**Hint:**

Clone() method is protected method which will call only inside this class or inside that package where it got declared and defined as well as also inside those classes who is the child classes of that class where this clone() method got declared and defined.

Main important thing is that without implementing Cloneable interface this method can be written but that is the proper way for creating clone object and also using this process we can’t create clone of any class objects. Actual process is that first implement this Cloneable interface which is Marker Interface. After implementation of this interface then we can assigned this class object or this class object now qualify for creating clone objects. After this we can create clone object of this class.

Marker Interface is the interface inside which there is no method available but marking that class as Cloneable or Serializable type class. Mean that class object now able to serialize or able to create clone.

This clone() method is not available inside Cloneable interface. This method is belongs to Object class. Object class first check that this class is able to qualify for cloning operation or not. After checking this Object class will allow to override clone() method. Otherwise it will throws CloneNotSupportedException.

Q: Cloning Operation is Secure or non-Secure?

Ans: It is an unsecured operation. So for this reason runtime environment provide it to mark to that operation. That mark is expecting from the developer while developing this program. So developer can provide mark through marker interface. Marker Interface is an interface which doesn’t have any method. So there are many marker interface available in java.lang.\* among them Cloneable interface is one of the Marker Interface which mark that class to allow to create clone objects.

Serializable Interface is also marker interface which is used for serialization process in java. This operation is also unsecure operation.

Q: Without using new keyword how to create a new object of a class?

Ans: Using Cloning operation we can create a new object of a class without using new keyword.

**Hint:**

Deep Copy:

If we create a copy from an original object of a class and if original and duplicate object both are not dependent each other or disjoint from each other, then that operation is called as deep copy operation. This operation can happens in Wrapper class, String class and primitive or attributes also or either combination of these String class, Wrapper class or primitive or attributes. In Object class also deep copy operation is provided.

Example:

**package**com.lara.ObjectNotes;

**publicclass** ONotes58 **implements**Cloneable

{

StringBuffersb = **null**;

**publicstaticvoid** main(String[] args) **throws**CloneNotSupportedException

{

ONotes58 on = **new**ONotes58();

on.sb = **new**StringBuffer("sailendra");

ONotes58 on1 = (ONotes58)on.clone();

//Here we are trying to modify the original value of sb and trying to put some new value into it after cloning operation.

on1.sb = **new**StringBuffer("Bhagyalaxmi");

System.*out*.println("on.sb: "+on.sb);

System.*out*.println("on1.sb: "+on1.sb);

}

}

/\*

Here in this program we have created two types of different object. One is created using new keyword and another is using cloning operation. Here both objects are belongs to same

class but there is not dependency with each other. Here this operation is called deep copy operation. Here original object and clone object are disjoint with each other. So clone

can't be modify original object value.

Here is the output of this program:

on.sb: sailendra

on1.sb: Bhagyalaxmi

\*/

**Hint:**

append() Operation:

In this operation JVM add the content of original object into copy/cloneableobject and modify the original object value.

**Shallow Copy:**

Through this operation we are creating a clone/copy object which is dependent with original object or both are having not 100% disjoint or both are having joint each other, then that operation is called shallow copy operation. Here both original and clone or copy object is dependent with each other.

* If both object is primitive or mutable types objects then in that case it is deep copy.
* If object references or attributes of the class is not cloneable data types then that using that object operation for modification process is call shallow copy process. Check Example ONotes59Append.java in this program we have created one class name H5 which is not implements Cloneable interface. But this class objects reference we have used in class ONotes59Append. So in this program operation is called as shallow copy operation. Here in this operation we can modify the value of data any one of the object either original or clone objects.
* If the attributes or data types are Cloneable type then that operation is called deep copy operation. Here in this case we can’t modify the data except original objects. Using original object we can modify value of data.

**finalize():**

Here final and finally is a keyword and finalize() is a method in the Object class.

It is mainly used for restrict some specialization.

final variable can’t be re-initialized, final method can’t be override, final class can’t be subclass.

This method is actually available inside Object class. While we are calling System.gc(); method in our code then in that time JVM will understand that we have to call finamelize() method internally.

The purpose of this method is that when we are creating any object in our class and after use of that object we are making that object as an abandoned object by removing reference from that object. Then in that case we have to clean our heap memory to for space and for also keep new object into it. So this method is need for this case clean heap memory. While we got to know that this object is no more need then in that case we can inform to Garbage Collector for remove this object. Here we can’t either give any commands or instruction or any order to Garbage Collector to clean this object from the memory. We are just to inform or request to Garbage Collector for removing this objects.

(This method is available inside Object class. It will be used/called by Garbage Collector whenever an object is about to sweep out or remove from the memory. Whenever object is abandoned then Garbage Collector remove that object by calling finalize() method.)

* If an object has no reference from the stack then that object is called abandoned or terra object and which is eligible for Garbage Collector to remove this object.
* Once object got abandoned then we can’t reach to that object.
* Garbage Collector removes abandoned objects from the heap memory not the live objects.
* When we are keeping one object as an abandoned object that not mean that Garbage Collector will come and remove objects. When Garbage Collector will come to memory location and find any abandoned Objects into this location then it will start removing those objects not live objects by calling finalize() method. If we want to explicitly give request to Garbage Collector regarding this removing objects then we should go for this below statement.

System.gc();

Runtime.getRuntime().gc();

* We can’t either give any kind of command or any kind of order or any kind of instruction to Garbage Collector. We can just request this Garbage Collector regarding this removing objects from memory location.
* Garbage Collector will not depend on us means if we are request him regarding this removing objects from memory then it can either remove whatever we have mentioned or given data to it or might be more than ours objects.
* System.gc():

This gc() method is available inside System class. This gc stands for Garbage Collector.

* Runtime.getRuntime().gc():

We can also call this gc() method by using Runtime environment.

Here these process are using for same purpose. Only one different between these are working for same reason i.e for Garbage Collector.

* All objects are able to call this finalize() method only for once per object for calling Garbage Collector.

**getClass():**

By using this getClass() method we are using indirectly reflection api. Inside java.lang.\* package there is a class name as “Class” is available which is used for checking class information.

Every class is having one this Class object for retrieve information about its class. For retrieving class object information then we required this getClass() method through which we can fetch the class related information. This method actually available inside this Object class.

Here we can also load class in main method. Like this below statement,

Class.forName(“package name with class name”);