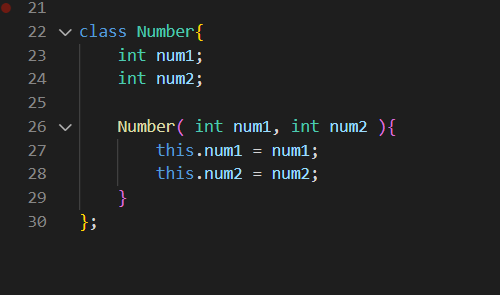
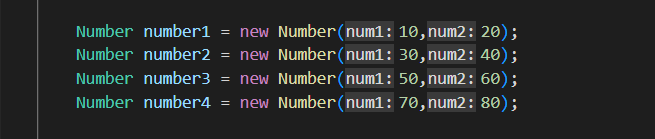
What does it mean that a class has non static variables inside it ?

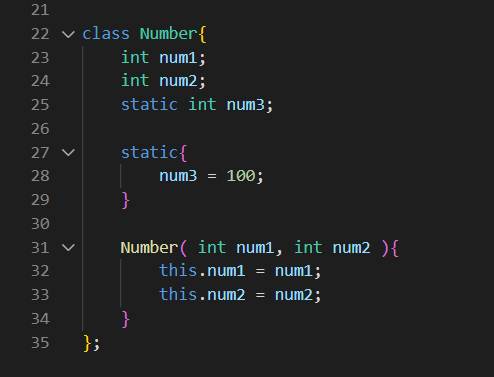
It means that everytime an object is created each object will have it’s own unique those variables and their unique values in the memory



Here this is a Number class



Everytime we create objects like this we are having each objects in memory and each of them have their own unique values of num1 and num2



Now we have num3 as static variable

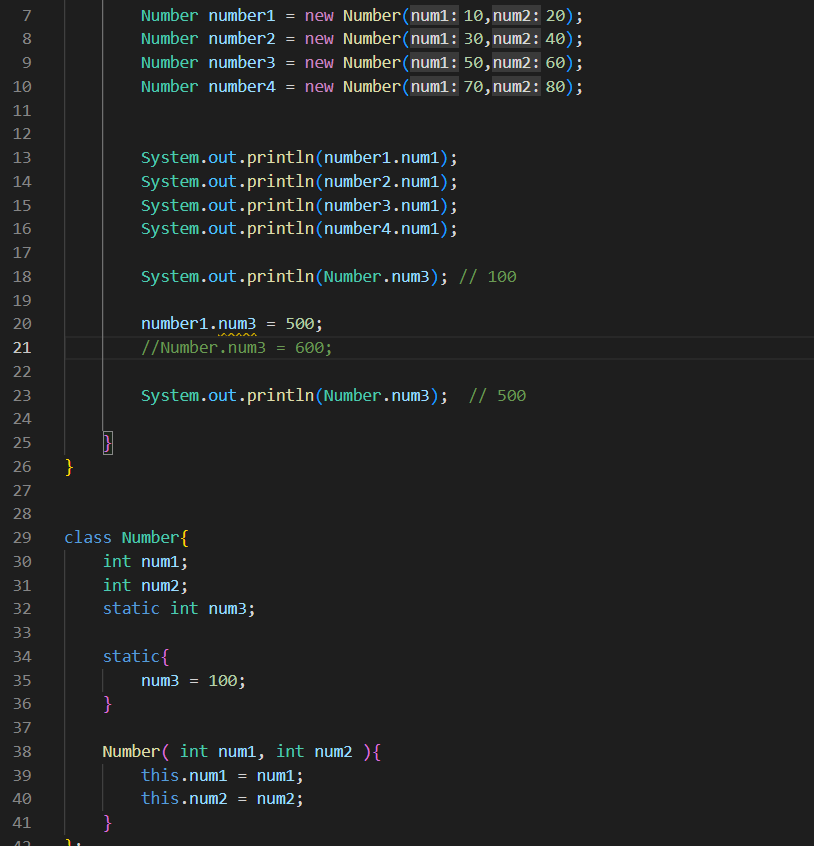
Now what happens is, this is stored at separate place in the heap and is shared by all the instances

So in other words this means this static num3 has no any dependencies with object, it does not depend upon object

This is created the moment Class is loaded

**But this is not created in the each object in the memory but is stored on only one place and is shared by all the instances of the class**

But the instances of the class can change the value of this static variable, if that happens the new value becomes available to all the instances of the class



here static variable num3 value is changed by number1 , an instance of Number class

But this is a bad practice

So always use the Class Name for dealing with the static variables, like shown in comment

The static variables are those variables that are common to all the instances of the class; only a single copy of the static variable is created and shared among all the instances of the class. Because it is a class-level variable, memory allocation of such variables only happens once when the class is loaded in the memory.

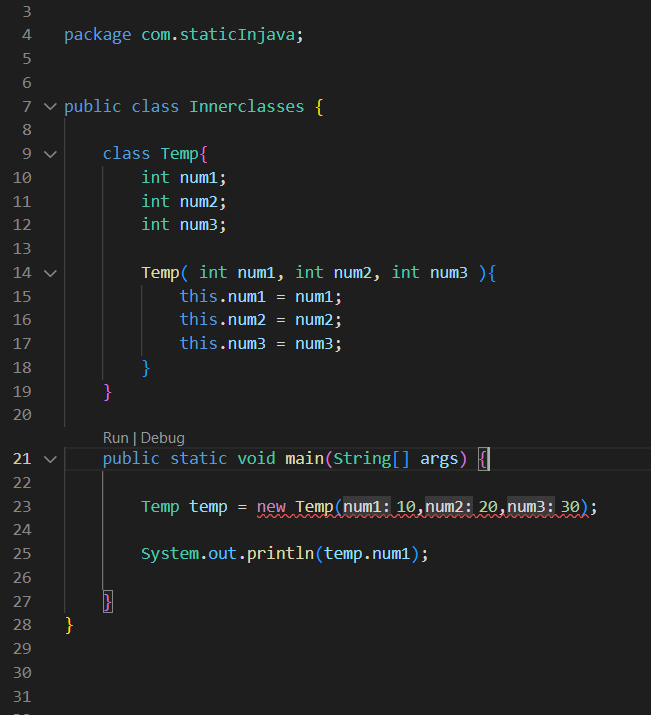
**Conclusion**

A static variable is stored at separate place only once but becomes available to all the instances, so we say it does not depend upon the objects

Always use class name to change and access the static variable never use the object’s name

**In Java the outer class can not be static**

Java's creators had not allowed an outer class to be static because there is no need to make it static. Allowing to make the outer class static will only increase complications, ambiguity, and duplicity.



**Why were we not able to make object temp of Temp class here ?**

**Reasoning 1 :**

here we were not able to make the object temp of the class Temp which is inside the class Innerclasses

**A class is an template a blue print of an object, until and unless an object is created the variables, methods and everything inside the class does not exist anywhere in the memory, unless an object is created so we must provide an object to make temp object**

Only when the object is created the memory is allocated for the variables, methods and other things inside the class

So here when ab object is Innerclasses is created then only that object will have Temp class in it and we can create object of Temp class like this

**Reasoning 2 :**

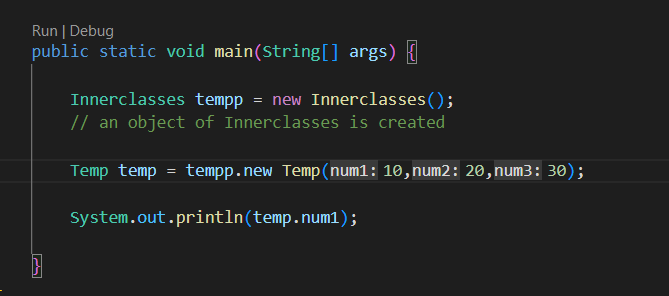
here when objects of Innerclasses class would be created each one will be different object, so each one will have Temp class defined in it

So when we do 

it is not sure that Temp class of which object is creating temp object

*( like jun object ko Temp class pani hunaa sakxaa, kunai pani hunaa sakdainaa yedi eautaa pani object create vako xainaa vannae, so yesari object banaunaa java lae didainaa. Yaa but yedi Innerclasses ko object banayo vannae chai dinxaa, kinaa ki java lai thaha vayo ki yo object bhitra ko Temp class ko object banaunae vanaeraa )*

We must mention the Innerclasses object’s name, on doing that it would be clear that we are refering to the Temp class, defined in that object so we would be able to make temp object



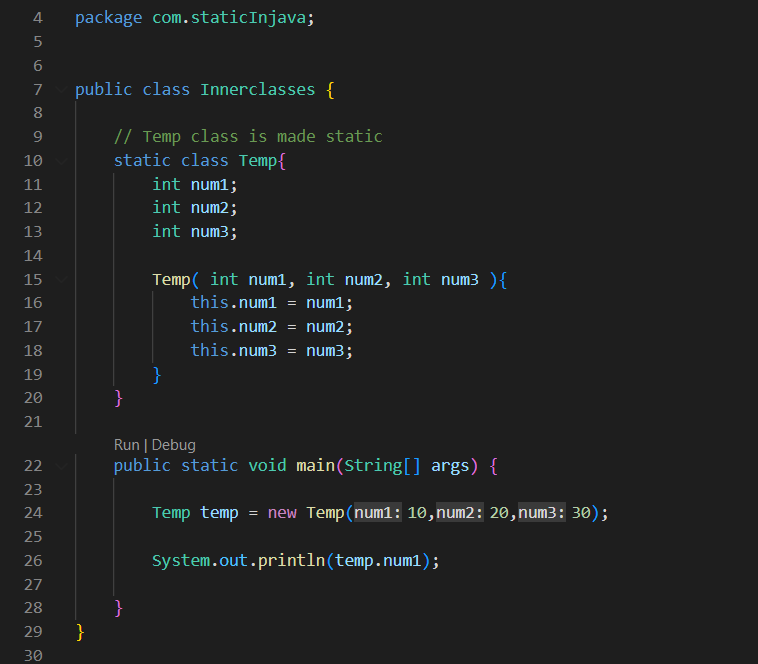
**Here we can also reason that all the non static members inside the class are object dependent so we must provide the object’s name ( actually reference variable to an object ) to work with them. So here when we provided that object’s name tempp we were able to work with the Temp class**

Now here it is clear that we are making temp object refering to the Temp class defined in the tempp object of Innerclasses class

**Reasonong 3**

When we make the inner class static, we would be able to make object without any problem

**When the Temp class is static it does not depend upon the object of the Innerclasses, so we do not need any object of Innerclasses to work with the Temp class**



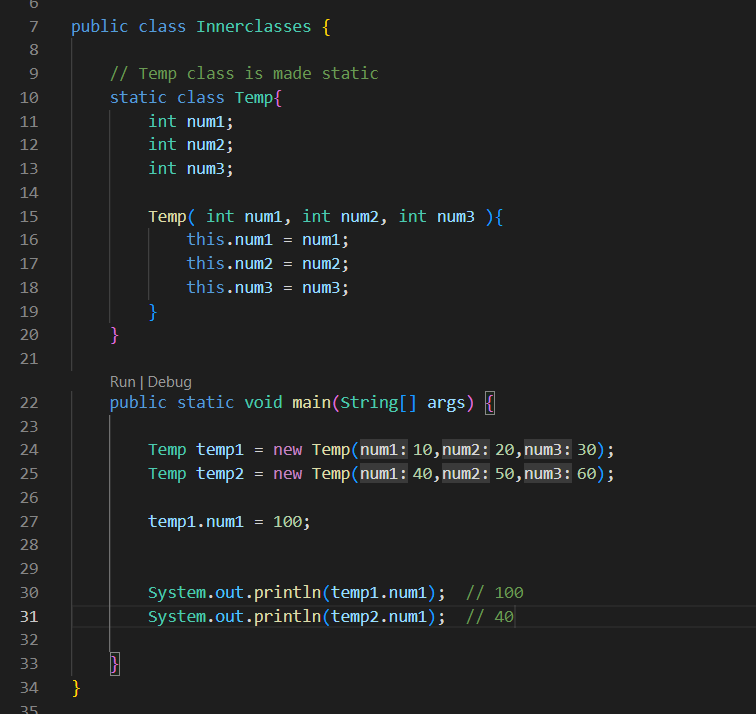
here Temp class is made static

so, Temp class is a static member of Innerclasses class

And we know static members lives in the different part in the heap memory only once the moment the class is loaded during compilation, and their copies goes to each objects when they are created, unlike to non static members which are created on creation of every new object and can be different from each other

So now it’s clear that Temp class is static member of Innerclasses class, java know where the class is located so it will easily make the temp object

**Here only the Temp class is static, the members inside the Temp class are still non static ?**



here only temp class is static the members inside the Temp class are not static

*Kunai pani class ko member static hunxaa vannae, compiling hunae bela tyo class load hudaa tyo static member variable heap ko eautaa special memory maa gayaeraa basxaa. Also yoo time samma objects bannie sakyaa hudainaa. Objects run time maa banxaa. Eautaa class ko jati pani objects haru banna sakxaa, let say 100 oota objects banyo. Now tyo 100 ootai objects ko reference variable chai stack maa basxaa ani tyo reference variables haru lae heap ko tyo 100 objects haru lai point garxaa*

*Now non static variables haru chai harek objects maa bannie raa hunxaa, meaning kunai eautaa variable non static xaa vannae yo case maa 100 objects maa 100 ootai tyo variable banyaa hunxaa, eautaa object maa eauta garaeraa*

*But static variable chai aaginai compile time mai, object naa bandaa nai eautaa special thau maa gayaeraa basaeko hunxaa. Now aba 100 oota objects banae paxi tyo static variable chai 100 oota objects lain ai available hunxaa*

*So yesari eautaa matrai thau maa memory occupy garxaa raa sabai objects maa available hunxaa, so static variables are not dependent to objects vanaeko, they are class dependent*