Destructor

PRESENTED BY-
NEELAM SINGH
DEPARTMENT OF COMPUTER SCIENCE

What is a Destructor in C++?

- Destructor is a member function that is instantaneously called whenever an object is destroyed. The destructor is called automatically by the compiler when the object goes out of scope i.e. when a function ends the local objects created within it also gets destroyed with it. The destructor has the same name as the class name, but the name is preceded by a tilde
- **Syntax of Destructor:**

```
class scaler {
 public:
scaler(); //constructor
~scaler(); //destructor
};
```

Characteristics of a Destructor in C++

- A destructor deallocates memory occupied by the object when it's deleted.
- A destructor cannot be overloaded. In function overloading, functions are declared with the same name in the same scope, except that each function has a different number of arguments and different definitions. But in a class, there is always a single destructor and it does not accept any parameters, hence, a destructor cannot be overloaded.
- A destructor is always called in the reverse order of the constructor. In C++, variables and objects are allocated on the Stack. The Stack follows LIFO (Last-In-First-Out) pattern. So, the deallocation of memory and destruction is always carried out in the reverse order of allocation and construction. This can be seen in code below.
- A destructor can be written anywhere in the class definition. But to bring an amount of order to the code, a destructor is always defined at the end of the class definition.

Implementation of Constructors and Destructors in C++

```
#include <iostream>

using namespace std;
class Department {
 public:
   Department() {
 //constructor is defined
  cout << "Constructor Invoked for Department class" << endl;
  }

   ~Department() {
     //destructor is defined
     cout << "Destructor Invoked for Department class" << endl;
   }
};
class Employee {
 public:
  Employee() {
    //constructor is defined
    cout << "Constructor Invoked for Employee class" << endl;
   }

   ~Employee() {
     //destructor is defined
     cout << "Destructor Invoked for Employee class" << endl;
   }
};
int main(void) {
 Department d1; //creating an object of Department
 Employee e2; //creating an object of Employee
 return 0;
}
```

OUTPUT

Constructor Invoked for Department class

Constructor Invoked for Employee class

Destructor Invoked for Employee class

Destructor Invoked for Department class

Explanation:

- When an object named d1 is created in the first line of main() i.e (Department d1), it's constructor is automatically invoked during the creation of the object. As a result, the first line of output "Constructor Invoked for Department class" is printed. Similarly, when the e2 object of Employee class is created in the second line of main() i.e (Employee e2), the constructor corresponding to e2 is invoked automatically by the compiler and "Constructor Invoked for Employee class" is printed.
- A destructor is always called in the reverse order as that of a constructor. When the scope of the main function ends, the destructor corresponding to object e2 is invoked first. This leads to printing "Destructor Invoked for Employee class". Lastly, the destructor corresponding to object d1 is called and "Destructor Invoked for Department class" is printed.

Difference Between Constructors and Destructors

Constructor	Destructor
A constructor is called when a new instance of a class is created.	A destructor is called when an already existing instance of a class is destroyed.
It is called each time a class is instantiated.	It is called automatically when an object is deleted from the memory.
Constructors often accept arguments.	Destructors cannot have arguments.
It allocates memory to a newly created object.	It deallocates memory of an object after its deletion.
Multiple constructors can coexist inside a class.	There can always be only one destructor in a class.
It can be overloaded.	It cannot be overloaded.
They have the same name as the class name.	They have the same name as the class name except they are prefixed with a ~ (tilde) symbol.

THANK YOU