**Naming Convention**

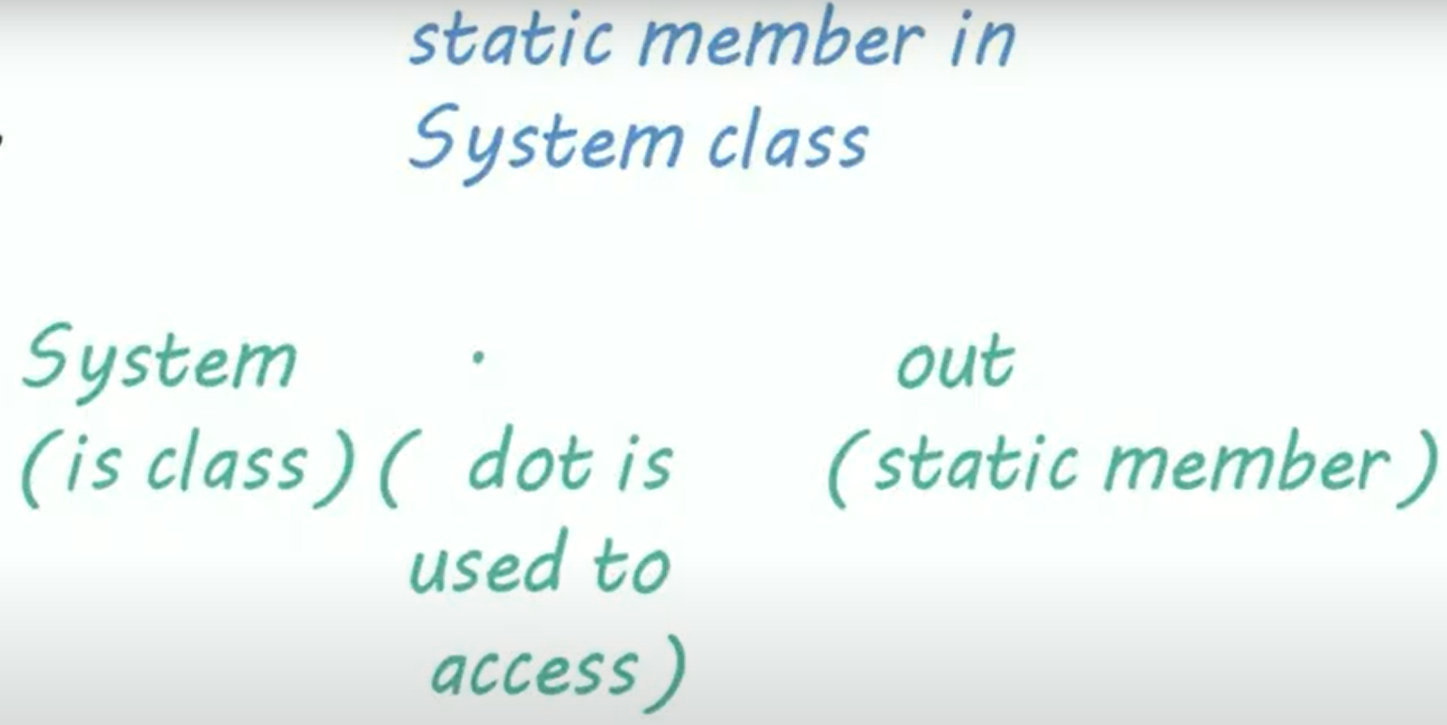
****

**System.out.print(“hello”);**

System – prdefined class in java

out – static member in System class (static members can be accessd by their classname, with “.” operator.

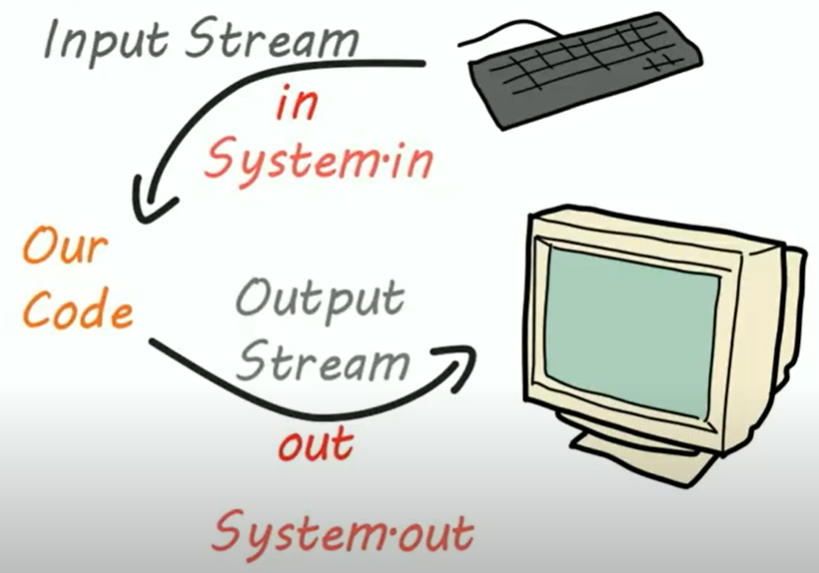
print() - method/function to print the message.



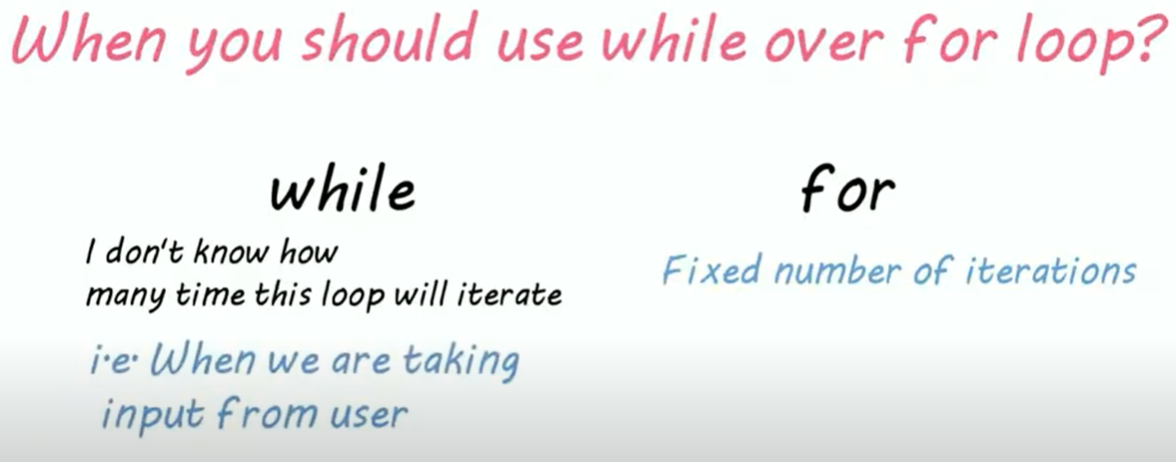
.

\*\*out is standard output stream, used to give output to the user.

J



**When to use for vs when to use while loop**



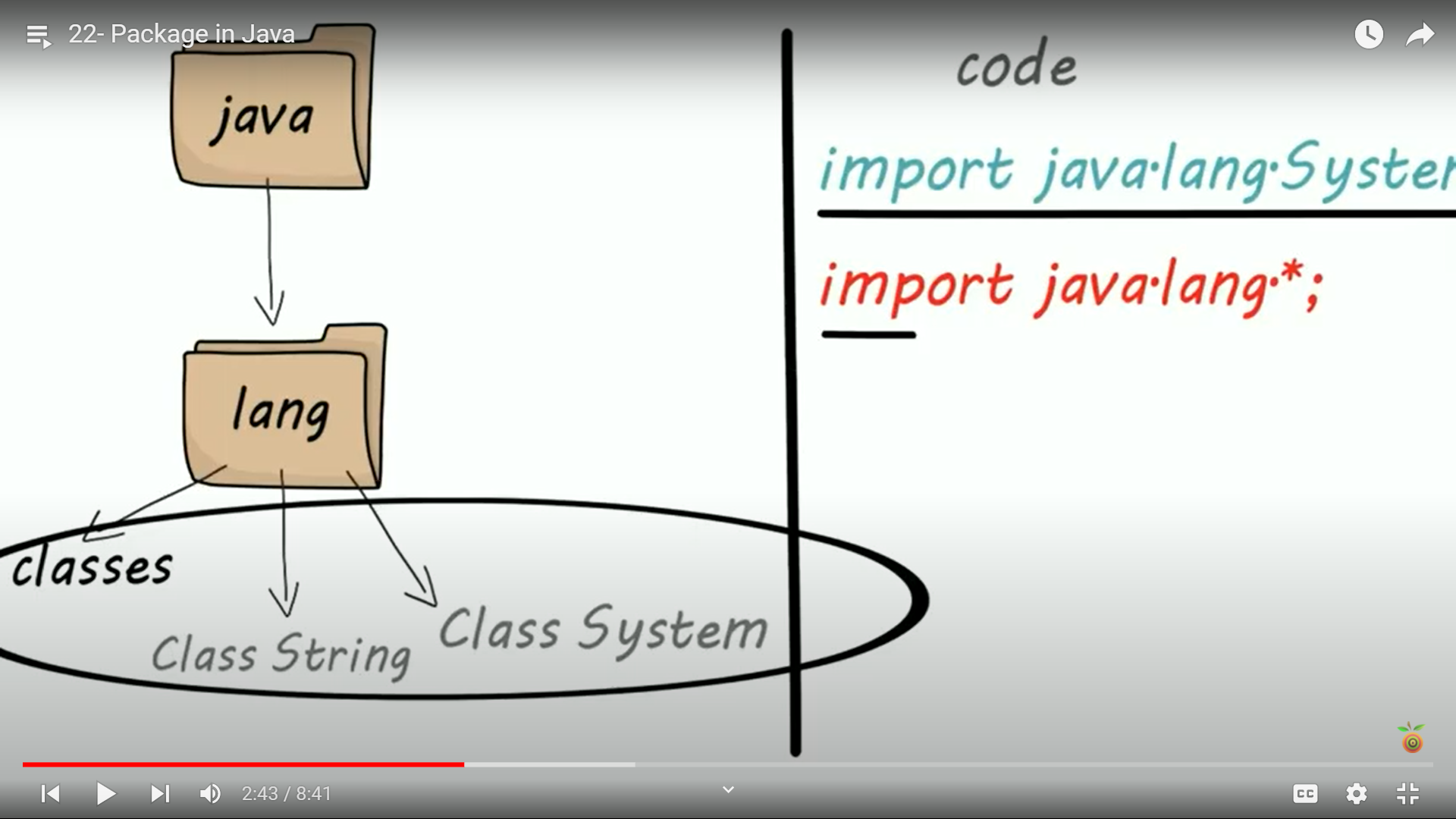
**Why we need packages ?**

To better organize and increase reusability and readability of our code.

Two types of packages

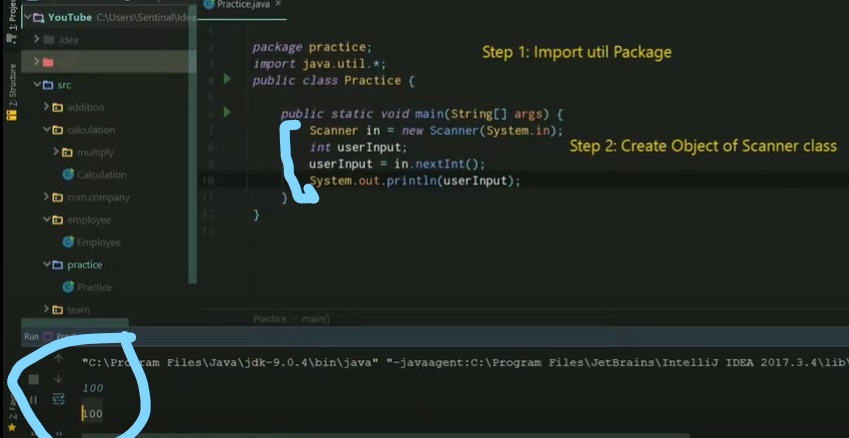
- built in packages (such as java, lang, awt, javax, swing, net, io, util, sql etc)

- user defines packages (the ones we make)

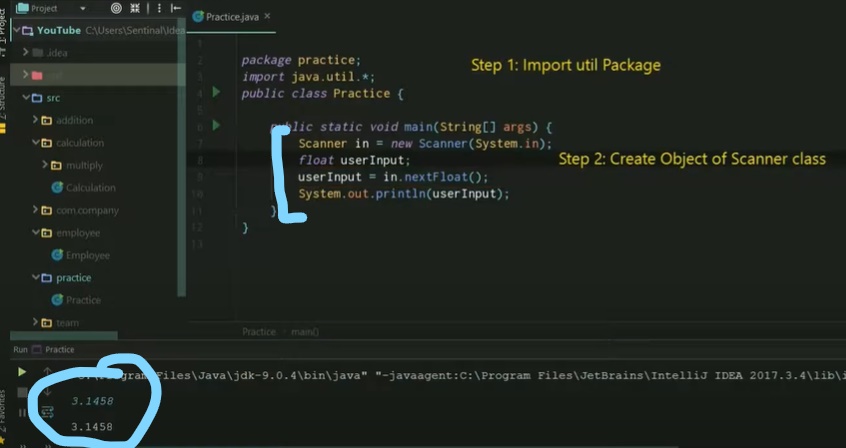


**Taking input from user (Scanner class)**

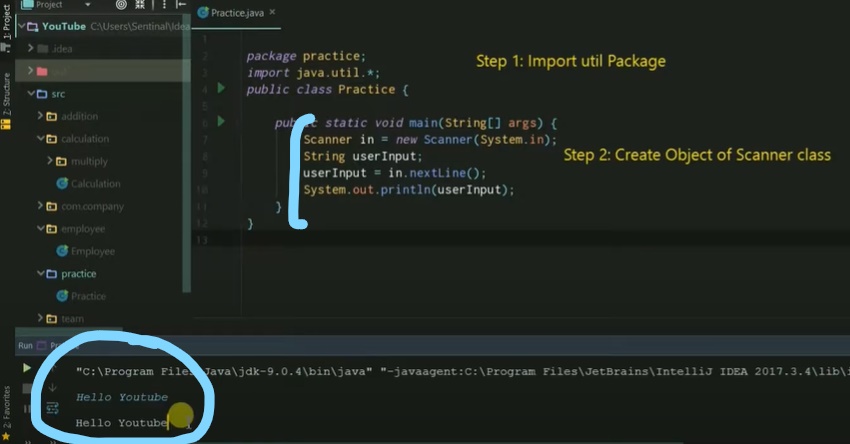
**int**



**float**



**strting**



**Diff b/w Stack Memory vs Heap Memory**

Stack Memory in Java is used for static memory allocation and the execution of a thread. It contains primitive values that are specific to a method and references to objects that are in a heap, referred from the method.

When the method finishes execution, it’s corresponding stack frame is flushed, the flow goes back to the calling method and space becomes available for the next method.

\*\*If this memory is full, Java throws java.lang.StackOverFlowError

Heap space in Java is used for dynamic memory allocation for Java objects and JRE classes at the runtime. New objects are always created in heap space and the references to this objects are stored in stack memory.

\*\*If heap space is full, Java throws java.lang.OutOfMemoryError

**Why the main () method in Java is always static?**

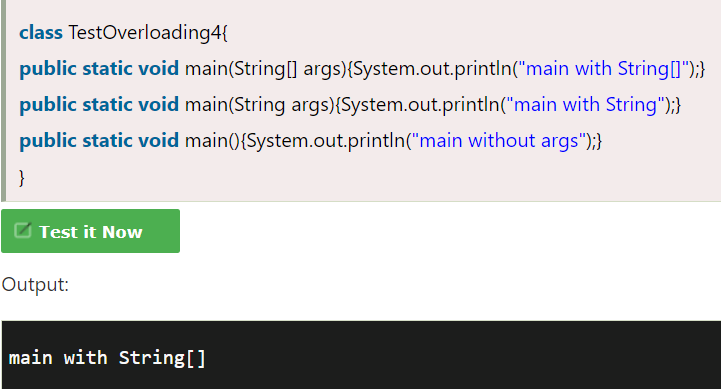
Java main() method is always static, so that compiler can call it without the creation of an object or before the creation of an object of the class. As Static method of a class can be called by using the class name only without creating an object of a class.

**Why Non-Static variables does not work in Static method in Java?**

For the non-static variable, there is a need for an object instance to call the variables. We can also create multiple objects by assigning different values for that non-static variable. So, different objects may have different values for the same variable.

**Can we overload java main() method?**

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main() method which receives string array as arguments only. Let's see the simple example:



**Can we override java main method?**

No, because the main is a static method.

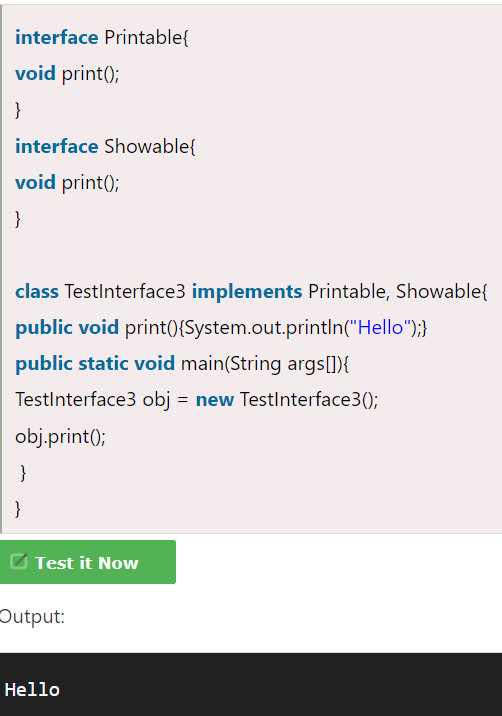
\*\*Static methods can not be overridden because they are not part of the object's state. Rather, they belongs to the class (i.e they are class methods). And as overriding a method is done by object prefernce, hence we can’t override static method.

**Is final method inherited?**

Yes, final method is inherited but you cannot override it.

**Multiple inheritance is not supported through class in java, but it is possible by an interface, why?**

As we have explained in the inheritance chapter, multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.



As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface1, so there is no ambiguity.

**Call by Value and Call by Reference in Java**

- There is only call by value in java, not call by reference. If we call a method passing a value, it is known as call by value. The changes being done in the called method, is not affected in the calling method.

