, int b,c,d)    // with parameters
        {
            name = a;
            d1=b; d2=c; d3=d;
        }
        public double findav()
        {
         av=(d1+d2+d3)/3;
         return av;
        }
    }
 class Program
    {
        static void Main(string[] args)
        {
         student st = new student("Ali“,20,40,100);
           ava= st.findav();
          console.writeline(ava);
          Console.ReadLine();
        }
    }
}
```

Output

```
constructor called once th
e
instance of class created
```

Destructors:

garbage cleanup is automatic system, (framework will free the objects that are no longer in use)

BUT there may be times where we need to do some manual cleanup. In this case we can use Destructor, which is used to destroy the objects that we no longer want to use (free or cleanup resources used by the object)

A destructor method called once an object is disposed.

```
class Bike
{
    public Bike()
    {
        //Constructor
    }

    ~Bike()
    {
        //Destructor
    }
}
```

Once the class object is instantiated, *Constructor* will be called

and when object is collected by the garbage collector, *Destructor* method will be called.

The purpose of the destructor method is to remove unused objects and resources.

Destructors are not called directly in the source code but during garbage collection.
A destructor is invoked at an undetermined moment. More precisely a programmer can't control its execution; rather it is called by the Finalize () method.
Like a constructor, the destructor has the same name as the class except a destructor is prefixed with a tilde (~).

There are some limitations of destructors as in the following;

- Destructors are parameterless.
- A Destructor can't be overloaded. (only one function)
- Destructors are not inherited.

```
using System;
program P5Constructor1
{
class traingle
      {
         private int tbase;
         private int thight;
         private double ar;

         public traingle()                    // constructor with parameters
         {
            Console.WriteLine(" THE OBJECT HAS BEEN CREATED "); // just comment

            thight = 10;                              // init variables
            tbase  = 20;

            Console.Write("input hight : ");                  // input variables
            thight = Convert.ToInt32(Console.ReadLine());
            Console.Write("input base : ");
            tbase = Convert.ToInt32(Console.ReadLine());
         }

         public void printinfo() // print information
         {
            Console.WriteLine("BASE = " + tbase.ToString() );
            Console.WriteLine("HIGHT= " + thight.ToString());
         }

         public double findarea()            // find area
         {
            ar = 0.5 * tbase * thight;
            return ar;                 //  we should return the area
         }
      }
```

```
    static void Main(string[] args)
     {
         traingle tshape = new traingle();

         //tshape.printinfo();

         double tar;
         tar = tshape.findarea();

         Console.WriteLine("-----------------");
         Console.WriteLine("AREA="+"{0:N2}",tar);
         Console.WriteLine("-----------------");

         Console.ReadLine();
     }

}
```

```
using System;
program P5Constructor1
{
class triangle
    {
        private int tbase;
        private int thight;
        private double ar;

        public triangle(int a, int b) // constructor with parameters
        {
            thight = a;          // initialize variables when create object
            tbase  = b;
        }

        public void printinfo() // print information
        {
            Console.WriteLine("BASE = " + tbase.ToString() );
            Console.WriteLine("HIGHT= " + thight.ToString());
        }

        public double findarea()          // find area
        {
            ar = 0.5 * tbase * thight;
            return ar;                //  we should return the area
        }
    }
```

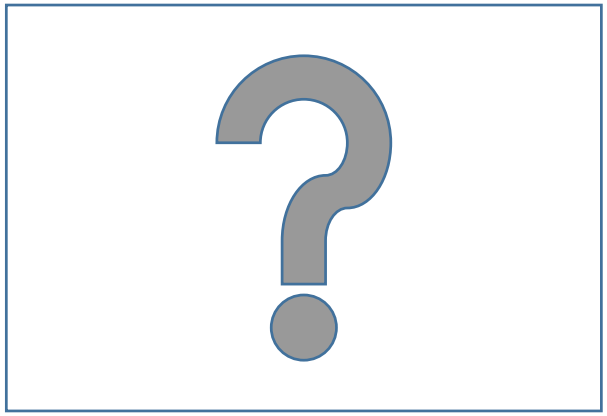
```
    static void Main(string[] args)
    {
        triangle tshape = new triangle(6,12);

        double tar;
        tar = tshape.findarea();

        Console.WriteLine("-----------------");
        Console.WriteLine("AREA="+"{0:N2}",tar);
        Console.WriteLine("-----------------");

        Console.ReadLine();
    }
}
```

QUESTION



Google Classroom :