**Method Overloading**

## ***Compile time Polymorphism (or Static polymorphism)***

Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading is an example of compile time polymorphism.

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to [**constructor overloading**](https://beginnersbook.com/2013/05/constructor-overloading/) in Java, that allows a class to have more than one constructor having different argument lists

Java supports method overloading and always occur in the same class

## **Three ways to overload a method**

1. Number of parameters.  
For example: This is a valid case of overloading

add(int, int)

add(int, int, int)

2. Data type of parameters.  
For example:

add(int, int)

add(int, float)

3. Sequence of Data type of parameters.  
For example:

add(int, float)

add(float, int)

**Method overloading** is an example of [Static Polymorphism](https://beginnersbook.com/2013/04/runtime-compile-time-polymorphism/).

**Points to Note:**  
1. Static Polymorphism is also known as compile time binding or early binding.  
2. [Static binding](https://beginnersbook.com/2013/04/java-static-dynamic-binding/) happens at compile time. Method overloading is an example of static binding where binding of method call to its definition happens at Compile time.

**public** **class** OverlodingDemo

{

//Number of parameters.

**public** **void** method(**int** a, **int** b)

{

**int** c=a+b;

System.***out***.println(c);

}

**public** **void** method(**int** a, **int** b, **int** c)

{

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

System.***out***.println(d);

}

//Difference in data type of parameters

**public** **void** method(**int** a)

{

System.***out***.println(a);

}

**public** **void** method(String a)

{

System.***out***.println(a);

}

//Sequence of data type of arguments

**public** **void** method(**double** a, **int** b)

{

**double** c=a+b;

System.***out***.println(c);

}

**public** **void** method(**int** a, **double** b)

{

**double** c=a+b;

System.***out***.println(c);

}

}

OverlodingDemo over=**new** OverlodingDemo();

//compiler called 1st method

over.method(10,10);

//compiler called 2nd method

over.method(10,10,10);

//compiler called 3nd method

over.method(10);

//compiler called 4th method

over.method("A");

//compiler called 5th method

over.method(25.25,25);

//compiler called 6th method

over.method(25,50.568);

**Type Promotion table:**

When a data type of smaller size is promoted to the data type of bigger size than this is called type promotion, for example: byte data type can be promoted to short, a short data type can be promoted to int, long, double etc.

Java supports automatic type promotion

The data type on the left side can be promoted to the any of the data type present in the right side of it.

byte → short → int → long

short → int → long

int → long → float → double

float → double

long → float → double

### **Overloading main Method**

In Java, we can overload the main () method using different number and types of parameters but the JVM only understand the original main () method.