

Check access modifiers about the methods to be overridden, Public, Static, Default, Private and Final. Write a report and provide an example for all of them.

## Access Modifiers with Method Overriding in Java

Prepared by:

Alan Ari Mohammed

Supervised by:

Mr. Akam Hamad

Access modifiers in java are special keywords that can change the accessibility and visibility level of classes, methods, and fields.

Overriding in java is used with inheritance which allows a subclass to make its own implementation of a method that is already declared in its superclass.

An access modifier with a method can change the ability of that method to be overridden. In this report those cases will be discussed with a code example for each one.

Note 1: The Main class code is merely here to avoid duplicated code:



```
1 public class Main {
2     public static void main(String args[]) {
3         Stage1 obj = new Stage1();
4         obj.study();
5     }
6 }
```

Here the Main class and the main method is created. An object is created for a class which one of its methods is invoked after that.

Note 2: Overriding can occur only if the overridden and overriding method have same name and the overriding method cannot be more restrictive than the overridden method.

## Public

Public methods in the parent class can be overridden only with public methods not with other access modifiers.

```
9 //Parent class
10 class Software {
11     //Public method to be overridden
12     public void study() {
13         System.out.println("parent");
14     }
15 }
16
17 //1st Child class
18 class Stage1 extends Software {
19     //Method overriding
20     public void study() {
21         System.out.println("1st child");
22     }
23 }
```

Here the “child” will be printed to the console. If the access modifier in the study method in Stage1 class changed the javac (name of java compiler) will detect it and give an error before running the code.

## Protected

Protected methods in the parent class can be overridden only with public and protected methods in the child classes not with default and private.

```
9 //Parent class
10 class Software {
11     //Protected method to be overridden
12     protected void study() {
13         System.out.println("parent");
14     }
15 }
16
17 //1st Child class
18 class Stage1 extends Software {
19     //Method overriding
20     public void study() {
21         System.out.println("1st child");
22     }
23 }
24
25 //2nd Child class
26 class Stage2 extends Software {
27     //Method overriding
28     protected void study() {
29         System.out.println("2nd child");
30     }
31 }
```


## Default

There is no keyword that refers to the default access modifier. If the method does not have any access modifier, then the default access modifier will be applied. They can be overridden only with public, protected and default methods not with privates.

```
9 //Parent class
10 class Software {
11     //Default method to be overridden
12     void study() {
13         System.out.println("parent");
14     }
15 }
16
17 //1st Child class
18 class Stage1 extends Software {
19     //Method overriding
20     public void study() {
21         System.out.println("1st child");
22     }
23 }
24
25 //2nd Child class
26 class Stage2 extends Software {
27     //Method overriding
28     void study() {
29         System.out.println("2nd child");
30     }
31 }
```

## Private

In private method overridden will not occur because the private modifier does not allow the method to be inherited and will only be private to that class.



```
9 //Parent class
10 class Software {
11     /*Private methods can not be overridden because
12     only in this class private methods are accessible*/
13     private void study() {
14         System.out.println("parent");
15     }
16 }
```

## Final

When a method is declared as final, it means that method cannot be overridden and can have only one implementation. The final keyword has no link with restrictiveness of methods.

```
9 //Parent class
10 class Software {
11     /*Final methods can not be overridden because the
12     final keyword is used to make the method have final
13     implementation*/
14     static void study() {
15         System.out.println("parent");
16     }
17 }
18
19 //1st Child class
20 class Stage1 extends Software {
21     /*Will give an error because final does not allow
22     the method to be overridden*/
23     final void study() {
24         System.out.println("1st child");
25     }
26 }
```



## Static

Static methods cannot be overridden because static methods are bound to the class, not to the object. If a subclass defines a static method with the same name and signature as a static method in its parent class, it is said to be hiding the static method of the parent class. The static keyword also has no link with restrictiveness of methods

```
9 //Parent class
10 class Software {
11     /*Static methods bound to the class so it
12     cannot be overridden*/
13     static void study() {
14         System.out.println("parent");
15     }
16 }
17
18 //1st Child class
19 class Stage1 extends Software {
20     /*Hiding the study method that has been declared
21     in the parent class*/
22     static void study() {
23         System.out.println("1st child");
24     }
25 }
```

However, if the study method in the child class declared without static, then it will give an error. This is because hiding is not like overriding and other methods

in child class cannot have the same name of the hidden method if they are not static.