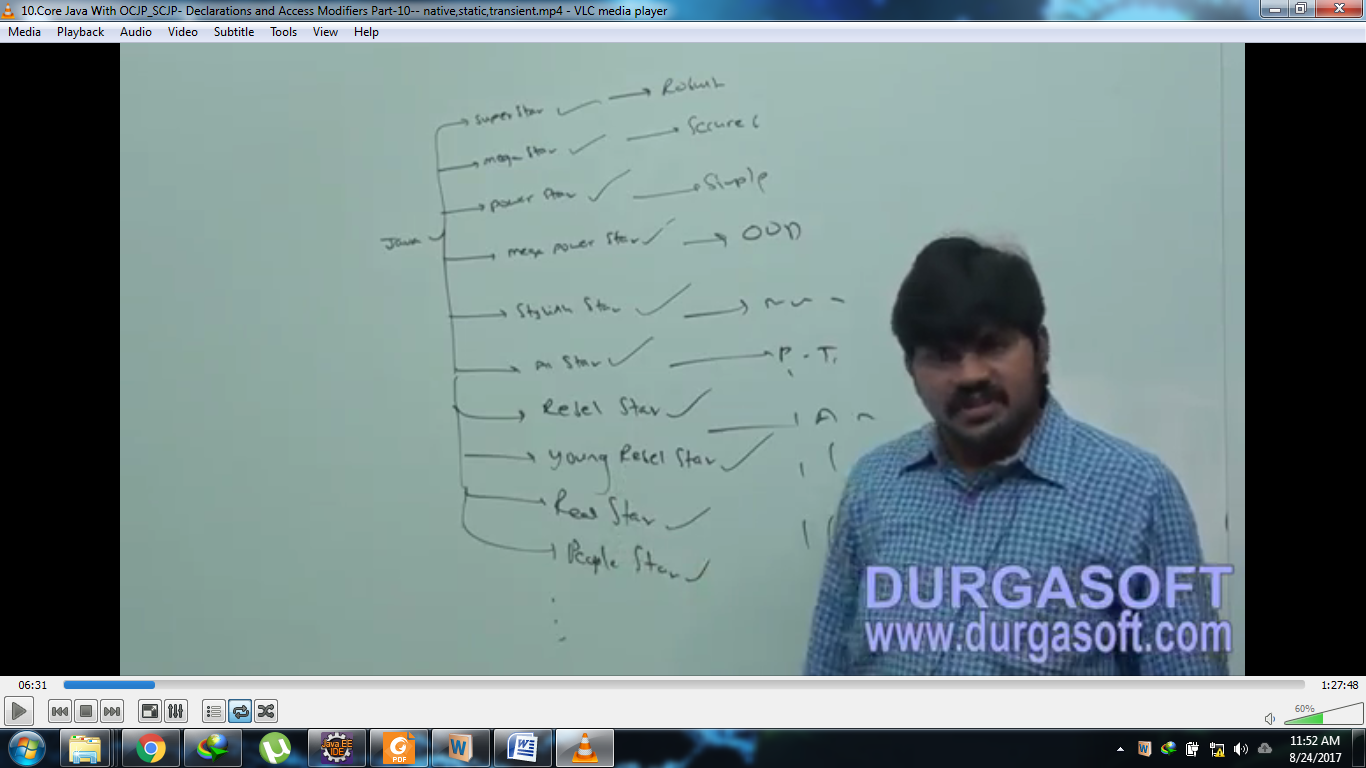
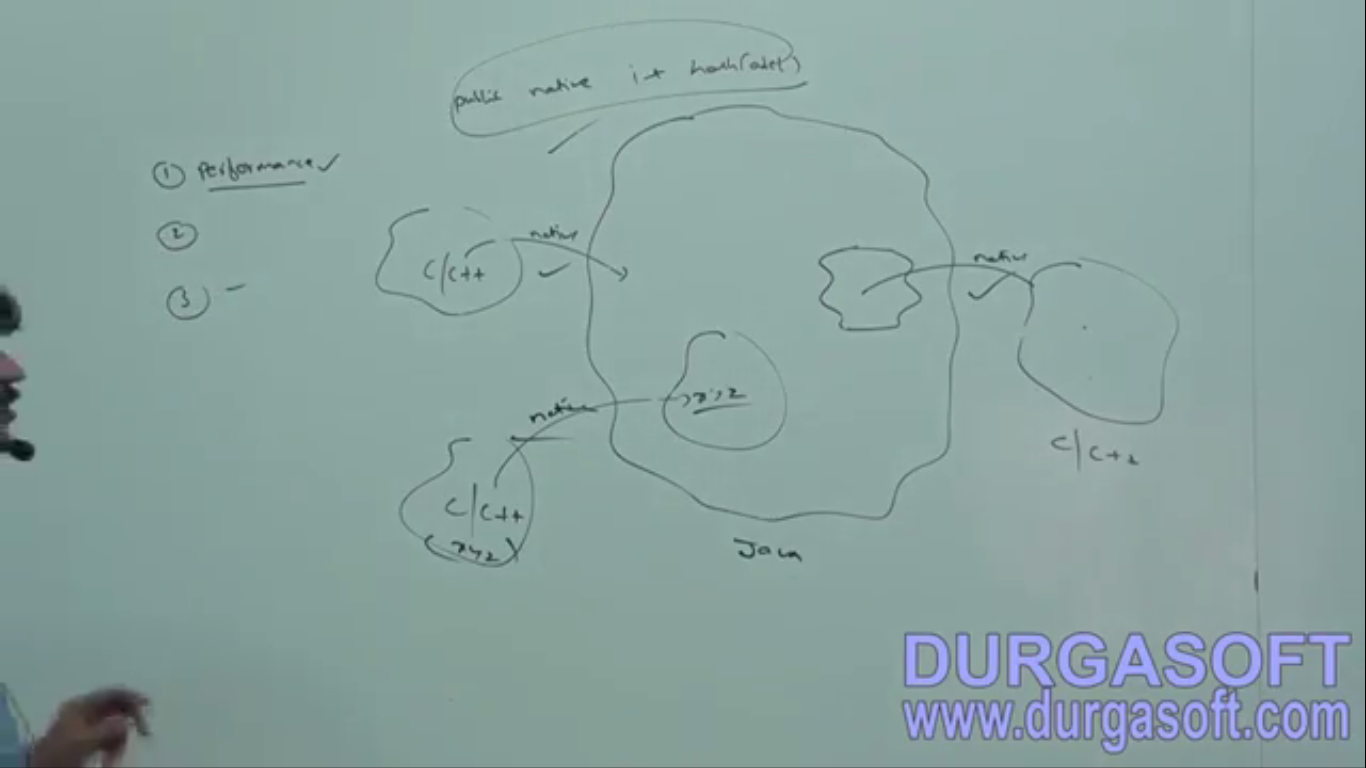
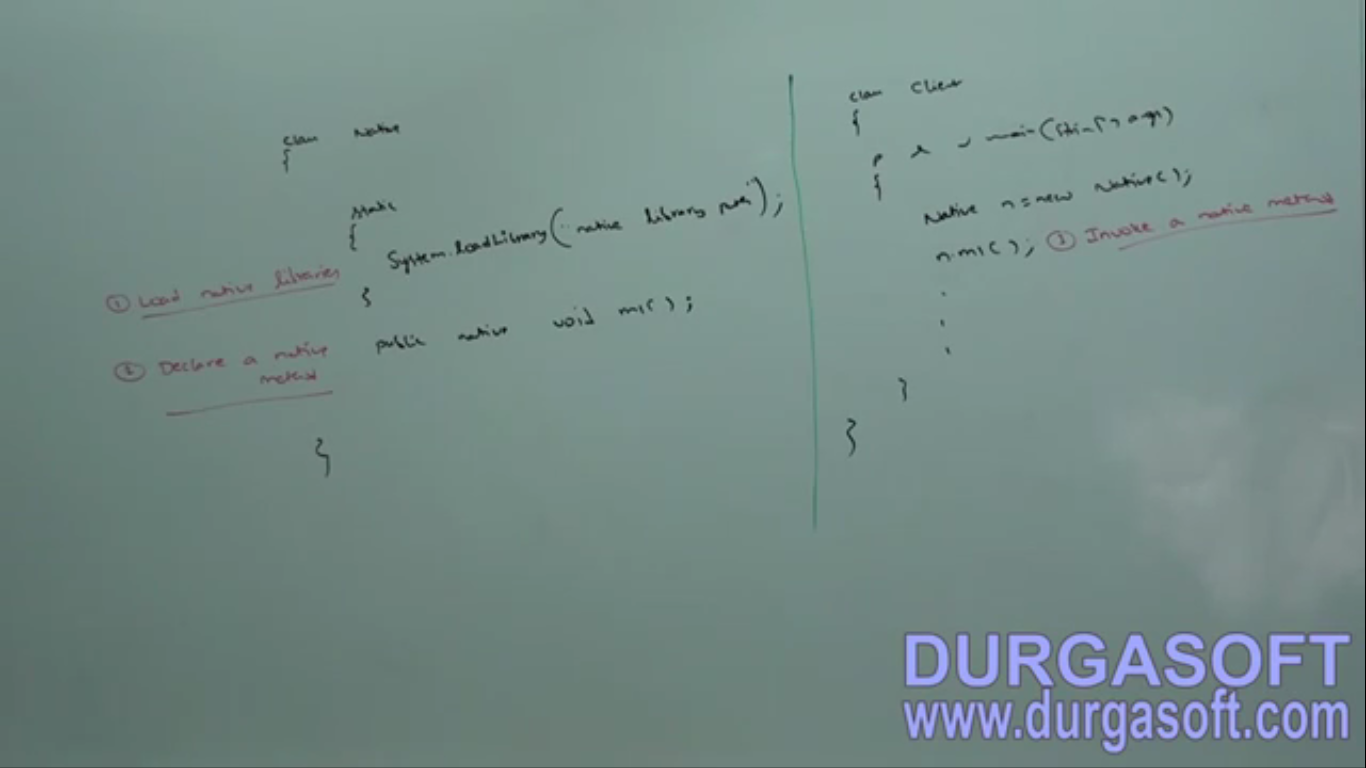
native modifier

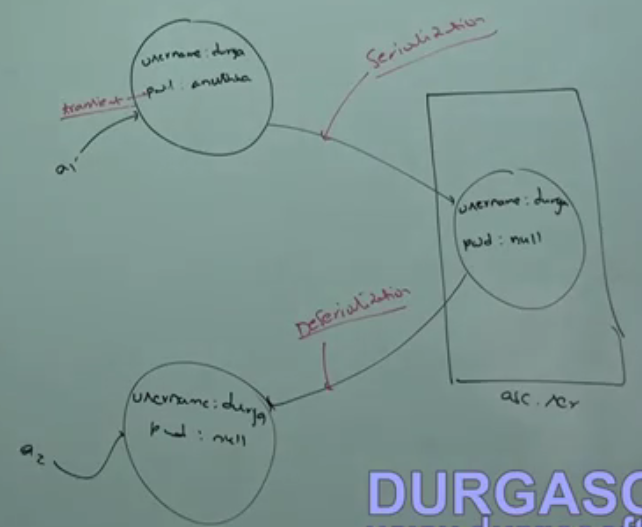
1. **Applicable** only for methods and nowhere else.
2. **native method**: A method which is already implemented in non-java (mostly C or C++).   
   also known as **foreign method**.
3. Why for native method?  
   

the main objectives of native keyword are:

* **Performance**: To improve the performance of the system.
* **Machine Interaction**: To achieve machine level or memory level communication.  
  i.e. public **native** int hashCode(): Hash code is based on object memory address. In java, no way to interact with memory.
* **Non-java legacy Code**: To Use already existing legacy non-java code.   
  

1. **What is the pseudo-code to native keyword in java**?.
   1. **Load native library**: Load at the time of class loading.
   2. **Declare a native method**: Only declaration as abstract method as native method implementation is already available in old languages like C, C++ etc and we are not responsible to provide implementation. Hence, Native Method declaration should end with semicolon.   
      If you try to give body even empty, compiler will give error: native method can’t have a body
   3. **Invoke the native method**:  
      NOTE: There is JNI (Java Native Interface) to make the above possible. 
2. **NOTE**: For native methods, implementation is already available in old languages whereas for abstract methods, implementation should not be available, hence, we can’t declare native method as abstract. That is “native abstract” combination is illegal combination for methods.
3. **NOTE**: We can’t declare native method as **strictfp** because there is no guarantee that old languages follow IEEE 754 standard. Hence, “**native strictfp**” combination illegal combination for methods.
4. **Advantage**:
   1. The main advantage of native is Performance will be improved.
5. Dis**Advantage**:
   1. The main disadvantage of native is that it breaks the platform-independent nature of java.
6. d

Transient Modifier

1. **Applicable for only variable.**
2. We can use transient keyword in **Serialization Context**.
3. **When to use transient**?
   1. At the time of serialization, if we don’t want to save the value of a particular variable to meet security constraint, then we should declare that variable as transient.
   2. At the time of serialization, JVM ignores original value of transient variable and saves the default value to the file.
   3. Hence transient means “Not to Serialize”.
   4. **Example**: password property field  
      
4. d

Volatile modifier

1. **Applicable** for only variable and can’t be applied anywhere else.
2. If the value of a variable keeps on changing by multiple threads, then there may be chance of data-inconsistency problem. We can solve this problem by using volatile modifier.
   1. If we declare a variable as volatile, then for every thread, JVM will create a separate local copy. Every modification performed by a thread will take place in local copy so that there is no effect on the remaining threads.
3. **Advantage:** 
   1. The main advantage of volatile keyword is that we can overcome “data-inconsistency” problem.
4. **Disadvantage:** 
   1. The main disadvantage of volatile keyword is that creating and maintaining a separate copy of variable for every thread increases complexity of programming and creates **performance problem**. Hence, if there is no specific requirement, it’s never recommended to use volatile keyword and it’s almost deprecated keyword.
5. **NOTE**: final variable means the value never changes whereas volatile variable means the value keeps on changing. Hence the combination “volatile final” is illegal.

Summary for 12 modifiers

1. Interface is same as class. By default interface is abstract and every method in interface is abstract. So final interface is now allowed.
2. Enum is same as class. Enum by default is final. We explicitly we can’t declare Enum as final. It’s final so can’t be abstract.   
      
   NOTE: The only applicable modifier for local variable is **final ☺**

**NOTE: native is applicable for only methods.**

**NOTE: volatile and transient modifiers are applicable only for instance variables.**

**NOTE: final is only modifier which is applicable for class but not for interface.**

**NOTE: final and abstract are only modifiers which are applicable for class but not for enum.**