overloading:
Two methods are said to be overloaded iff both methods having the same name but different arguments is called overloading.

Example:

```
class Test
{
public void m1(int i)
{
}

public void m1(long l)
{
}

}
```

---

in C language,method overloading concept is not available hence we can't declare multiple methods with same name but different argument types

if there is a change in argument type compulsory we should go fornew method name which increses complexity of programming.

But in Java
we can declare muultiple methods with same name but different argument types such type of methods are called overloaded methods.

Having overloading concept in java reduces complexity of programming.

Ex1:Overloading with an example program

```
class Test1
{
	public void m1()
	{
		System.out.println("no-arg");
	}
	public void m1(int i)
	{
```

```
		System.out.println("int-arg method");
	}

	public void m1(double d)
	{
		System.out.println("double-arg method");
	}
	public static void main(String[] args)
	{
		Test1 t=new Test1();
		t.m1();
		t.m1(10);
		t.m1(10.5);

	}
}
```

output:

```
D:\Yellaswamy_ClassNotes\OverloadingEx>javac Test1.java

D:\Yellaswamy_ClassNotes\OverloadingEx>java Test1
no-arg
int-arg method
double-arg method

D:\Yellaswamy_ClassNotes\OverloadingEx>
```

In overloading method resolution always takes care by compiler based on reference type,hence overloading is considered as

compile time polymorphism

or
static polymorphism

or
Early binding

case1:Automatic promotion in overloading

class Test1

```
{
	public void m1()
	{
		System.out.println("no-arg");
	}
	public void m1(int i)
	{
		System.out.println("int-arg method");
	}

	public void m1(double d)
	{
		System.out.println("double-arg method");
	}
	public static void main(String[] args)
	{
		Test1 t=new Test1();
		t.m1();
		t.m1(10);
		t.m1(10.5);
		t.m1('a');//*******
		t.m1("cmrcet");
	}
}
```

---

while resolving overloaded method if exact matched method is not available then we won't get any compile time error immedately.
first it will promote argument to the next level and check whether matched method is available or not if matched method is available then it will be considered.

ifthe matched method is not available then compiler promotes argument once again to the next level this process will be continued until all possible promotions still if the matched method is not available then we will getcompile time error.

Ex2:

```
class Test
{
	public void m1(String s)
	{
		System.out.println("String Version");
```

```
	}
	public void m1(Object o)
	{
		System.out.println("Object Version");
	}
	public static void main(String[] args)
	{
		Test t=new Test();
		t.m1(new Object());
		t.m1("Yellaswamy");
		t.m1(null);
	}
}
```

whileresolving overloaded methods compiler will gives always precedence forchild type arguments,when compared with parent type arguments.

---

case 3:

```
class Test
{
	public void m1(String s)
	{
		System.out.println("String Version");
	}
	public void m1(StringBuffer sb)
	{
		System.out.println("StringBuffer Version");
	}
	public static void main(String[] args)
	{
		Test t=new Test();

		t.m1("Yellaswamy");

		t.m1(new StringBuffer("Yellaswamy"));

		t.m1(null);
	}
}
```

output:

```
D:\Yellaswamy_ClassNotes\OverloadingEx>javac Test.java
Test.java:19: reference to m1 is ambiguous, both method m1(java.lang.String) in
Test and method m1(java.lang.StringBuffer) in Test match
            t.m1(null);
             ^
1 error

D:\Yellaswamy_ClassNotes\OverloadingEx>
----------------------
case 4:
class Test
{
        public void m1(int i,float f)
        {
                System.out.println("int-float Version");
        }
        public void m1(float f,int i)
        {
                System.out.println("float-int Version");
        }
        public static void main(String[] args)
        {
                Test t=new Test();

                t.m1(10,10.5f);
                t.m1(10.5f,10);
        /*      t.m1(10,10);
        Test.java:17: reference to m1 is ambiguous, both method m1(int,float) in Test an
d method m1(float,int) in Test match
            t.m1(10,10);
             ^
1 error*/

t.m1(10.5f,10.5f);

        }
}


output:
```

```
D:\Yellaswamy_ClassNotes\OverloadingEx>javac Test.java
Test.java:24: cannot find symbol
symbol  : method m1(float,float)
location: class Test
t.m1(10.5f,10.5f);
 ^
1 error

D:\Yellaswamy_ClassNotes\OverloadingEx>
```