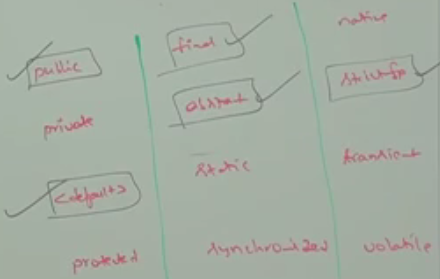
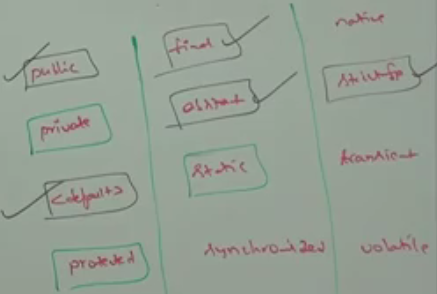
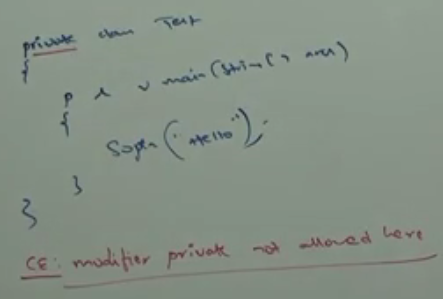
Class Level Modifiers

1. Whenever we are writing our own classes, we have to provide some info about our classes to the JVM such as .
   1. Where this class is accessible.
   2. Whether its child class creation is possible or not.
   3. Whether its instance can be created or not.
   4. etc.  
      we can specify this info by using appropriate **modifiers**.
2. Total modifiers = 12
   1. public
   2. private
   3. <default>
   4. protected
   5. final
   6. abstract
   7. static
   8. synchronized
   9. native
   10. strictfp
   11. transient
   12. volatile.
3. Which modifiers are allowed for top-level modifiers?
   1. 
4. Which modifiers are allowed for inner-class?  
   
5. 
6. What is difference b/w Access Specifiers and Access modifiers?  
   1. Specifiers:
      1. public
      2. private
      3. protected
      4. <default>
   2. Modifiers:
      1. static
      2. final
      3. abstract
      4. volatile
      5. synchronized
      6. strictfp
      7. transient
      8. native
   3. **\*NOTE**: But this rule is applicable only for old languages such as C++ but not for Java.
   4. **NOTE**: In java all are considered as modifiers only. There is no word like Specifiers.
7. d

# Let’s discuss top-level access modifiers in detail.

public class

1. If a class is declared as public, then we can access that class from anywhere.

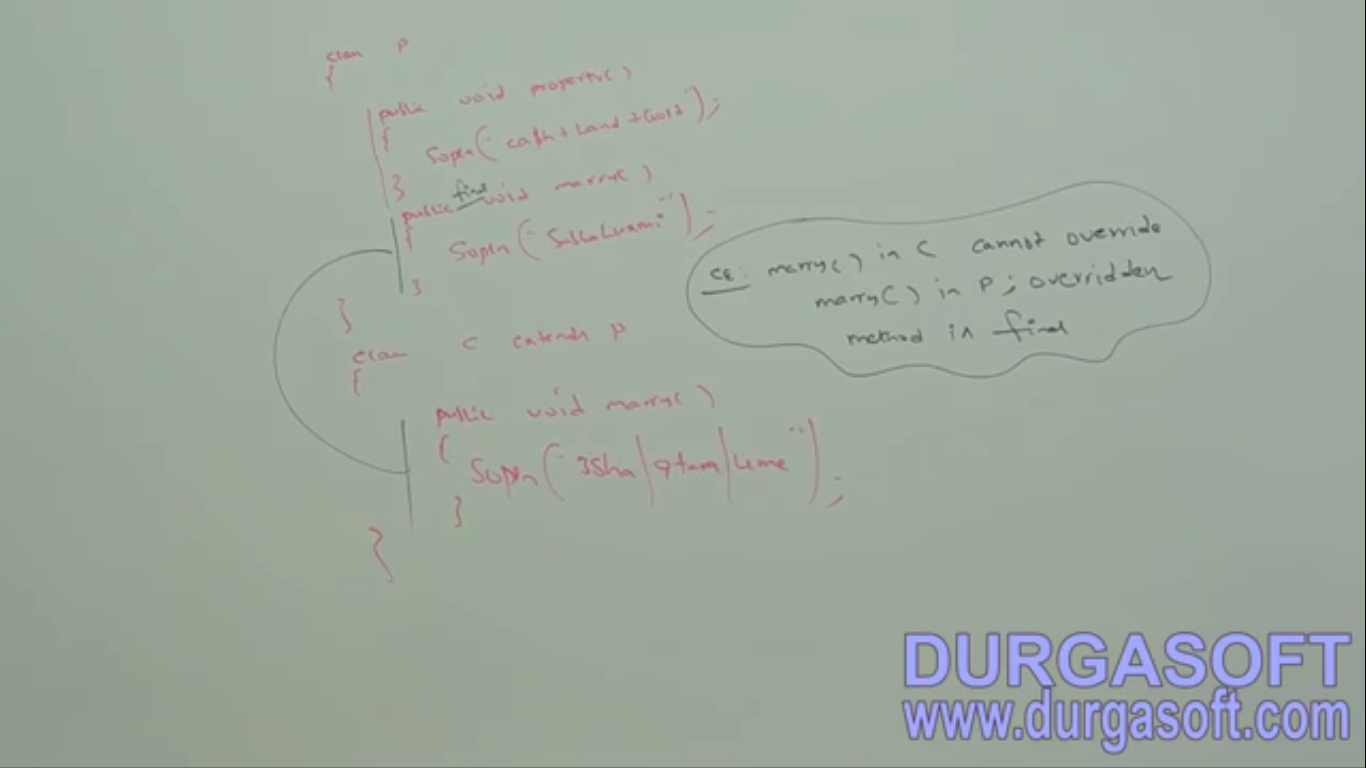
default class

1. If a class is declared as default then we can access that class only within the current package. That’s from outside package, we can’t access. Hence, Default access also known as package level access.

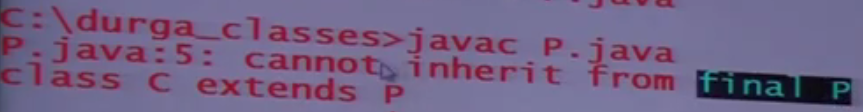
final modifier

1. final is the modifier applicable for classes, methods, and variables.

final method

1. Whatever methods parent has, by default are available to a child via inheritance. If the child is not satisfied with parent method implementation, then child is allowed to redefine that method based on its requirement. This process is called overriding.
2. If the parent class method is declared as final then we can’t override that method in the child class. Because its implementation is final.  
   
3. d

final class

1. If a class is declared as final, we can’t extend the functionality of that class because we can’t create child class of that class. That’s inheritance is not possible for final classes.   
   
2. Every method in final class is final but instance variables are not final.
3. **Advantages**:
   1. Security
   2. Unique implementation
4. **Disadvantages**:
   1. Can’t inherit the final class.
   2. No polymorphism.
5. NOTE: If there is no specific requirement then it’s not recommended to use final keyword.