Discuss about

**1.Java Access Modifiers with Method Overriding with example .**

**Answer :** [Java access modifiers are keywords that specify the visibility and accessibility of classes, methods, fields and constructors1](https://www.geeksforgeeks.org/method-overriding-with-access-modifier/). There are four types of access modifiers in Java:

* **private**: The member is accessible only within the same class.
* **default** (no keyword): The member is accessible within the same package.
* **protected**: The member is accessible within the same package and by subclasses in other packages.
* **public**: The member is accessible from anywhere.

Method overriding is a feature that allows a subclass to provide a specific implementation of a method that is already defined in its superclass[2](https://www.geeksforgeeks.org/overriding-in-java/). The subclass method must have the same name, parameters and return type as the superclass method.

When overriding a method, the access modifier of the subclass method can be the same or less restrictive than the superclass method, but not more restrictive[2](https://www.geeksforgeeks.org/overriding-in-java/)[3](https://www.mygreatlearning.com/blog/the-access-modifiers-in-java/). For example:

* If the superclass method is **private**, it cannot be overridden by the subclass (it can only be hidden).
* If the superclass method is **default**, it can be overridden by the subclass with default, protected or public access modifier.
* If the superclass method is **protected**, it can be overridden by the subclass with protected or public access modifier.
* If the superclass method is **public**, it can be overridden by the subclass only with public access modifier.

The reason for this rule is to ensure that the subclass does not violate the principle of substitution, which states that an object of a subclass can be used wherever an object of its superclass is expected[4](https://www.tutorialspoint.com/method-overriding-with-access-modifiers-in-Java). If the subclass method is more restrictive than the superclass method, it may cause unexpected errors or access violations when using polymorphism.

For example, consider the following code snippet:

class A {

public void test() {

System.out.println("A");

}

}

class B extends A {

private void test() { // Error: attempting to assign weaker access privileges

System.out.println("B");

}

}

public class Main {

public static void main(String[] args) {

A a = new B(); // Polymorphism: using a subclass object as a superclass reference

a.test(); // Which method will be called?

}

}

In this example, class B tries to override the test() method of class A with a private access modifier. This is not allowed because it makes the test() method less accessible in class B than in class A. If this was allowed, then the polymorphic call a.test() in the main method would cause an error, because the test() method of class B would not be visible to the reference variable of type A. Therefore, Java prevents this situation by giving a compile-time error. To fix this error, class B should use either public or protected access modifier for its test() method.

2.Write a program in class uses private access control.

Answer 2: / A class that uses private access control

class Person {

// Private fields to store the name and age of a person

private String name;

private int age;

// A constructor to initialize the fields

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// A public method to display the details of a person

public void display() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

// A private method to check if a person is eligible to vote

private boolean isEligibleToVote() {

return age >= 18;

}

// A public method to call the private method and display the result

public void checkVotingEligibility() {

if (isEligibleToVote()) {

System.out.println(name + " is eligible to vote.");

} else {

System.out.println(name + " is not eligible to vote.");

}

}

}

// A class to test the program

public class Main {

public static void main(String[] args) {

// Create a person object using the Person constructor

Person p1 = new Person("Raj", 20);

// Display the details of the person using the display method

p1.display();

// Check the voting eligibility of the person using the checkVotingEligibility method

p1.checkVotingEligibility();

}

}