Q:Define a university class as follows:

Private members	**Name, Student _No, Fees**
Public Functions	Read( ): set the private variables
	Show( ): display the variables
	Check(): check the fees if they are grater than 300$ then write “private”, otherwise print “Suitable”
Define Anbar and Baghdad as objects	

Lecture 6
Constructors and Destructors

University of Anbar
College of Computer Science and Information Technology
Department of Computer Science
Object Oriented Programming
Second Class
Dr. Ruqayah R. Al-Dahhan

Outlines:

- Constructors
- Destructors
- Copy constructor

Constructors

- Classes can have a special member function - a **constructor** - that is called when an object is created.

```
class Point {
  public:
  Point(int i, int j);
  int x,y;      };
Point::Point(int i, int j)
  { x = i; y = j;}
```

- The constructor function has the **same name** as the class name, it has **no return type.** It is often just inline**.**

Object initialization

- We now create a new point with:

```
Point p(4,5);
```

- This method has a problem though - we can’t ask for an uninitialized point:

```
Point t;
```

produces an error-Point now needs two arguments.

Object initialization

- We use **function overloading** to have several versions of the Point constructor function:

```
class Point {
  public:
    Point();
    Point(int i, int j);
  private:
   int x,y;   };
 Point::Point(){x = 0; y = 0;}
 Point::Point(int i, int j)
     {x = i; y = j;}
```

Object initialization

Point t; //now valid: x,y are 0,0

- A constructor with no arguments is called the **default** constructor.
- If a class does not contain **any** constructor the compiler inserts a **system default constructor** (function).

Destructors

When an object is destroyed - the object’s **destructor** is called.

If we don’t free that memory before the object disappears, then the memory will never be freed - **a memory leak**. Can cause programs to crash

Destructors

Destructors are used to release any resources allocated by the object.

Destructors are a "prepare to die" member function.

They are often abbreviated "dtor".
Constructors are “ctor”.

Destructors - same name as class with ~ prefix

```
class Str {
 public:
   Str();
   ~Str();
 private:
   char s;
};

Str::Str()
{s = ' '; ………..}

Str::~Str() {cout<<"delete s";
…………….}
```

Destructors

A destructor:

- called by the system for you when an object is destroyable (e.g about to go out of scope)

- has the same name as the class;

- with a ~ at the front;

- does not have return values;

- cannot have arguments.

When are constructors/destructors called?

Constructors and destructors are called automatically.

The order in which they are called depends on the order in which **execution enters and leaves the scope** in which objects are instantiated and **the type of storage** for objects.

General rule: destructor calls are made in the reverse order of the constructor calls.

```
class C {
public:
   C(int);        //constructor
  ~C();            //destructor
private:
   int data;
};
C::C(int value){
  data = value;
  cout<<"\nCtor called: "<< data;
}
C:: ~C(){
  cout<<"\nDtor called: "<< data;
}
```

```
void createF();
C one(1);                                       //global object
int main(){
   cout <<"Main starts here."<<endl;
   C two(2);                        //local object
   cout<<"After two(local)in main."<<endl;
   createF();                                 //f call
}

void createF(){
   cout <<endl<<" F STARTS HERE. "<<endl;
   C ten(77); //local object
   cout<< "LAST IN F. "<<endl<<endl;
}
```

Output:

```
Ctor called: 1
Main starts here.
Ctor called: 2
After two (local ) in main.

F STARTS HERE.
Ctor called: 77
LAST IN F.

Dtor called: 77
Dtor called: 2
Dtor called: 1
```

When are constructors/destructors called?

For stack objects defined:	Constructors called:	Destructor called:
In global scope	Before any other function (including main)	When `main` terminates, or `exit` is called
Local objects	When the object enters scope.	When the object leaves scope
local objects	Once, when the object enters scope the first time.	When `main` terminates, or `exit` is called

Summary

A **constructor** constructs objects of its class type. This process may involve data members and allocating free store, using operator `new`.

A **default constructor** is a constructor requiring no arguments .

A **destructor** “release any resources allocated by the object, typically by using delete.