**Method Overloading**

Developing multiple methods with **same method name** but **different method signatures** is known a method overloading.

The signature should change either in terms of

1. Number of arguments.
2. Datatype of arguments.
3. Sequence of argument.
4. Return type of method with different arguments

**class** Overload {

**void** addition(**int** a, **int** b) {

**int** result = a + b;

}

**void** addition(**int** a, **float** b) {

}

**void** addition(**float** a, **int** b) {//13.3, 10

}

**void** addition(**float** a, **float** b) {

}

**void** addition(**int** a, **int** b, **int** c) {

**int** result = a + b + c;

}

}

**class** SubOverload **extends** Overload {

**void** addition(**int** a, **int** b, **int** c, **int** d) {

}

}

**public** **class** Run2 {

**public** **static** **void** main(String[] args) {

SubOverload sub = **new** SubOverload();

sub.addition(10,10.13f);

}

}

**Important**

1. Method overloading applies to both **static and non-static methods.**
2. Method overloading can happen between **static methods and non static methods**
3. Method can be overloaded in the same class (usually) or sub class.
4. varying with **only return type cannot achieve method overloading**

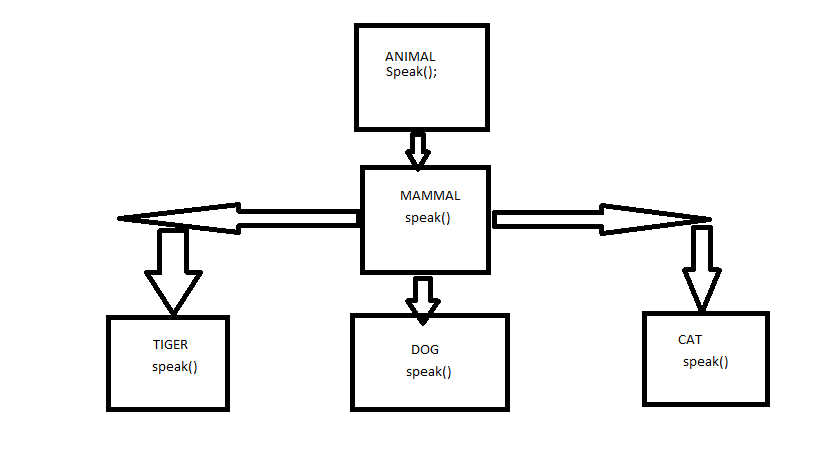
**Method Overriding**

**Definition:**

Subclass providing new implementation of already defined method in the super class i.e., **redefining the non-static method which is already defined in the super class.**

**Rules for method overriding:**

1. There should be **inheritance Method** should be non-static
2. Method should have same name as in superclass with same signature and same sequence and return type should also be same. For non-primitive return type can change
3. **Static methods are not over ridden** in subclass because static member belongs to class but it is hidden.



package methodOverriding;

class Animal{

void speak()

{

System.out.println("Animal Speaking");

}

}

class Mammal extends Animal{

void speak()

{

System.out.println("Mammal Speaking");

}

}

class Dog extends Mammal{

void speak()

{

System.out.println("BOW BOW");

}

}

public class Demo50 {

public static void main(String args[])

{

Mammal m1=new Mammal();

m1.speak();

Animal a1=new Animal();

a1.speak();

Dog d1=new Dog();

d1.speak();

}

}

**Can we override Static methods?**

No, Static members cannot be over ridden. They belong to class , though it get inherited , if we redefine . The original method gets hidden.

**What is method hiding?**

When you have **a static method in the subclass** which has same method name and signature and return type as the method in super class the then **sub class method is set to hide the super class inherited method**.

This applies for only static method

**Can we override final methods?**

No, final methods cannot be over ridden because its constant or fixed.

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| Method overloading is used *to increase the readability* of the program. | Method overriding is used *to provide the specific implementation* / NEW implementation of the method that is already provided by its super class. |
| Method overloading is performed *within class* or different class | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| In case of method overloading, *parameter must be different*. | overriding, *parameter must be same*. |
| Method overloading is the example of ***compile time polymorphism*.** | Method overriding is the example of ***run time polymorphism*.** |
| In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |