Method Overloading

Process of declaring multiple methods with same name and different parameters is known as method overloading. Parameters should differ either in the length or in the type. Method can be overloaded in the same class and also in the subclass.

In java private method, final method and static methods can be overloaded. In case of method overloading method name should be same and parameters should be different. Return type of method can be same or different.

In method overloading, the overloaded method can be called based on the type of argument. In java we can overload main method, but always JVM will invoke the method which takes string array as an argument.

Whenever if we want to perform same operations with different set of datas, then we have to go for method overloading.

```
public class OverLoadingAdd
{
	void add(int x,int y)
	{
		int z=x+y;
		System.out.println("The sum of integers: "+z);
	}
	void add(double x,double y)
	{
		double z=x+y;
		System.out.println("The sum of doubles: "+z);
	}
}
public class Main
{
	public static void main(String[] args)
	{
		OverLoadingAdd a=new OverLoadingAdd();
		a.add(1418,1984);
		a.add(1.4,1.6);
	}
}
```

2. Write a program to perform a login operation either by using username or phone number

```
public class OverLoading3
{
	String uname="Alex";
	String pwd="12345";
	long PhNo=123456;

	public void login(String uname, String pwd)
	{
		if((this.uname==uname)&&(this.pwd==pwd))
```

```
		{
			System.out.println("Login successful");
		}
		else
		{
			System.out.println("Login Unsuccessful");
		}
	}

	public void login(double PhNo, String pwd)
	{
		if((this.PhNo==PhNo)&&(this.pwd==pwd))
		{
			System.out.println("Login successful");
		}
		else
		{
			System.out.println("Login Unsuccessful");
		}
	}

}
```

```
package com.Overloading3;

public class Main
{
	public static void main(String[] args)
	{
		OverLoading3 l=new OverLoading3();
		l.login("Alex","12345");
		l.login(123,"12345");
	}
}
```

3. Write a program to display train details based on train name and train number.

```
package com.OverLoading4;

public class OverLoadingTrain
{
	String Tname="Tippu";
	String TdTime="12.45am";
	String TaTime="4:40am";
	long Tno=123456;

	public void Train(String Tname)
	{
		if((this.Tname==Tname))
		{
			System.out.println("Train Details: ");
			System.out.println("Train Name: "+Tname);
			System.out.println("Train number: "+Tno);
			System.out.println("Train departure: "+TdTime);
			System.out.println("Train arrival: "+TaTime);
		}
		else
```

```
                {
                        System.out.println(" Details not matching");
                }
        }
        public void Train(long Tno)
        {
                if((this.Tno==Tno))
                {
                        System.out.println("Train Details");
                        System.out.println("Train Name: "+Tname);
                        System.out.println("Train number: "+Tno);
                        System.out.println("Train departure: "+TdTime);
                        System.out.println("Train arrival: "+TaTime);
                }
                else
                {
                        System.out.println(" Details not matching");
                }
        }

}

public class Main
{
        public static void main(String[] args)
        {
                OverLoadingTrain t=new OverLoadingTrain();
                t.Train(123456);
                t.Train("tip");
        }
}
```

In Method overloading return type of method can be same or different.

Eg:

```
Class demo
{
Void m1(int x)
{
}
Int m1(int x,int y)
{
Return 10;
}
}
```

Method can be overloaded in same method or in the subclass

```
Class demo
{
Void m1(int x)
{
}
Int m1(int x,int y)
{
return 10;
}
```

```
}
Class test extend demo
{
Void m1(double z)
{
}
}
```

In a class we cannot declared multiple function with same name and same parameter

```
Public class demo
{
Public void m1(int x)
{
}
Public void m1(int x) // error a method cannot be declared with same name and
parameters
{
}
}
```

```
Public class demo
{
        Public static void m1(int x)
        {
        //static method
        }
        Private static void m1(int x)
        {
        //private method
        }
        final void m1(int x)
        {
        //final method
        }
}
```

Main method can be overloaded, it will not throw any error

```
Public class demo
{
Public static vois main(String args[])
{
}
Public static void main(int x)
{
}
```