Constructor and Destructors

Can a function has same name of class name ?

- **constructor** is a special member function of a class that is executed whenever we create new objects of that class.
- A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Facts

- By default all the objects are calling Zero argument constructor
- Constructors are invoked when objects are created
- No return type because while creating objects return is not possible
- Very good for initialization of variables
- Life of the object is in the hand of programmer

Types of Constructor

- Default
- Parametric
- copy

Default Constructor

```
#include <iostream.h>
 class san
{
private : int a;
public:
san()
{
   a=0;
} };

int main(void)
{
san s1;
cout<<s1.a;   // not possible
return 0;
}
```

Parameterized Constructor

```
#include <iostream.h>
 class san
{
private : int a;
public:
san(a)
{
a=a+1;
cout<<a;
} };

int main(void)
{
san s1(20);
return 0;
}
```

Parameterized Constructor

```
#include <iostream.h>
 class san
{
private : int a;
public:
san(a)
{
a=a+1;
cout<<a;
} };

int main(void)
{
san s1(20);
return 0;
}
```

Parameterized Constructor

```
#include <iostream.h>
 class san
{
private : int a,float b;
public:
san(a,b)
{
b=a+b;
cout<<b;
} };

int main(void)
{
san s1(20,23.56);
return 0;
}
```

Copy Constructor

- In this Constructor we pass the object of class into the another Object of Same Class. As name Suggests you Copy, means Copy the values of one Object into the another Object of Class . The copy constructor is used to –
- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

```
class  san
{ int a, b;
public:
san(int  x, int  y)
{
a = x; b = y;
}
void out()
{ cout << a << b; }
san( const san &x1)
{
  a=x1.a; b=x1.b);
}
}; // end of class
```

```
int main()
{ san s1(10, 20);
san s2=s1; //Copy Constructor
s1.out();
s2.out();
return 0;
}
```

Destructor

Constructors are used to initialise variables and for Assigning Some Values this contains memory so that to free up the Memory which is Allocated by Constructor, destructor is used.

- Called Automatically at the End of Program and we doesn’t have to Explicitly Call a Destructor

- Destructor Cant be Parameterized or a Copy This can be only one means Default Destructor which Have no Arguments.

- For Declaring a Destructor we have to use ~ tiled Symbol in front of Destructor.

Destructor

A destructor function is called automatically when the object goes out of scope:
(1) the function ends
(2) the program ends
(3) a block containing local variables ends
(4) a delete operator is called

Destructor

```
class san
{
private : int a;
public:
san()
{
   a=0;
}
~san();
 };

int main(void)
{
san s1;
cout<<s1.a;   // not possible
return 0;
}
```

Important Question

- How Many constructors are possible in a class
- How Many destructor are possible in a class
- Can we overload a constructor