Tutorial Link https://codequotient.com/tutorials/class Destructors/5b38c9aac6a1d0259e728ea9

TUTORIAL

class Destructors

Chapter

1. class Destructors

A destructor is used to destroy the object and release the memory. Destructors take no arguments. Destruction of objects takes place when the object leaves its scope of definition or is explicitly destroyed. The latter happens, when we dynamically allocate an object and release it when it is no longer needed. Destructors are declared similar to constructors. But, they also use the name prefixed by a tilde (~) of the defining class. For example:

C++

```
#include<iostream>
#include<cstdio>
#include<cmath>
// Include headers as needed

using namespace std;

class Circle
{
  private:
  double radius;
  public:
  Circle ( ) // Constructor with no argument
  {
    radius=0;
```

```
  }
  Circle (double r) // Constructor with one argument
  {
    radius = r;
  }
  ~ Circle ( ) // Destructor
  {
    cout<<"\n object is being destroyed.";
  }
  double showArea() // Member function
  {
    return radius*radius*3.1416;
  }
};

int main( )
{
  Circle c1, c2(3.5);
  cout<<"Area = "<<c2.showArea()<<endl;
  return 0;
}


```

OUTPUT:

```
area = 38.4846
object is being destroyed
object is being destroyed
```

Some important characteristics are summarized as follows:

- Destructor has the same name as that of the class it belongs to and preceded by ~ (tilde).
- Like constructors, the destructors do not have return type and not even void.

- Destructors can be virtual.
- The destructors do not have any arguments.
- The destructors cannot have default values.
- They cannot be overloaded.
- They cannot be inherited. However a derived class can call the base class destructor.