**Static** keyword is used for almost same purpose in both C++ and Java. There are some differences though.

* (i) They can only call other static methods.
* (ii) They must only access static data.
* (iii) They cannot access t**his or super**

**1. Static Local Variables:** **Unlike C++,** Java doesn’t support static local variables. For example, the following Java program fails in compilation.

**class Test {**

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

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

**}**

**static int fun()**

**{**

**static int x= 10; //Compiler Error: Static local variables are not allowed**

**return x--;**

**}**

**}**

C++ Program: Static local variable

***void incrementAndPrint()***

***{***

***static int s\_value{ 1 }; // static duration via static keyword.  This initializer is only executed once.***

***++s\_value;***

***std::cout << s\_value << '\n';***

***} // s\_value is not destroyed here, but becomes inaccessible because it goes out of scope***

***int main()***

***{***

***incrementAndPrint();***

***incrementAndPrint();***

***incrementAndPrint();***

***return 0;***

***}***

Output:2 3 4

**Program:2: So a reference to static variable can be returned.**

int &fun() {

static int a = 10;

return a;

}

int main() {

int &y = fun();

y = y +30;

cout<<fun();

return 0;

}

**2.Static Block: Unlike C++,** Java supports a special block, called static block (also called static clause) which can be used for static initialization of a class.

Static Member in C++

A static member is shared by all objects of the class. All static data is initialized to zero when the first object is created, if no other initialization is present. **We can't put it in the class definition but it can be initialized outside the class as done i**n the following example