In object-oriented programming (OOP) languages, access modifiers are keywords that define the visibility of classes and their members (such as fields, properties, and methods) and, in some cases, the class itself. These modifiers regulate how much of a program can use members or classes.

Different programming languages may use different keywords for access modifiers, but the basic concepts are similar. Here are common access modifiers:

1. Public access modifier:

Members marked as public are accessible from any part of the program.

Public access modfier example in Java language:

C++

```
class MyClass {
public:
    int myPublicField;

    void myPublicMethod() {
        cout<<"Welcome to EasyExamNotes.com";
    }
};
```

2. Private access modifier:

Members marked as private are only accessible within the same class. They are not accessible from outside the class.

Private access modifier example in C++ language:

C++

```
class MyClass {
private:
    int myPrivateField;
public:
    void setPrivateField(int value) {
        myPrivateField = value;
    }
};
```

3. Protected access modifier:

Members marked as protected are accessible within the same class and its subclasses (derived classes).

Protected access modifier example in C++:

C++

```
class MyBaseClass {
protected:
    int myProtectedField;
};

class MyDerivedClass : public MyBaseClass {
public:
    void AccessProtectedField() {
        myProtectedField = 10; // Accessible in the derived class
    }
```

```
};
```

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. Static members of a Class
15. Instances in OOP
16. Message Passing in OOP
17. Construction and destruction of Objects