Chapter 8 – Methods

What is a Method?

A group of statements into a signle unit is called as a Method.

Why do we use methods?

We use methods for code reusability and code maintanability

How do declare Methods in java?

Syntax:

Returnvaluedatatype/void methodName([method parameters])

{

Statements;

[return value];

}

Note: If we return any value from the method then we should use corresponding data type as return type

Note: If we don’t want to return any value then we should use void

Note: method names should starts with small letter, when ever if it contains multiple words then second word onwards first letter is capital letter.

Ex: Method with return type

Int demoMethod()

{

System.out.println(“hello”);

return 12;

}

Ex: Method without return type

void demoMethod()

{

System.out.println(“hello”);

}

Ex:

**package** pack1;

**public** **class** Methods

{

//method with no return type

**void** demoMethod1()

{

System.***out***.println("demo method1");

}

//method with return type

**int** demoMethod2()

{

System.***out***.println("demo method2");

**return** 10;

}

**boolean** login()

{

System.***out***.println("login method");

**return** **true**;

}

**void** logout()

{

System.***out***.println("Logout method");

}

**static** **void** display()

{

System.***out***.println("display");

}

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

{

// **TODO** Auto-generated method stub

Methods obj=**new** Methods();

obj.demoMethod1();

**int** a=obj.demoMethod2();

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

**if**(obj.login()==**true**)

obj.logout();

Methods.*display*();

*display*();

}

}

Note : Methods will execute whenever we call the methods only

Note: We can execute instance methods with class object Reference

Note: We can execute static methods by using class name or directly

Ex: methods with parameters

**package** pack1;

**public** **class** Methods

{

**static** **void** display(**int** a)

{

System.***out***.println("given value is "+a);

}

**static** **void** login(String userName,String pwd)

{

System.***out***.println("user name is "+userName);

System.***out***.println("password is "+pwd);

}

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

{

*display*(10);

*display*(100);

*display*(-100);

*login*("lokesh","lokesh password");

}

}

Types of Methods:

ReturnType Parameters

With with

With without parameters

Without with

Without without

1. With Return Type and with parameters

**package** pack1;

**public** **class** Methods

{

**static** **int** cubeValue(**int** i)

{

**return**(i\*i\*i);

}

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

{

System.***out***.println(*cubeValue*(-10));

}

}

1. With Return type and without parameter
2. **package** pack1;
3. **public** **class** Methods
4. {
5. **static** **int** cubeValue()
6. {
7. **int** i=10;
8. **return**(i\*i\*i);
9. }
11. **public** **static** **void** main(String[] args)
12. {
13. System.***out***.println(*cubeValue*());
14. }
16. }

3. Method without return type with parameter

**package** pack1;

**public** **class** Methods

{

**static** **void** cubeValue(**int** i)

{

System.***out***.println(i\*i\*i);

}

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

{

*cubeValue*(10);

}

}

4. Without return type and without parameter

**package** pack1;

**public** **class** Methods

{

**static** **void** cubeValue()

{

**int** i=10;

System.***out***.println(i\*i\*i);

}

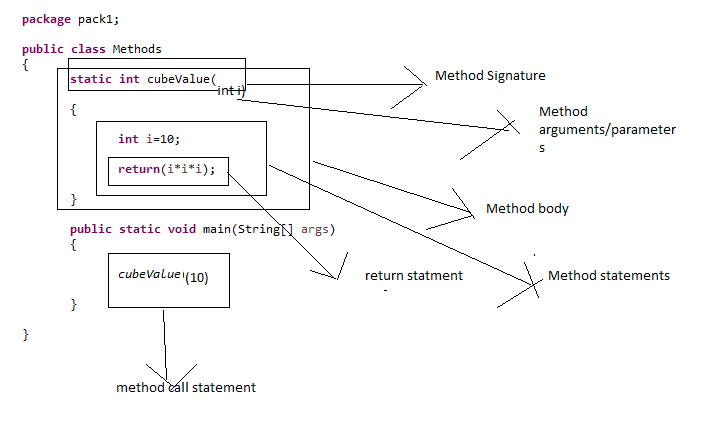
**public** **static** **void** main(String[] args)

{

*cubeValue*();

}

}



Method overloading and Method Overriding :

Method Overloading:

Declaring methods with same name and with different parameters is called as method overloading.

Ex:

**void** display()

{

System.***out***.println("no parameters method display");

}

**void** display(**double** a)

{

System.***out***.println("no parameters method display");

}

**void** display(**int** a)

{

System.***out***.println("one parameters method display");

}

**void** display(**int** a,**int** b)

{

System.***out***.println("one parameters method display");

}

Method Overriding:

Declaring methods with same name and same paramters is called as method overriding.

By default Method Overriding is not possible

Method Overrding is possible with interface

Ex:

**void** display(**int** a)

{

System.***out***.println("one parameters method display");

}

**void** display(**int** a)

{

System.***out***.println("one parameters method display");

}

Why Method Overriding is not possible?

Whenever we call the method then JVM will get confuse which method to call hence error