Methods in Java:

The action we perform to execute the logic is done by methods. Methods are of two types:

Syntax to define a method:

// Syntax:

// accessmodifier type\_of\_method return\_type name\_of\_method (arguments)

// {

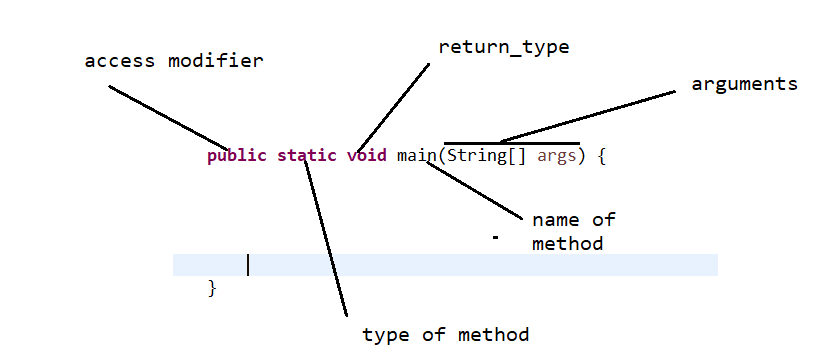
//

// }

a. Main method

b. Regular method

a. main method: This is a method which is responsible to decide the actual sequence of execution of java program.



b. Regular method:

There are two types of regular method:

i. static method:

ii. non static method

i. Static method:

syntax:

// Syntax:

// accessmodifier type\_of\_method return\_type name\_of\_method (arguments)

// {

//

// }

* To call static method we have to call it directly by name of method or class name.method name if it is in the same class.
* To call the static method which is in the **other class** we have to call it by classname.method name.

Example:

**public** **static** **void** methodOne()

{

System.***out***.println("executing method one");

}

**public** **static** **void** methodTwo()

{

System.***out***.println("executing method two");

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

}

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

*methodOne*();

*methodTwo*();

}

Example: Calling of static method from different class:

**public** **class** StaticMethods {

**public** **static** **void** methodOne()

{

System.***out***.println("executing method one");

}

**public** **static** **void** methodTwo()

{

System.***out***.println("executing method two");

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

}

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

*methodTwo*();

*methodOne*();

Test.*m1*();

*methodOne*();

StaticMethods.*methodTwo*();

}

}

**public** **class** Test {

**public** **static** **void** m1()

{

System.***out***.println("m1 method from Test class");

}

}

ii. non static method: In this method we don’t use any keyword to define the type of method.

// syntax:

// access\_modifier return\_type name\_of\_method ()

// {

//

// }

* Non static method can be called only after creation of object then call that method by reference name.methodname() whether it is in the same class or different class.

**public** **void** m1()

{

System.***out***.println("non static m1 method");

}

**public** **void** m2()

{

System.***out***.println("non static m2 method");

}

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

// object syntax:

// classname variablename = new classname();

NonStaticMethod p = **new** NonStaticMethod();

p.m1();

p.m2();

NonStaticMethod varname = **new** NonStaticMethod();

varname.m2();

varname.m1();

Test2 t2 = **new** Test2();

t2.addition();

Execution process of java program:

* JVM start
* Locate the .class file
* Load the .class file— memory allocation to static content
* Execute the main method—memory allocation and de-allocation to the non static content
* Unload the .class file – memory de-allocation to the static content.
* JVM shutdown.

Java is nearly 100% object oriented programming language because some content can be static.

|  |  |  |
| --- | --- | --- |
| **Sr. no** | **Static method** | **Non static method** |
| 1 | Static keyword is used to define the static method. | No keyword is used to define the method. |
| 2 | No need of object to call the method. | Compulsorily we have to call non static method by the help of object. |
| 3 | Memory allocation gets done at the time of class loading. | Memory allocation gets done at the time of execution. |
| 4. | It is not preferred because it doesn’t follow some of the oops principles e.g. overriding in polymorphism. | It is highly preferred as it follows oops principle. |
| 5 | Performance wise we don’t prefer it. | It is preferred with respect to performance. |

Calling of method inside another method:

a. Calling of static method inside another static method:

example:

**public** **static** **void** m1()

{

System.***out***.println("m1 method");

}

**public** **static** **void** m2()

{

*m1*();// calling of static method inside another static method

System.***out***.println("m2 method");

}

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

*m2*();

}

b. Calling of static method inside another non-static method:

example:

**public** **static** **void** m2()

{

*m1*();// calling of static method inside another static method

System.***out***.println("m2 method");

}

**public** **void** m3()

{

System.***out***.println("m3 non static method");

*m2*();// calling of static method inside non static method

}

c. Calling of non-static method inside static method:

example:

**public** **void** m1()

{

System.***out***.println("m1 method");

}

**public** **static** **void** m2()

{

CallingOfNonStaticMethod refname = **new** CallingOfNonStaticMethod();

refname.m1();// Calling of non static method inside static method

}

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

CallingOfNonStaticMethod var = **new** CallingOfNonStaticMethod();

var.m1();

}

d. Calling of non-static method inside another non-static method:

* If both methods are in the same class: We don’t need to create an object in the method in which are going to call the another one. We can call it directly by method name.
* If both methods are not in the same class: We have to create an object to call the method which is available in other class.

Example:

**public** **class** CallingOfStaticMethod {

**public** **void** m4()

{

System.***out***.println("m4 method");

}

**public** **class** CallingOfNonStaticMethod {

**public** **void** m3()

{

m1();// calling of non static method which is in the same class

CallingOfStaticMethod cosm = **new** CallingOfStaticMethod();

cosm.m4();// calling of non static method which is in another class

}

