The lifecycle of an object in object-oriented programming (OOP) involves the creation and destruction of objects.

These areas are uniquely addressed by constructors and destructors, which are special member functions in languages like C++ and others.

Construction of Objects:

1. Constructor:

- Any time a new object of the class is created, a constructor is automatically called.
- It initializes the state of the object, allocates resources and carries out some setup tasks that may be necessary.
- Constructors do not have any return type, but they bear the name of their class.

C++

```
class MyClass {
public:
    // Constructor
    MyClass() {
        // Initialization code goes here
        cout << "Constructor called!";
    }
};
```

2. Default constructor:

- If you don’t write a constructor for your class, the compiler will make one for you.
- Such automatic constructor creates an object with default values or uninitialized

members according to their types.

C++

```
class MyClass {
    // No explicit constructor, so a default constructor is provided
by the compiler
};
```

3. Parameterized Constructor:

- Constructors can take parameters, allowing you to initialize the object with specific values.

C++

```
class Point {
public:
    // Parameterized constructor
    Point(int x, int y) : xCoord(x), yCoord(y) {
        cout << "Parameterized constructor called!";
    }

private:
    int xCoord;
    int yCoord;
};
```

Destruction of Objects:

1. Destructor:

A destructor is a special member function called when an object goes out of scope or is explicitly deleted, and it is used to release resources, perform cleanup and deallocate memory.

C++

```
class MyClass {
public:
    // Constructor
    MyClass() {
        cout << "Constructor called!";
    }

    // Destructor
    ~MyClass() {
        // Cleanup code goes here
        cout << "Destructor called!";
    }
};
```

2. Automatic Destruction:

At the end of a function, for example when an object goes out of its scope, its destructor is called automatically.

C++

```
void someFunction() {
    MyClass obj; // Constructor called

    // obj goes out of scope here, and its destructor is automatically
called
    // Destructor called
}
```

3. Manual Destruction:

The ‘delete’ keyword can be used to manually delete a dynamic object.

C++

```
void anotherFunction() {
    MyClass* objPtr = new MyClass(); // Constructor called

    delete objPtr; // Destructor called
}
```

Related posts:

1. Abstraction and encapsulation
2. Object Oriented Programming & Methodolog Viva Voce
3. How to install compiler for code blocks
4. Object Oriented Programming
5. Differences between Procedural and Object Oriented Programming
6. Features of Object Oriented Paradigm

7. Inheritance in Object Oriented Programming
8. Object Oriented Programming
9. Introduction to Object Oriented Thinking & Object Oriented Programming
10. Difference Between Object-Oriented Programming (OOP) and Procedural Programming
11. features of Object oriented paradigm
12. Merits and demerits of Object Oriented methodology
13. Concept of Objects: State, Behavior & Identity of an object
14. Access modifiers
15. Static members of a Class
16. Instances in OOP
17. Message Passing in OOP