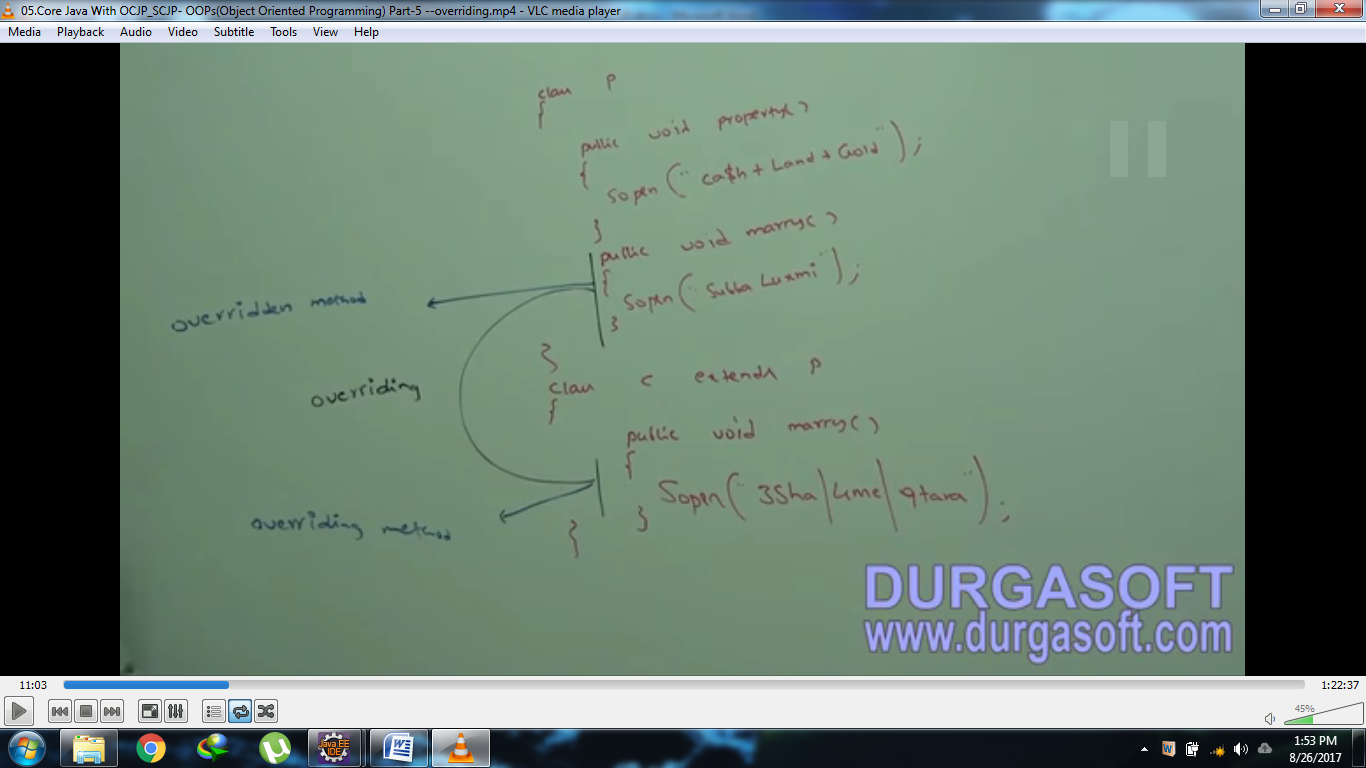
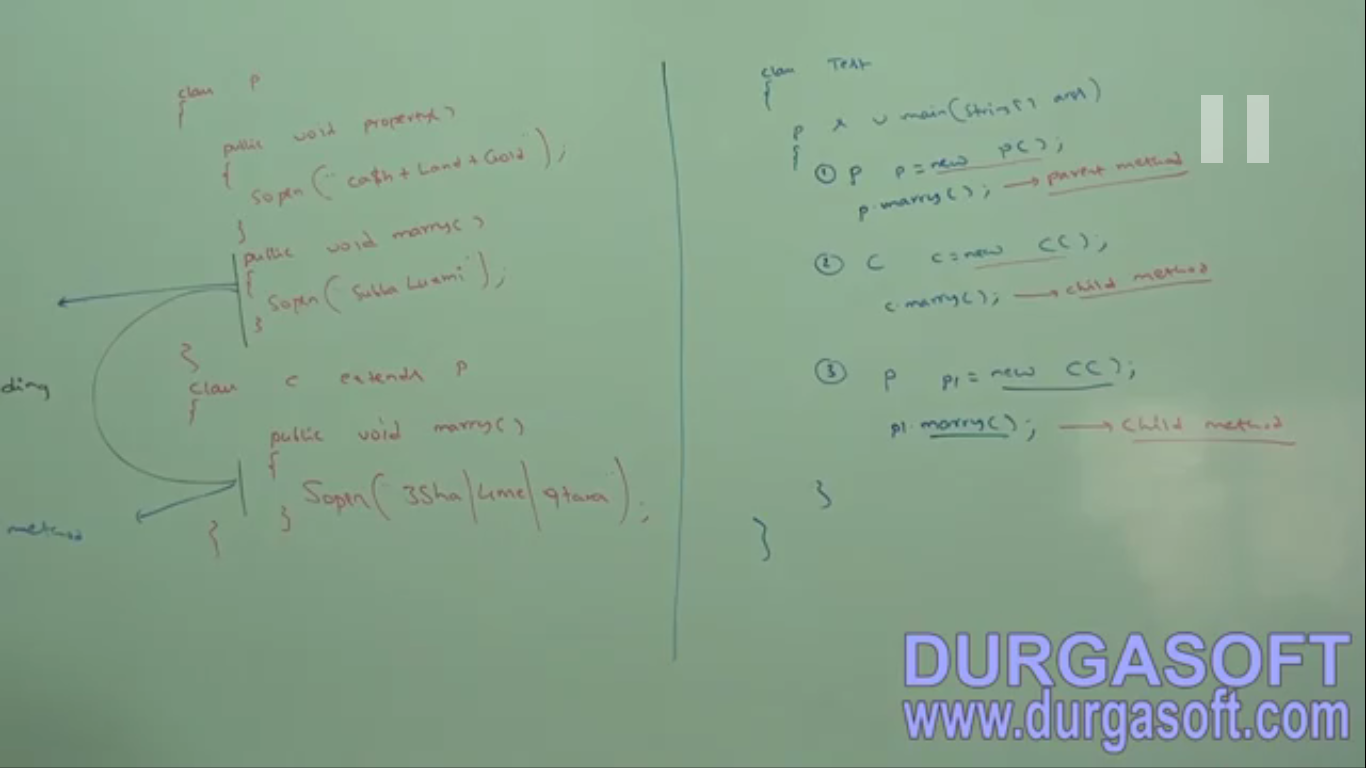
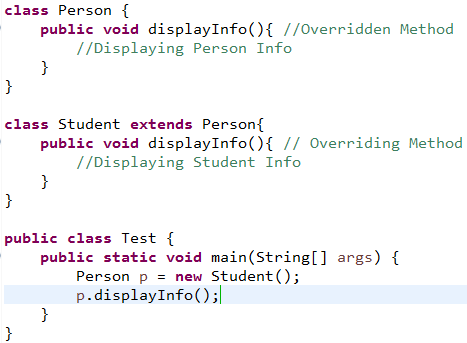
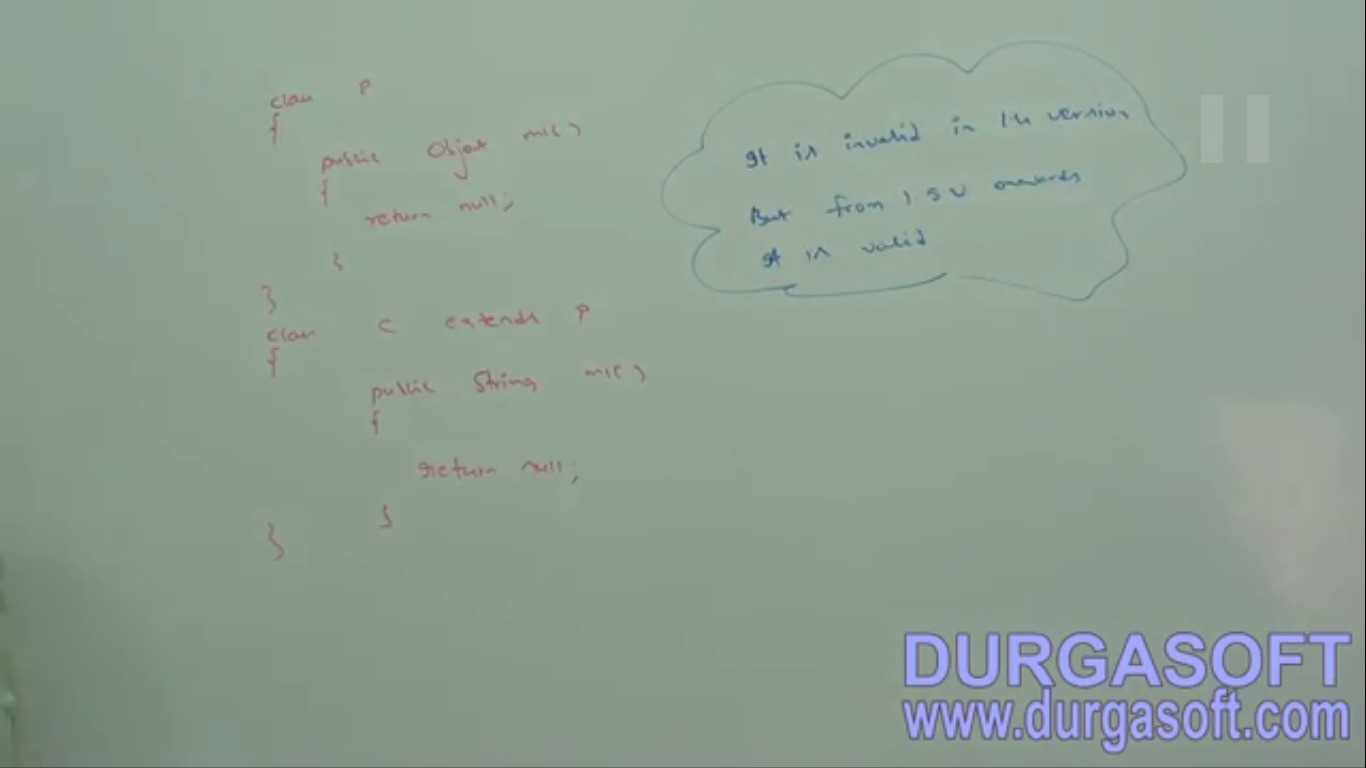
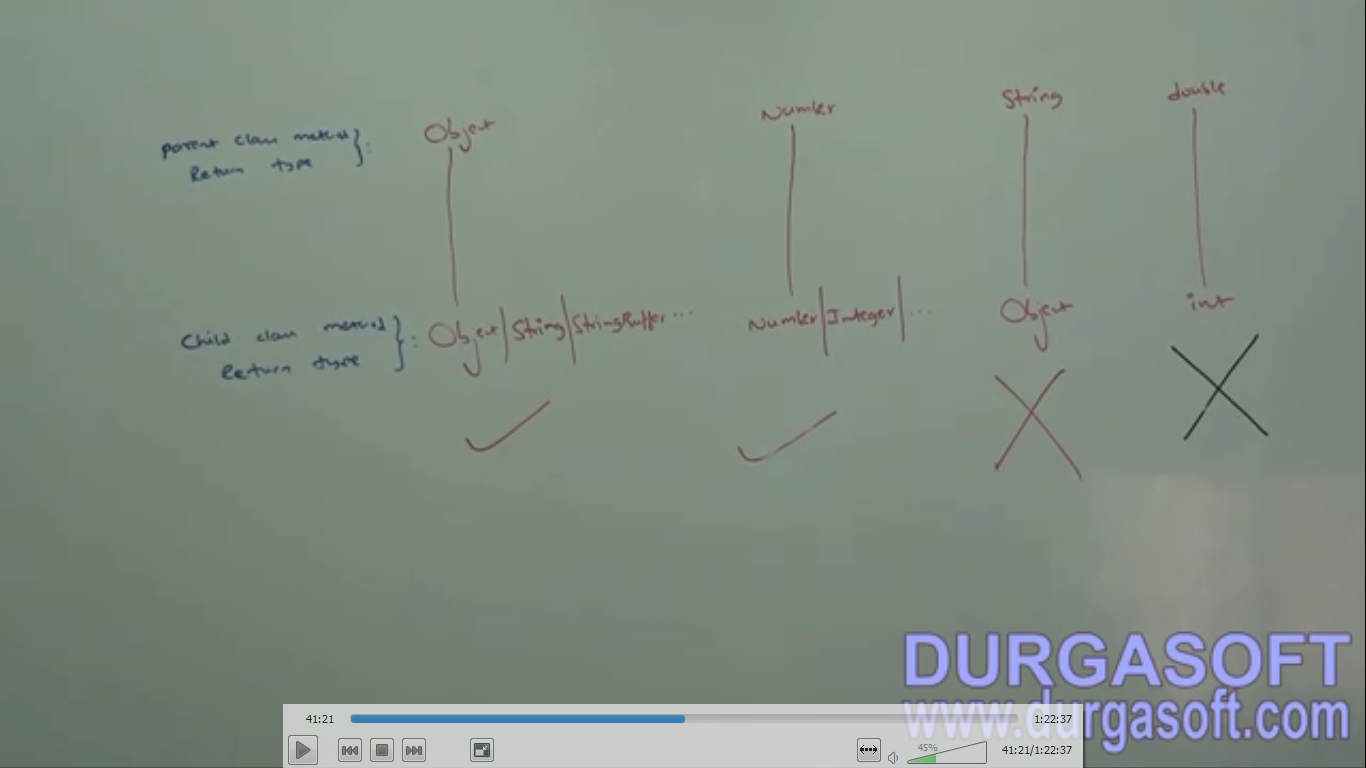
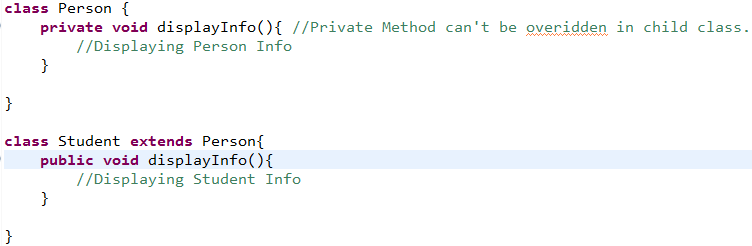
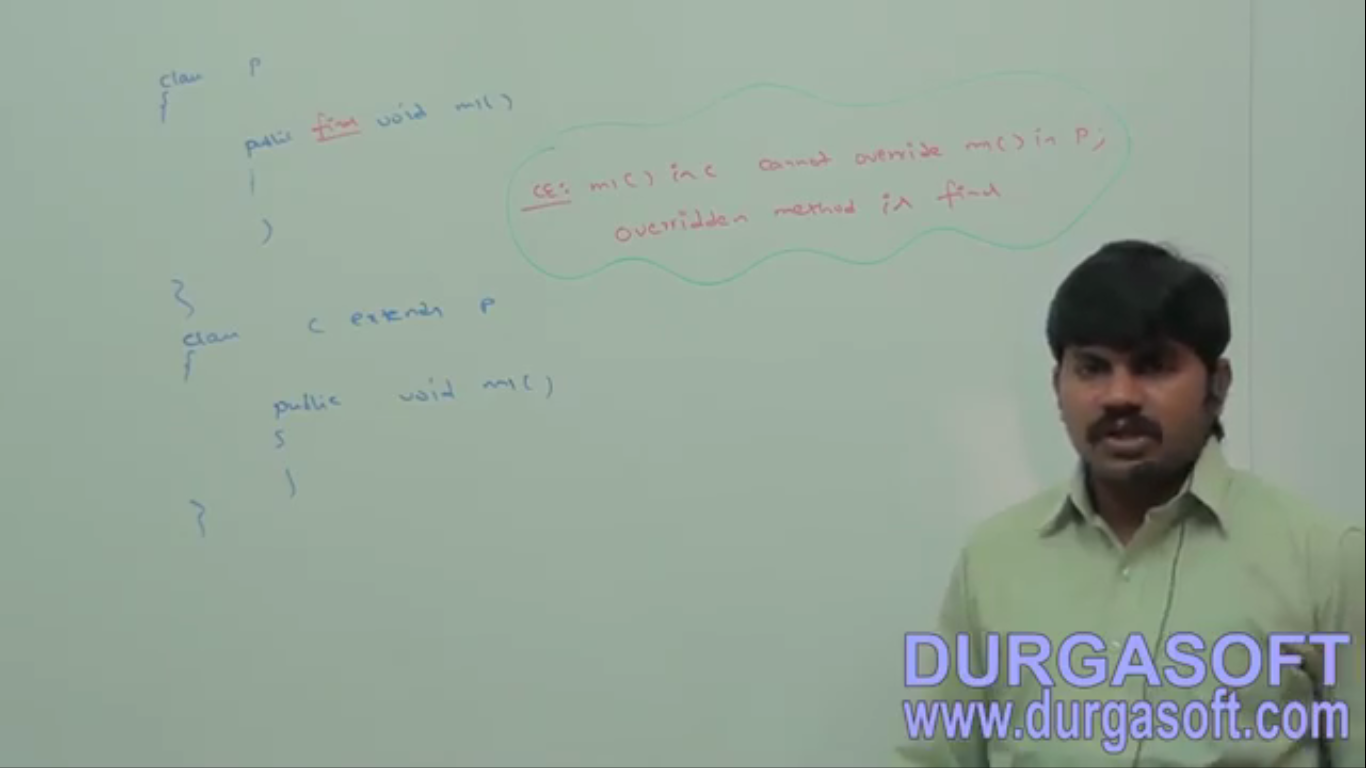
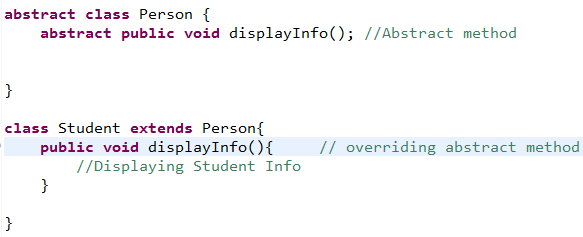
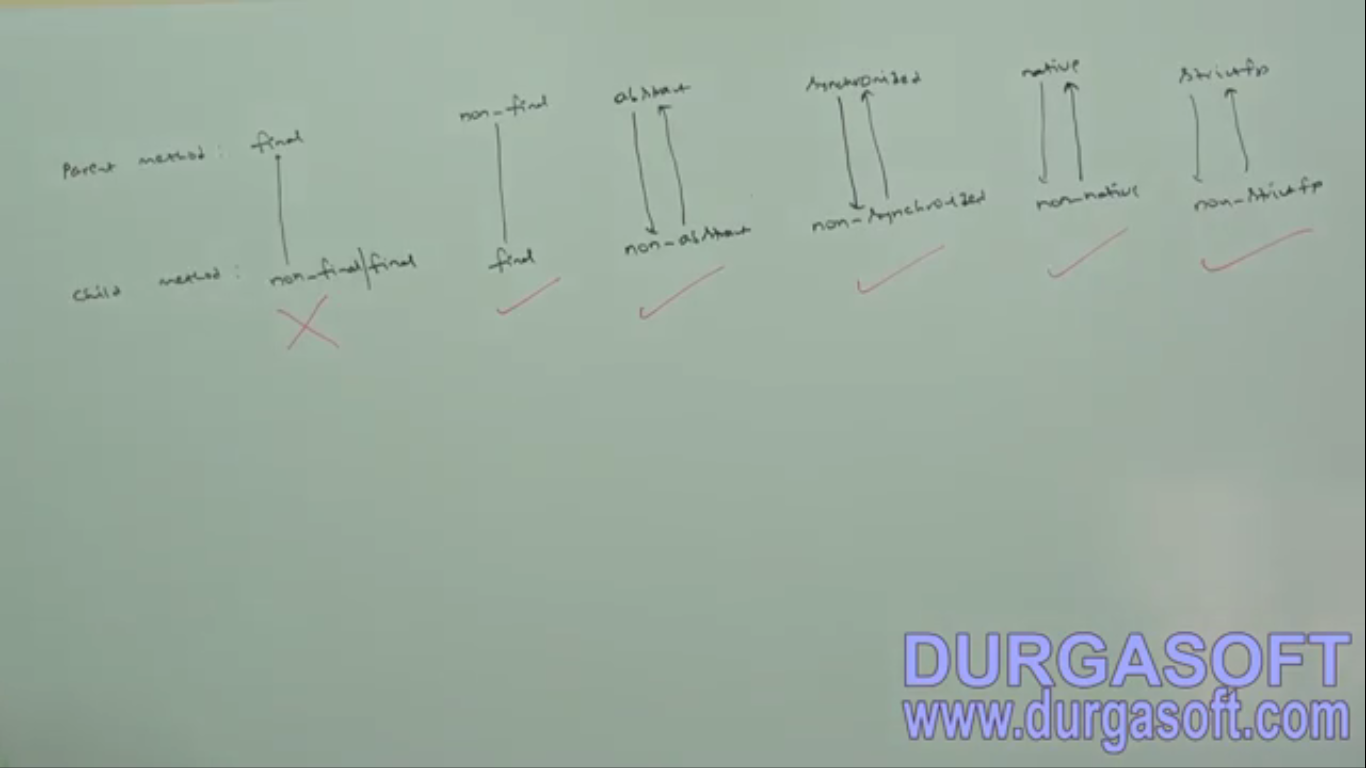
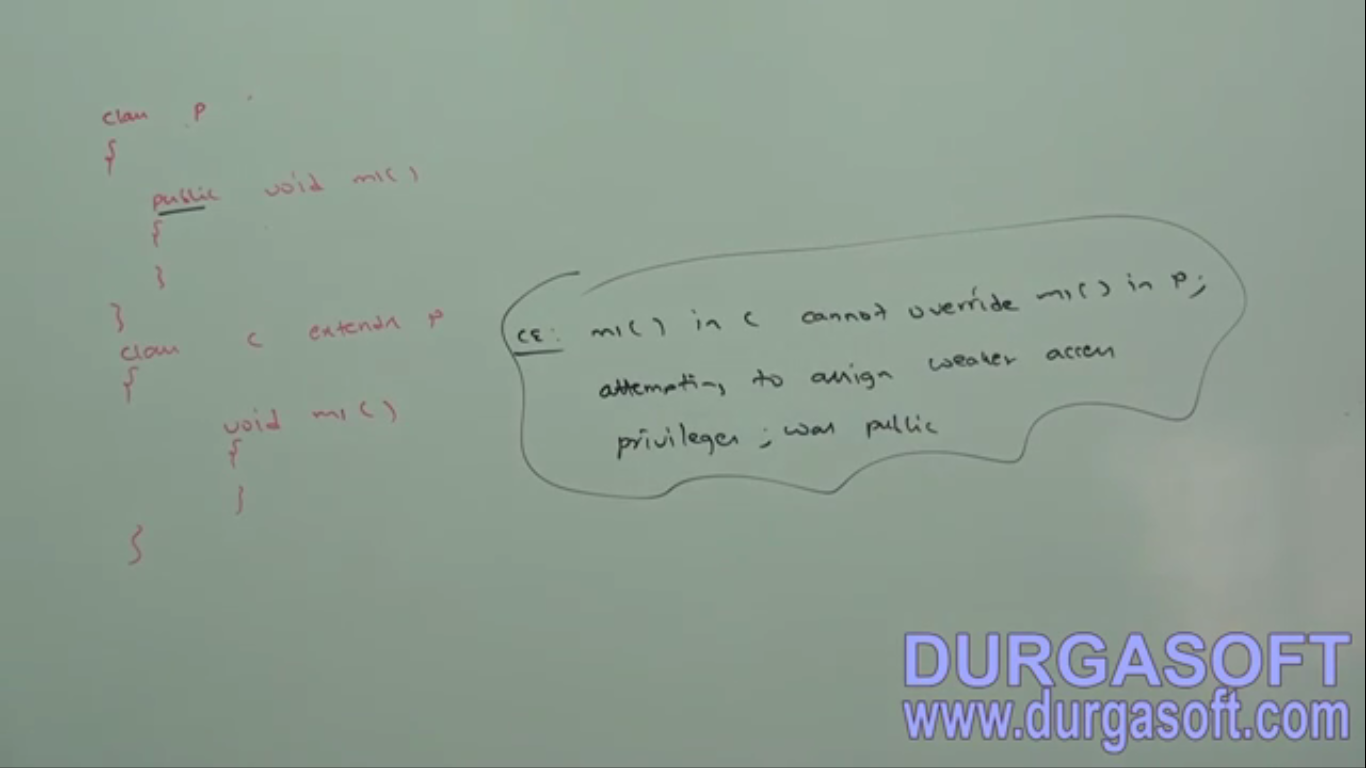
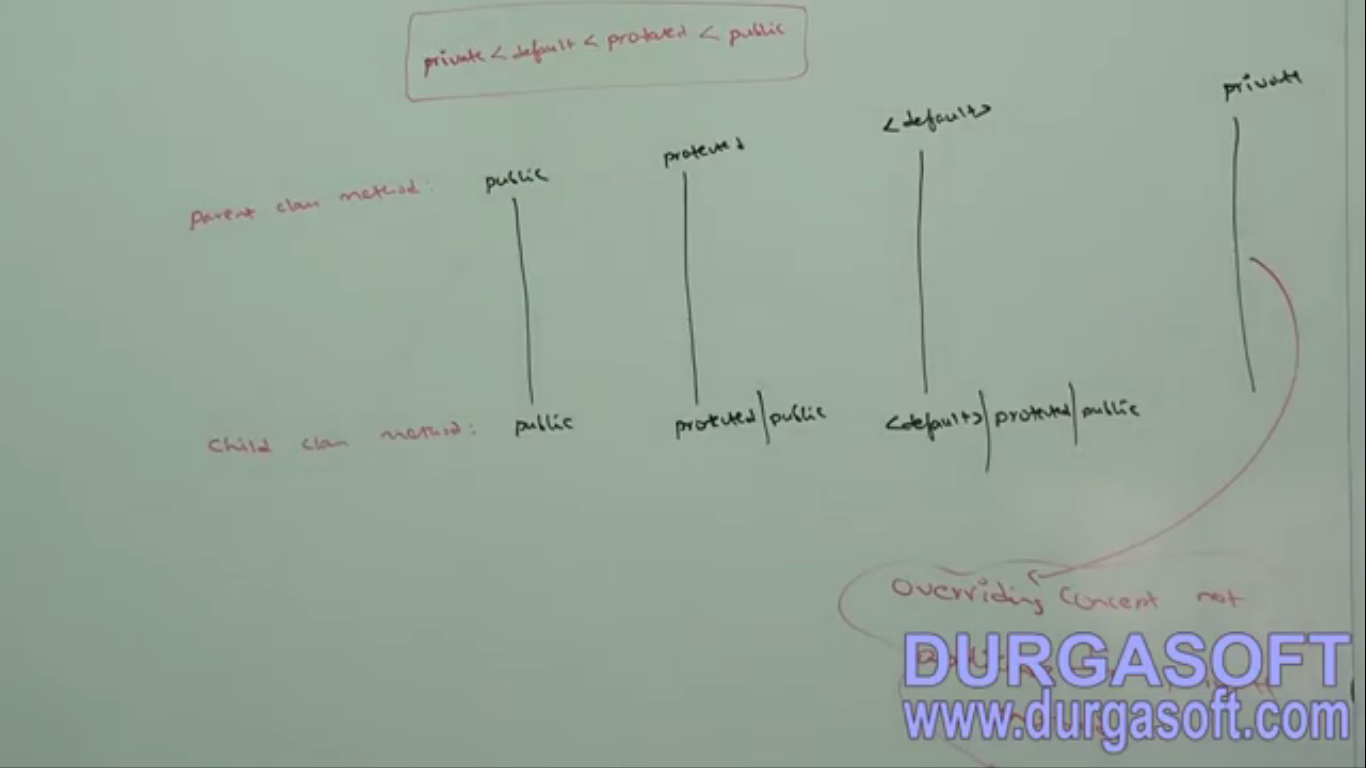
overriding

1. Whatever methods parent has, by default are available to the child through inheritance. If child class not satisfied with parent class implementation, then child is allowed to redefine that method based on its requirement. This process is called overriding .   
   The parent class method which is overridden is called Overridden Method and the child class method which is overriding is called Overriding Method.   
   
2. In overriding, method resolution always takes care by JVM based on runtime object and hence overriding is also considered as runtime polymorphism, dynamic polymorphism or late binding.   
   
3. Jatin🡪 Let’s try to understand in detail how compile time and runtime process are proceeded in overriding.   
   Here we have two classes🡪 Person, Student (Subclass)  
   Student class is overriding displayInfo() of parent class Person.  
   See the following Example:   
     
   Now pay the attention to p.displayInfo().   
   At compile time, compiler examine p.displayInfo() and finds out that p is of type Person and so Person class has the method displayInfo(). Yes compiler finds this method in Person. So compiler passes this statement. Did you notice, this is the reason why we can call only parent class methods via parent class type reference variable. Now compiler’s duty completes.   
   Now at runtime, JVM comes.   
   It also examine the statement p.displayInfo(). JVM finds out that Person class type reference variable contains the runtime object which is of type Student. So JVM checks that does Student contain displayInfo(). Yes Student Class has this method. If Person does contain its own method displayInfo(), then JVM calls that method otherwise definitely Parent class contains this method displayInfo() as passed by compiler, JVM will call that method.
4. d

Rules for overriding

1. In overriding, method names and argument types must be matched. That is, method signature must be same.
2. The return type must be same as the parent class overridden method’s return type till 1.4. From 1.5v, the co-variant return type is accepted in overriding method.   
   **\*\*\*NOTE**:  
   **Co-Variant**: This concept is applicable only for reference type not for primitive type.   
   
3. Parent class private methods are not available to the child. Hence, the overriding concept is not applicable for private method.   
   
4. Final declared method in parent class can’t be overridden in child class.   
   
5. Parent class **abstract method,** we should override in child class to provide implementation.   
   
6. We can override non-abstract method with abstract method in child class. In which case, we should do this? When child class doesn’t want this method and doesn’t know how to override this method. or when parent class doesn’t want any of inherited method to pass down to its child class.
7. In overriding, the modifiers don’t keep any restriction.   
   synchronized, native, strictfp, abstract. We will talk about static separately next.   
   
8. While overriding, we can’t reduce the scope of access modifier but we can increase the scope (visibility). 



Scope from weaker to stronger  
private < default < protected < public(Strongest Scope)  


Overriding concept is not applicable for private method in parent class.