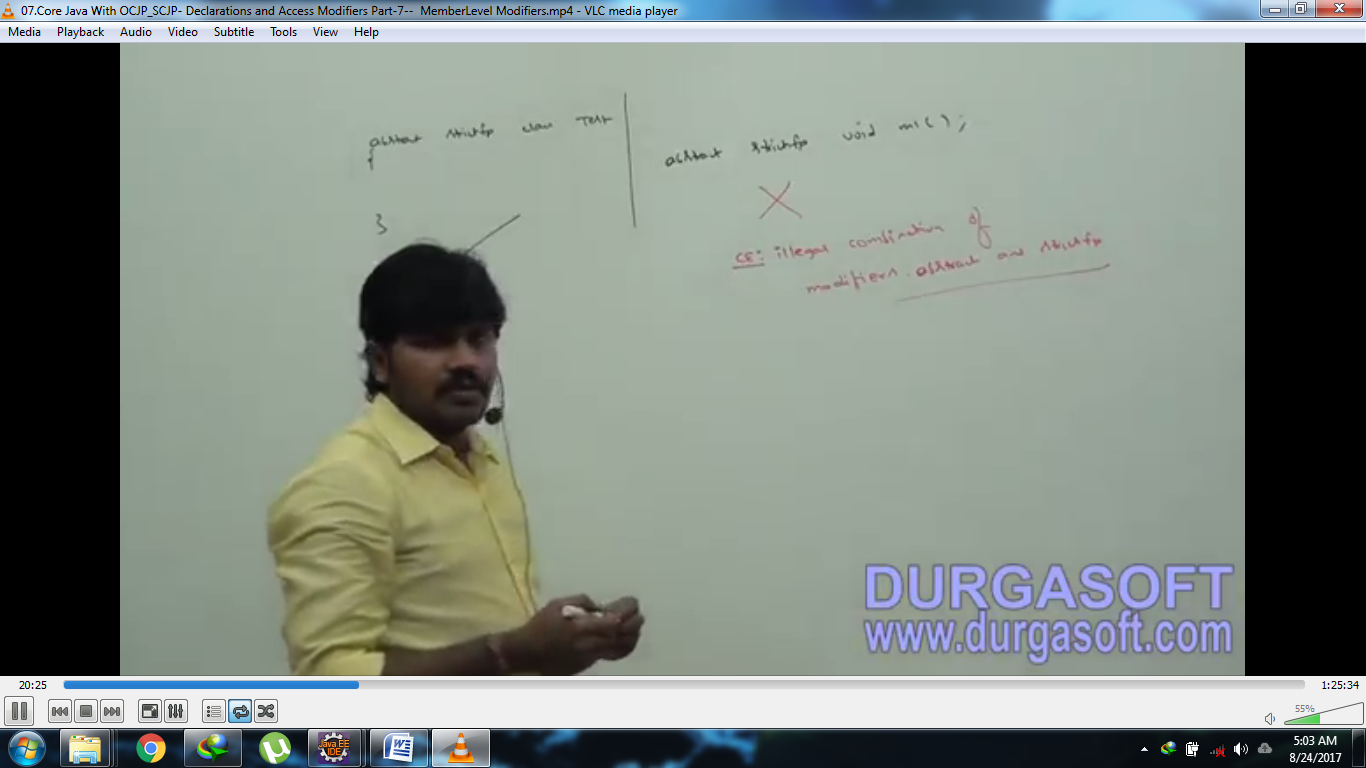
stritfp

1. **Full form**: strictfp 🡨 strict floating point 🡨 1.2 v
2. **Need**:
   1. Usually the result of floating point arithmetic varies from platform to platform. If we want platform independent result for floating point arithmetic, then we should go for **strictfp** modifier.
   2. For Example: the result of sop(10.0/3)
      1. in mathematics 🡪 3.3 bar
      2. on 16 bit machine🡪 upto 5-6 decimal place
      3. on 32 bit machine 🡪 upto 31-32 decimal place
      4. NOTE: result is varying from machine to machine but java is platform independent language.
3. **Applicable**: like abstract
   1. class level
   2. method level
4. if a method is declared to be strictfp, then all floating point calculations in that method must follow IEEE 754 standard result.
5. NOTE: abstract modifier never talks about implementation whereas strictfp always talks about implementation hence abstract strictfp combination is **illegal** combination

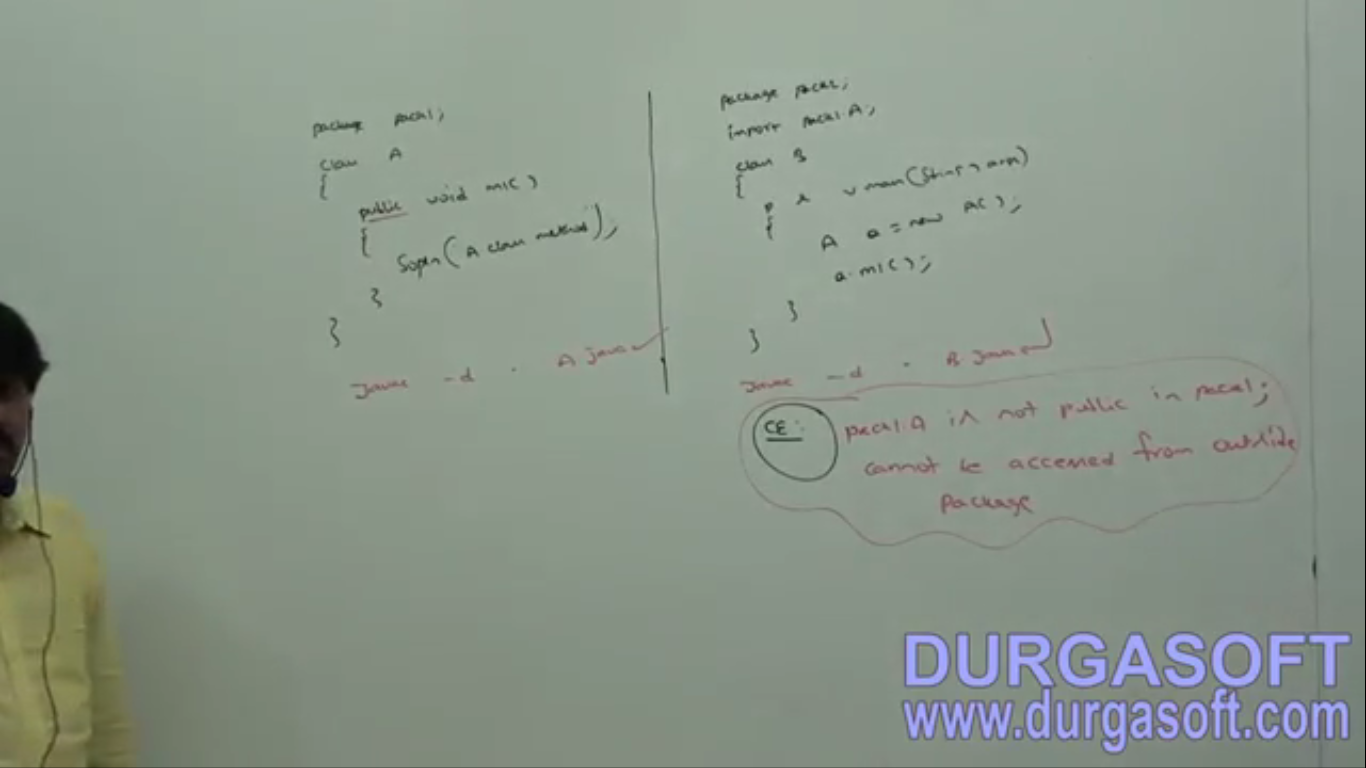
strictfp class

1. If a class is declared as strictfp, then every floating point calculation present in every concrete method has to follow IEEE 754 standard so that we will get platform independent results.
2. We can declare abstract strictfp combination for classes that is abstract strictfp combination is legal for classes but illegal for method.   
   
3. d

Member modifiers

# method or variable level modifiers

public members

1. If a member is public, then we can access that member from anywhere but the corresponding class should be visible that is before checking member visibility we have to check class visibility.  
   
2. d

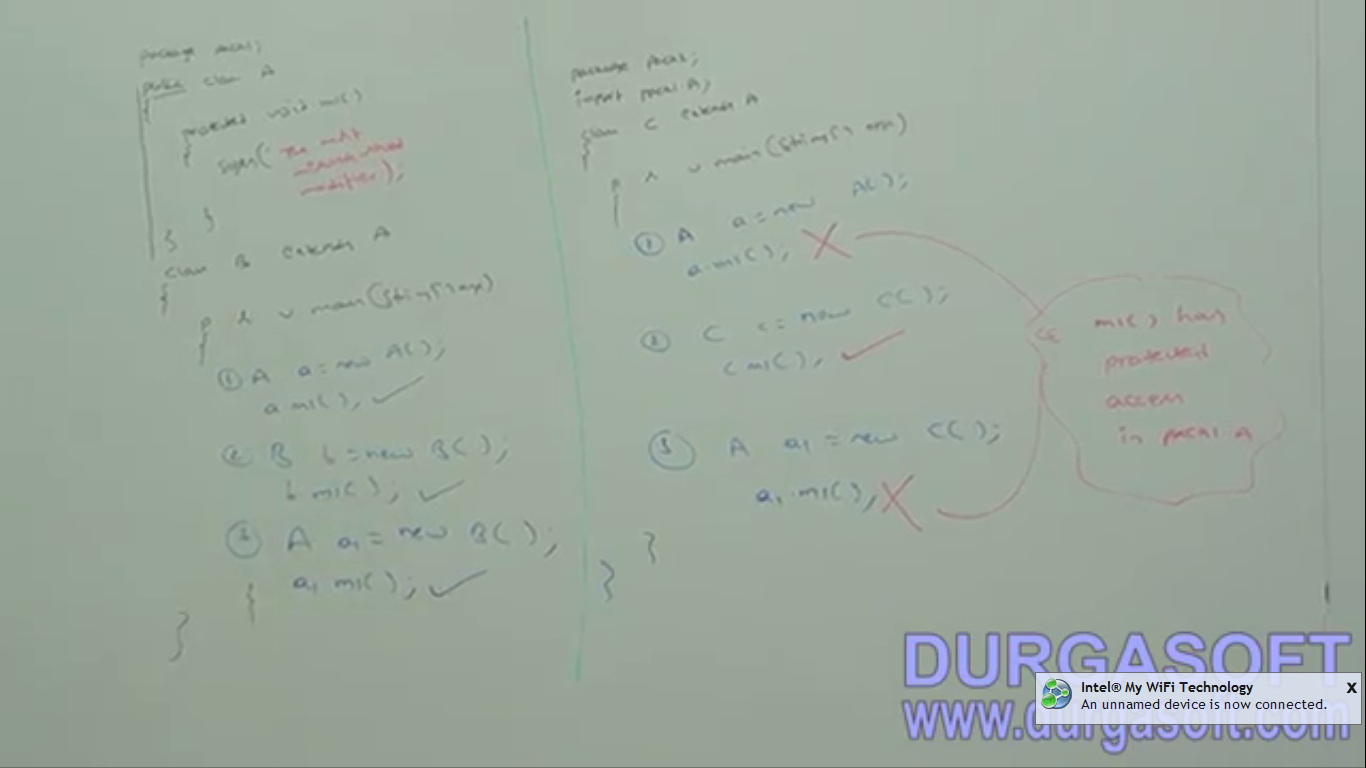
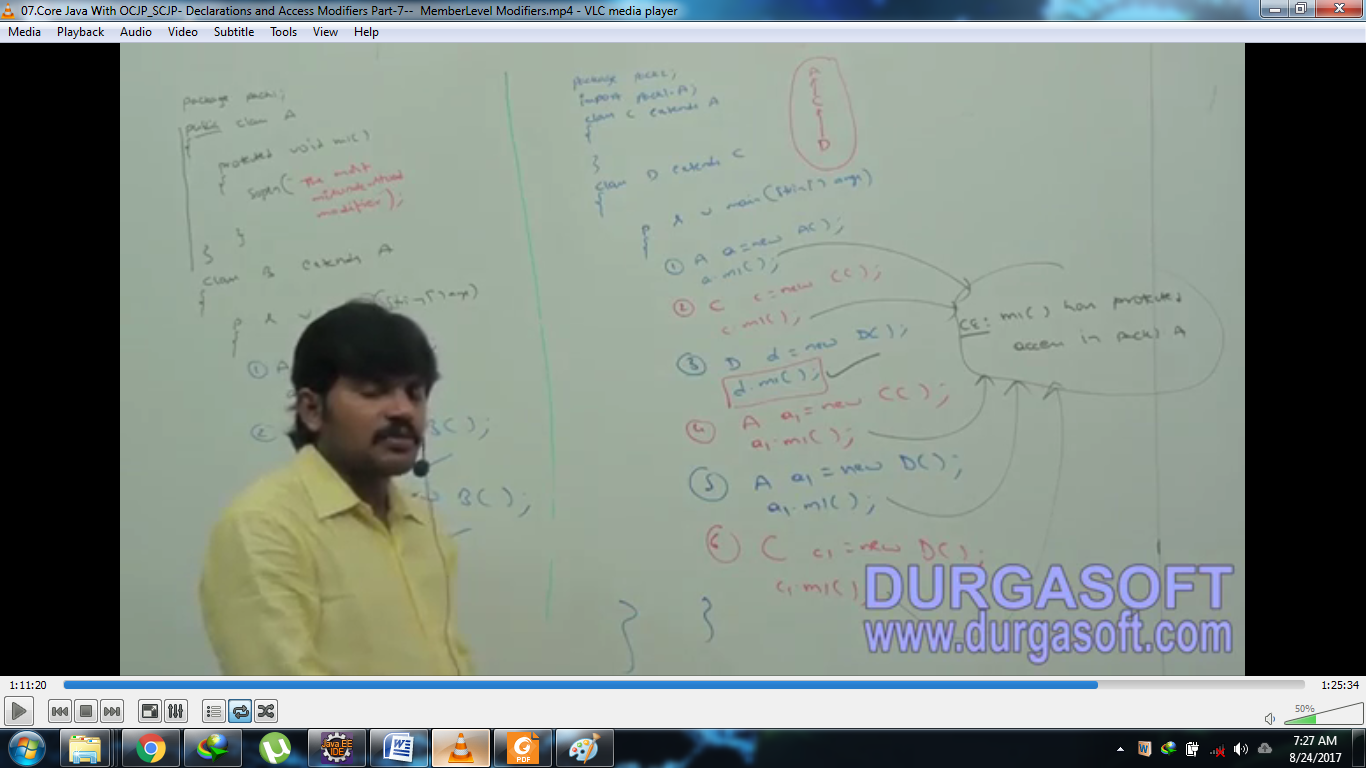
<default> members

1. If a member is declared as default, then we can access that member only within the current package. From outside the package, we can’t access. Hence, default access is also **known as** package-level access

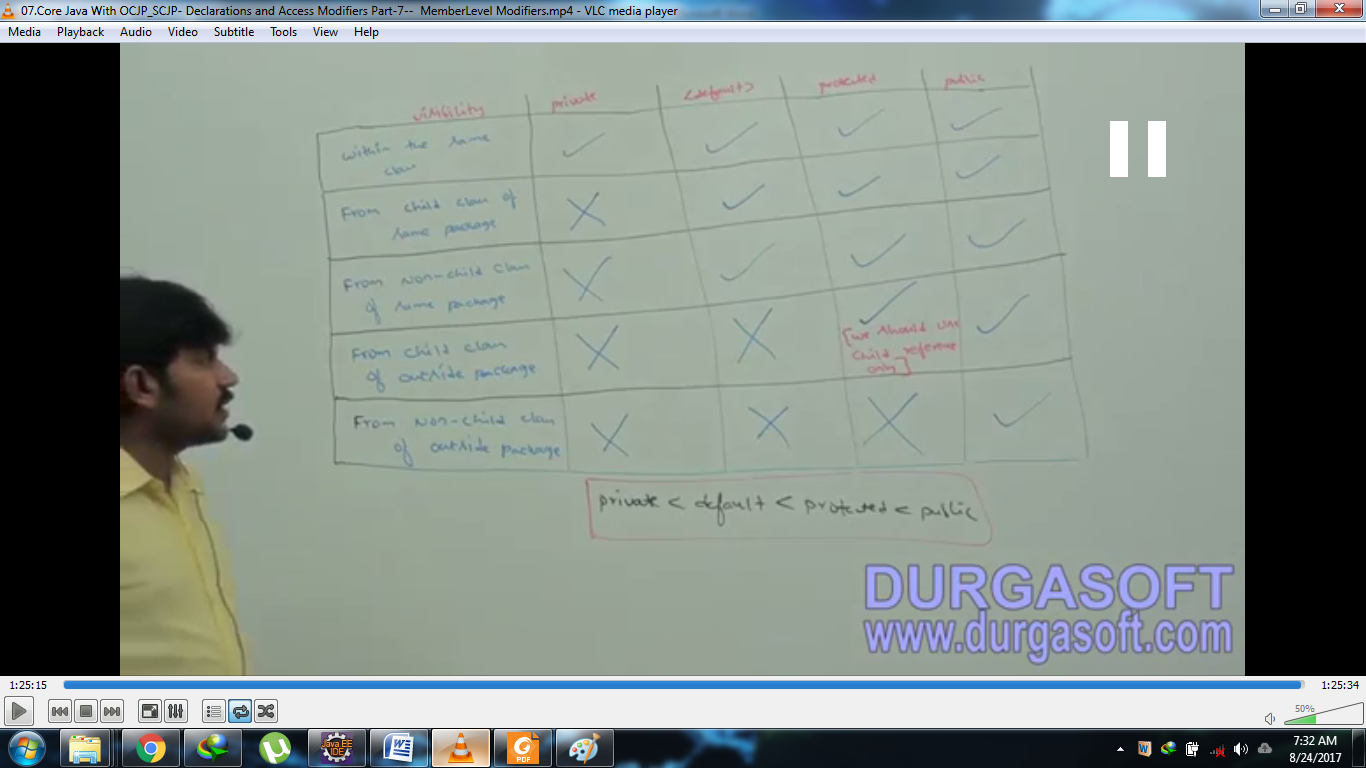
private members

1. If a member is private, we can access that member only within the class not outside the class. ☺
2. abstract method should be available to the child class to provide implementation whereas private methods are available to the child class to provide implementation. Hence **private abstract combination** is illegal for methods.

protected members

1. The most misunderstood modifiers in java.
2. If a member is declared as protected then we can access that member anywhere in the current package but only in child classes of outside packages.   
   It implies that 🡪   
   protected = <default> + kids(child)
3. We can access protected member within the current package anywhere either by using parent reference or by using child reference.   
   BUT!!!  
   We can access protected member outside package only in child classes and we should use child reference. That is parent reference can’t be used to access protected members from outside package.  
   Jatin: we can access parent class protected members inside the its child classes with the little restriction:   
   Restriction: when access the protected members inside the child class, the reference must be of child class.   
   
4. \*\*\*NOTE: we can access protected members from outside package only in child classes and we should use the exact same child class reference only.   
   Example: D extends B extends A (having protected member m1())   
   From D class when accessing A class m1(), we should use D class reference only otherwise meet compile time error.  
   Example:   
   
5. d

Summary Table for Modifiers



private is more restricted than default which is more restricted than protected which is more restricted than public.