Destructors

What

- Destructor is a special method of the class, which is used to close un-managed resources (such as database connections and file connections), that are opened during the class execution.

How

Destructor

```
~ClassName( )
{
    //body here...
}
```

Advantage

- We close database connections and file connections; so no memory wastage or leakage.

- Destructor doesn't de-allocate any memory; it just will be called by CLR (.net runtime engine) automatically, just before a moment of deleting the object of the class.

Rules

- Destructor's name should be same as class name, started with ~ (tilde) character.
- A Destructor is unique to its class i.e. there cannot be more than one destructor in a class.
- Destructor can't have parameters or return value.
- Destructor is "public" by default, we can't change its access modifier.
- Destructor doesn't support any other modifiers such as "virtual", "abstract", "override" etc.
- Destructors can be defined only in classes; but not in structs, interfaces etc.
- Destructors can't be overloaded or inherited.
- Destructors are usually called at the end of program execution.

Destructor (vs) Finalize method

Destructor

```
~ClassName
{
    //body here...
}
```

compile

Finalize [After compilation]

```
protected override void Finalize()
{
    try
    {
        //code of destructor
    }
    catch
    {
        throw;
    }
    finally
    {
        base.Finalize();
    }
}
```

- Internally, destructor is compiled as the "Finalize" method.
- The "destructor" is a term belongs to C# language; the "Finalize" method belongs to .net framework generally; and both are same (interchangeable).
- The compiled Finalize method calls the Finalize method of corresponding base class.