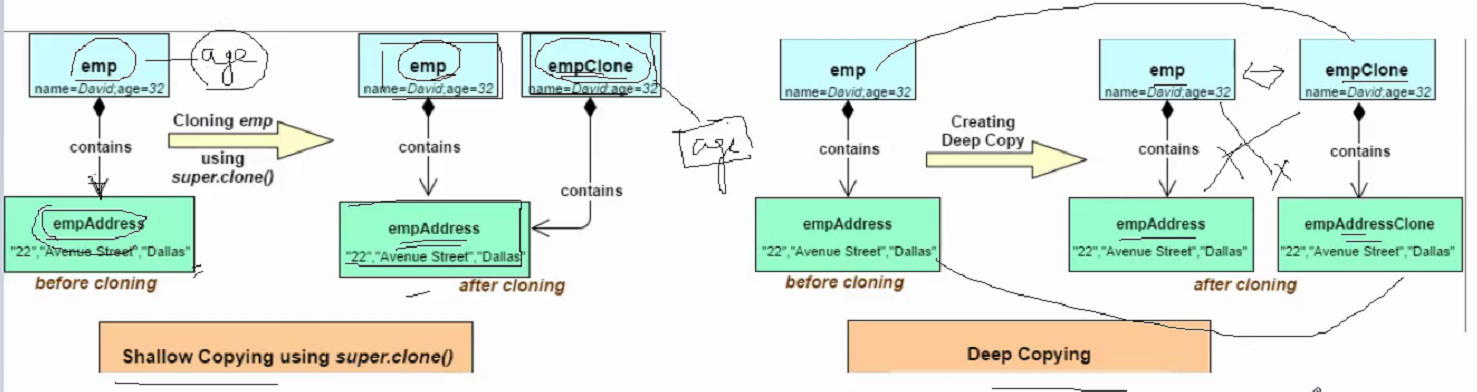
**What is Cloning**

Object cloning refers to creation of exact copy of an object. It creates a new instance of the class of current object and initializes all its fields with exactly the contents of the corresponding fields of this object.

**Shallow cloning** – is when the reference is pointing on the same reference. So if you change object 1 it will also change the object 2.

**Deep cloning** is its own reference. So both object does not affect each other.



**When to do shallow copy and deep copy?**

Its very simple that if the object has only primitive fields and reference of immutable objects, then obviously you will go for shallow copy but if the object has references to other objects, then based on the requirement, shallow copy or deep copy should be chosen. What I mean here is, if the references are not modified anytime, then there is no point in going for deep copy. You can just opt shallow copy. But if the references are modified often, then you need to go for deep copy. Again there is no hard and fast rule, it all depends on the requirement.

**role of Cloneable ?**

Cloneable is also a marker interface. The object to be cloned must implements Clonable interface weather if its shallow or deep cloning. When the Object class finds that the object to be cloned is not an instance of a class that implements Cloneable interface, it throws CloneNotSupportedException.

Cloneable is a marker interface

how a class allow clients to create its clone?

If a class wants to allow clients to clone it's instances, it must override Object's clone method with a public modifier.

clone( ) is a protected method

can we say serialization a way of implementing deep cloning?

Yes.........

Why do we need serialization?

● To transfer object over a network.● To persist object in a file system or a database.● To give an object an existence outside the lifetime of JVM.

**Which interface we have to implement to serialize an object of a class?**

Serializable interface or Externalizable interface.

**What is Transient keyword?**

When we serialize an object all its instance variables get serialized, if we don’t want some instance variables to participate in the serialization process we should use transient keyword with them.

**Can we serialize a static variable?**

Static variables belongs to the class, not to the objects, therefore they are not participate in the serialization process. But it is possible to serialize a static variable using a custom serialization process.

**What is volatile keyword?**

If we use volatile keyword with an instance variable, the value of that variable will be always read from main memory not from cache.

If we use volatile keyword with an instance variable, the value of that variable will be always read from main memory not from cache.

from Synergisticit Synergisticit to everyone:

If we don’t want subclass to be serializable what should we do?

from Synergisticit Synergisticit to everyone:

Implement readObject and writeObject in subclass and throw NotSerializableException.

from Synergisticit Synergisticit to everyone:

have a break

from Synergisticit Synergisticit to everyone:

To Find Duplicate Characters In A String In Java :