* A class doesn't have any memory.
* It is just a template / blue print / prototype to create objects.
* Once we create objects they occupy the Heap Memory of our computer system.
* More number of objects = More Heap Memory = Less Optimized App

∴ We need to destroy the objects after program execution.

To destroy objects we use a destructor.

~ClassName(); → It is automatically invoked after program execution.

There can be only 1 destructor in a class.

* Dynamic Memory Allocation in C++

Allocation → new

Deallocation → delete

1D array :→ Only one single row [2,8,6,4,1]

```
int * array = new int [size];
```

deallocation → `delete[ ]array;`

Two Dimensional Array Square Matrix (n×n)

```
int ** twoD = new int * [n];
```

This will create the n rows

n rows n cols

Initialize column size for each row separately.

i → 0 to n-1

```
twoD[i] = new int[n];
```

n cols

Two Dimensional Array Non-Square Matrix (n×m)

```
int ** twoD = new int * [n];
```

This will create the n rows

n row m cols

Initialize column size for each row separately.

i → 0 to n-1

```
twoD[i] = new int [m];
```

m cols

Story to text file

String → JIT College

file → file.txt

ASCII values → 32 4 8 6 65 9 8

text file → string → JIT College

6 - 8 LPA

Note : Whenever there is a name clash of class → attributes & other variables inside a class, we always use the this pointer. Example :

```
class Employee {
public:
    string Name;
    int Age;
    string Company;
    Employee(string Name, int Age, string Company){
        this->Name = Name;
        this->Age = Age;
        this->Company = Company;
    }
```

class attributes

constructor parameters

In this kind of scenarios : this pointer