Method Signature

A method signature consists only of the name of the method and the parameter types and their order. The modifiers, return type and throws clause are not part of the signature.

Function prototypes are a C concept that is not relevant to Java.

public static int add(int i, int j) → called method declaration

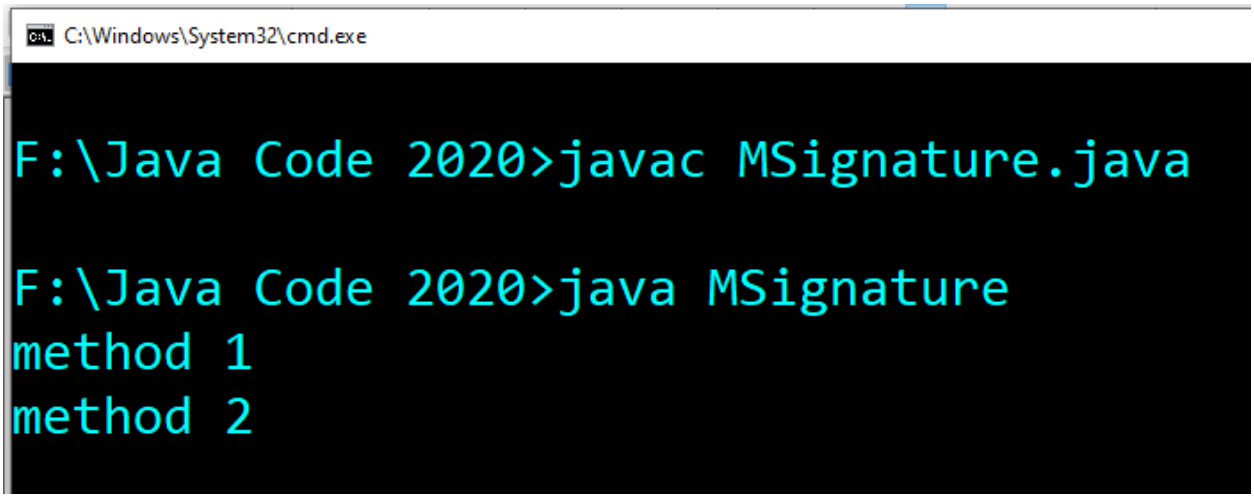
add(int,int)→Method Signature

```
class MSignature
{

        public void meth1(int i)
                {
                        System.out.println("method 1");
                }

        public void meth2(String s)
                {
                        System.out.println("method 2");
                }
        public static void main(String args[])
        {
                MSignature m=new MSignature();
                m.meth1(4);
                m.meth2("Saurabh");

}
}
```



```
class MSignature
{

	public void meth1(int i)
		{
			System.out.println("method 1");
		}

	public void meth2(String s)
		{
			System.out.println("method 2");
		}
	public static void main(String args[])
	{
		MSignature m=new MSignature();
		m.meth1(4);
		m.meth2("Saurabh");
		m.meth3(10.5);

}
}
```

Following compiler time error you are going to get:

```
F:\Java Code 2020>javac MSignature.java
MSignature.java:18: error: cannot find symbol
                m.meth3(10.5);
                 ^
  symbol:   method meth3(double)
  location: variable m of type MSignature
1 error
```

~~Question: Who is use going to use method signature?~~

~~Answer: compiler~~

~~Question: When the compiler will use the method signature??~~

~~Answer: while calling/invoking methods, for resolving method calls.~~

```
class Msign
{
        public void meth1(int i)
        {
                /////
        }
        public int meth1(int j)
        {
                retrun 10;
        }
```

is it valid or not???

Answer: Not valid.

```
F:\Java Code 2020>javac Msign.java
Msign.java:7: error: method meth1(int)
is already defined in class Msign
        public int meth1(int j)
                   ^
```

Within a class two methods with the same signature are not allowed.

Method Overloading:

Defining function again with: Same name, in same class, with different arguments it is known as **Method**

Overloading.
If we have to perform only one operation, having same name of the methods increases the readability of the program. Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs. So, we perform method overloading to figure out the program quickly.

Advantage: Method overloading **increases the readability of the program**.

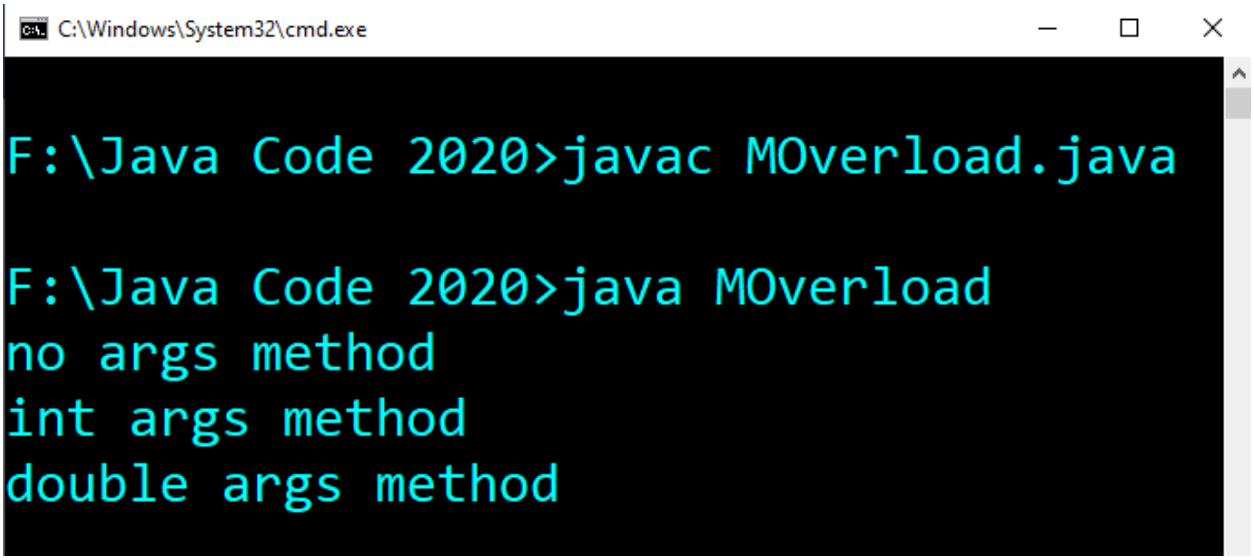
In C: Method overloading concept is not available.

```
class MOverload
{
public void m1()
{
System.out.println("no args method");
}

public void m1(int i)
{
System.out.println("int args method");
}

public void m1( double d)
{
System.out.println("double args method");
}

public static void main(String args[])
{
MOverload m=new MOverload();
m.m1();//
m.m1(5);
m.m1(7.5);
}
}
```



In overloading, method resolution is always takes care by compiler based on reference type.
Hence overloading is also consider as ~~Compile time polymorphism/Static polymorphism/Early binding.~~

Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

BY CHANGING THE NO. OF ARGUMENTS

In this example, we have created two overloaded methods, first sum method performs addition of two numbers and second sum method performs addition of three numbers.

```
class Calculation{
  void sum(int a,int b)
{
System.out.println(a+b);
}
  void sum(int a,int b,int c)
{
System.out.println(a+b+c);
}

  public static void main(String args[]){
  Calculation obj=new Calculation();
  obj.sum(10,10,10);
  obj.sum(20,20);

  }
}
```

Output:30
40

BY CHANGING DATA TYPE OF ARGUMENT

In this example, we have created two overloaded methods that differs in data type. The first sum method receives two integer arguments and second sum method receives two double arguments.

```
class Calculation
{
  void sum(int a,int b)
  {
System.out.println(a+b);
  }
  void sum(double a,double b)
  {
 System.out.println(a+b);
  }
  public static void main(String args[]){
  Calculation obj=new Calculation();
  obj.sum(10.5,10.5);
  obj.sum(20,20);

  }

```

```
Output:21.0
        40
```

```
class Calculation{
  int sum(int a,int b){System.out.println(a+b);}

double sum(int a,int b){System.out.println(a+b);}

  public static void main(String args[]){
  Calculation obj=new Calculation();
  int result=obj.sum(20,20);
  }
}
```

Output: Compile Time error

int result=obj.sum(20,20); //Here how can java determine which sum() method should be called

In java, method overloading is not possible by changing the return type of the method because there may occur ambiguity.

```
class MOverCase1
{
public void m1(int i)
{
System.out.println("Int arg");
}
 public void m1(float f)
 {
 System.out.println("Float arg");
 }
  public static void main(String args[])
  {
  MOverCase1 c=new MOverCase1();
  c.m1(10);// int args
  c.m1(20.5f);//float args
  c.m1('a');//
  c.m1(10 l);
  //c.m1(20.5);
  }
  }
```

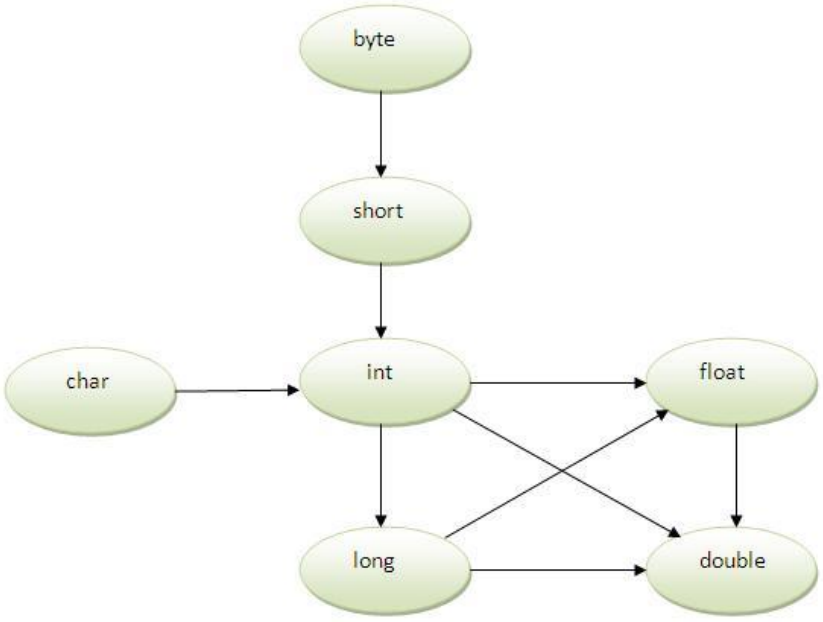
```
F:\Java Code 2020>javac MOverCase1.java

F:\Java Code 2020>java MOverCase1
Int arg
Float arg
Int arg
Float arg
```

```
F:\Java Code 2020>javac MOverCase1.java
MOverCase1.java:18: error: no suitable method found for m1(double)
  c.m1(20.5);
   ^
    method MOverCase1.m1(int) is not applicable
      (argument mismatch; possible lossy conversion from double to int)
    method MOverCase1.m1(float) is not applicable
      (argument mismatch; possible lossy conversion from double to float)
1 error
```

METHOD OVERLOADING AND TYPEPROMOTION

One type is promoted to another implicitly if no matching datatype is found. Let's understand the concept by the figure given below:



As displayed in the above diagram, byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int,long,float or double. The char datatype can be promoted to int,long,float or double and so on.

EXAMPLE OF METHOD OVERLOADING WITH TYPEPROMOTION

```
1.  class Calculation{
2.    void sum(int a,long b)
3.  {System.out.println(a+b);
4.  }
5.    void sum(int a,int b,int c)
6.  {System.out.println(a+b+c);
7.  }
8.
9.    public static void main(String args[]){
10.   Calculation obj=new Calculation();
11.   obj.sum(20,20);
12.   obj.sum(20,20,20);
13.
14.   }
15. }
```

```
Output:40
       60
```

If there are matching type arguments in the method, type promotion is not performed.

```
1.  class Calculation{
2.    void sum(int a,int b){System.out.println("int arg method invoked");}
3.    void sum(long a,long b){System.out.println("long arg method invoked");}
4.
5.    public static void main(String args[]){
6.    Calculation obj=new Calculation();
7.    obj.sum(20,20);
8.    }
9.  }
```

```
Output:int arg method invoked
```

If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

```
1.  class Calculation{
2.    void sum(int a,long b){System.out.println("a method invoked");}
3.    void sum(long a,int b){System.out.println("b method invoked");}
4.
5.    public static void main(String args[]){
6.    Calculation obj=new Calculation();
7.    obj.sum(20,20);
8.    }
9.  }
```

Output:Compile Time Error

```
class MOverCase2
{
public void m1(String s)
{
System.out.println("String arg");
}
public void m1(Object o)
 {
 System.out.println("Object arg");
 }
  public static void main(String args[])
  {
  MOverCase2 c=new MOverCase2();
  c.m1(new Object());
  c.m1("Saurabh");
  //c.m1(null);
  }
  }
```

```
F:\Java Code 2020>javac MOverCase2.java

F:\Java Code 2020>java MOverCase2
Object arg
String arg
String arg
```

Assignement Q.2:
Can we overload main method in java ?? Justify with example.