Constructor and Destructor

Constructor

It is a member function having same name as it's class and which is used to initialize the objects of that class type with a legal initial value. Constructor is automatically called when object is created.

Types of Constructor

Default Constructor-: A constructor that accepts no parameters is known as default constructor. If no constructor is defined then the compiler supplies a default constructor.

```
Circle :: Circle()
{
    radius = 0;
}
```

Parameterized Constructor -: A constructor that receives arguments/parameters, is called parameterized constructor.

```
Circle :: Circle(double r)
{
    radius = r;
}
```

Copy Constructor-: A constructor that initializes an object using values of another object passed to it as parameter, is called copy constructor. It creates the copy of the passed object.

```
Circle :: Circle(Circle &t)
{
    radius = t.radius;
}
```

There can be multiple constructors of the same class, provided they have different signatures.

Destructor

A destructor is a member function having sane name as that of its class preceded by ~(tilde) sign and which is used to destroy the objects that have been created by a constructor. It gets invoked when an object's scope is over.

```
~Circle() {}
```

Example : In the following program constructors, destructor and other member functions are defined inside class definitions. Since we are using multiple constructor in class so this example also illustrates the concept of constructor overloading

```
#include<iostream>
using namespace std;

class Circle //specify a class
{
    private :
        double radius; //class data members
    public:
        Circle() //default constructor
        {
            radius = 0;
        }
        Circle(double r) //parameterized constructor
        {
            radius = r;
        }
        Circle(Circle &t) //copy constructor
        {
            radius = t.radius;
        }
        void setRadius(double r) //function to set data
        {
            radius = r;
        }
        double getArea()
        {
            return 3.14 * radius * radius;
        }
        ~Circle() //destructor
        {}
};

int main()
{
```

```
    Circle c1; //default constructor invoked
    Circle c2(2.5); //parmeterized constructor invoked
    Circle c3(c2); //copy constructor invoked
    cout << c1.getArea()<<endl;
    cout << c2.getArea()<<endl;
    cout << c3.getArea()<<endl;
    return 0;
}
```

Another way of Member initialization in constructors

The constructor for this class could be defined, as usual, as:

```
Circle :: Circle(double r)
{
    radius = r;
}
```

It could also be defined using member initialization as:

```
Circle :: Circle(double r) : radius(r)
{}
```