**1.**

**🡪Static member can be used anywhere IIB, SIB, non-static block, constructor in the current class.**

**2.**

**🡪Non-static members cannot be used straight way inside the static member for this we required reference variable.**

**🡪In order to use non-static variable inside the static member we should required reference variable.**

**3.**

**🡪Inside the non-static body, we can able to use any member of the current class.**

**4.**

**🡪 Inside the body of the static member we can use only static member of the current class.**

\*\*inside the static method we can use only static member of the current class.

**\*\* Inside non static inner class we cannot develop any static member.**

**class** D

{

**class** E//non static inner class

{

**int** i;

**void** test1()

{

}

//static int j;//static attribute

/\*

static void test2()//static method not allowed inside the non static inner class

{

}

\*/

}

}

/\*

Compiler will not allow.\*/

***\*\*Inside the static inner class we can develop any members***

***Inside the static inner class we can only used the outer class static member.***

**class** E

{

Static int k=10;

Int p=10;

**static** **class** F

{

**int** i;

**static** **int** *j*;

**void** test1()

{

*Syso(k);*

*//Syso(p); cannot used because of non static*

}

**static** **void** test2()

{

}

}

}

\*\* Non-static member can be used in any of the non static method.

\*\*inside non-static inner class we cannot develop (declare/initialize) any static member.

\*\* We can only use static and non-static member of the outer class.

\*\* We can only develop the non-static member inside the non-static inner class.

***\*\*if the class is uses for data type purpose then we can use that class anywhere.***

**public** **class** G

{

**class** H

{

}

**static** **class** I

{

}

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

{

H h1 = **null**;//h1 is of H type ,here H is using for data type purpose.

I i1 = **null**;

i1 = **new** I();

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

}

}

**class** G1

{

**class** H1

{

}

**static** **class** I1

{

}

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

{

H1 h1 = **null**;

I1 i1= **null**;

h1 = **new** G1().**new** H1();

i1 = **new** I1();

}

}

/\*

if we want to create the object of class H1 inside the main method we need outer class reference variable e.i outer class OBJECT from G Object we are creating H1 OBJECT

WE can create the object of static inner class

\*/

**There are four type of inner class.**

**1. static inner class**

**2. non-static inner class.**

**3. local inner class.(LIC is very similar to the local varibale)**

**4. Annonymous inner class.**

**public** **class** I

{

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

{

**class** A//L-I-C

{**int** i = 10;}

A a1 = **new** A();

System.***out***.println(a1.i);

a1.i=20;

System.***out***.println(a1.i);

}

}

**public** **class** K

{

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

{

System.***out***.println("1111");

System.***out***.println("1111");

System.***out***.println("1111");

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

System.***out***.println("1111");

System.***out***.println("1111");

System.***out***.println("1111");

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

System.***out***.println("1111");

System.***out***.println("1111");

System.***out***.println("1111");

}

}

**/\***

**Here common code is executed three times here we are not achieving the reusebility of code is not organized here.**

**\*/**

**public** **class** M

{

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

{

**class** A

{

**void** test()

{

System.***out***.println("1111");

System.***out***.println("1111");

System.***out***.println("1111");

}

}

A a1 = **new** A();

a1.test();

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

a1.test();

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

a1.test();

}

}

/\*

Here we are achieving the security because common statement is getting used inside the main method only

\*/

By using L-I-C we are able to use reusability of test() method and also method level security we are achieving.

\*\* Nesting of method can be allowed in java. It should be the part of inner class.

\*\* L-I-C can be a part of following

IIB

CONSTRUCTOR

SIB

INSIDE STATIC INNER CLASS

INSIDE THE NON STATIC INNER CLASS

\*\* If we want use local variable inside the L-I-C then the local should be final.

**public** **class** O

{

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

{

**int** i = 10;

**final** **int** j = 20;

**class** A

{

**void** test()

{

//System.out.println(i);

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

}

}

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

/\*A a1 = new A();

a1.test();\*/

}

}

/\*

if we want to use local variable inside the local inner class then that variable should be declare as final

\*/

**public** **class** P

{

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

{

**class** A

{

//static int i;

}

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

}

}

/\*

L-I-C is almost like instance inner class and we cannot develop static variable

inside the l-n-c

\*/