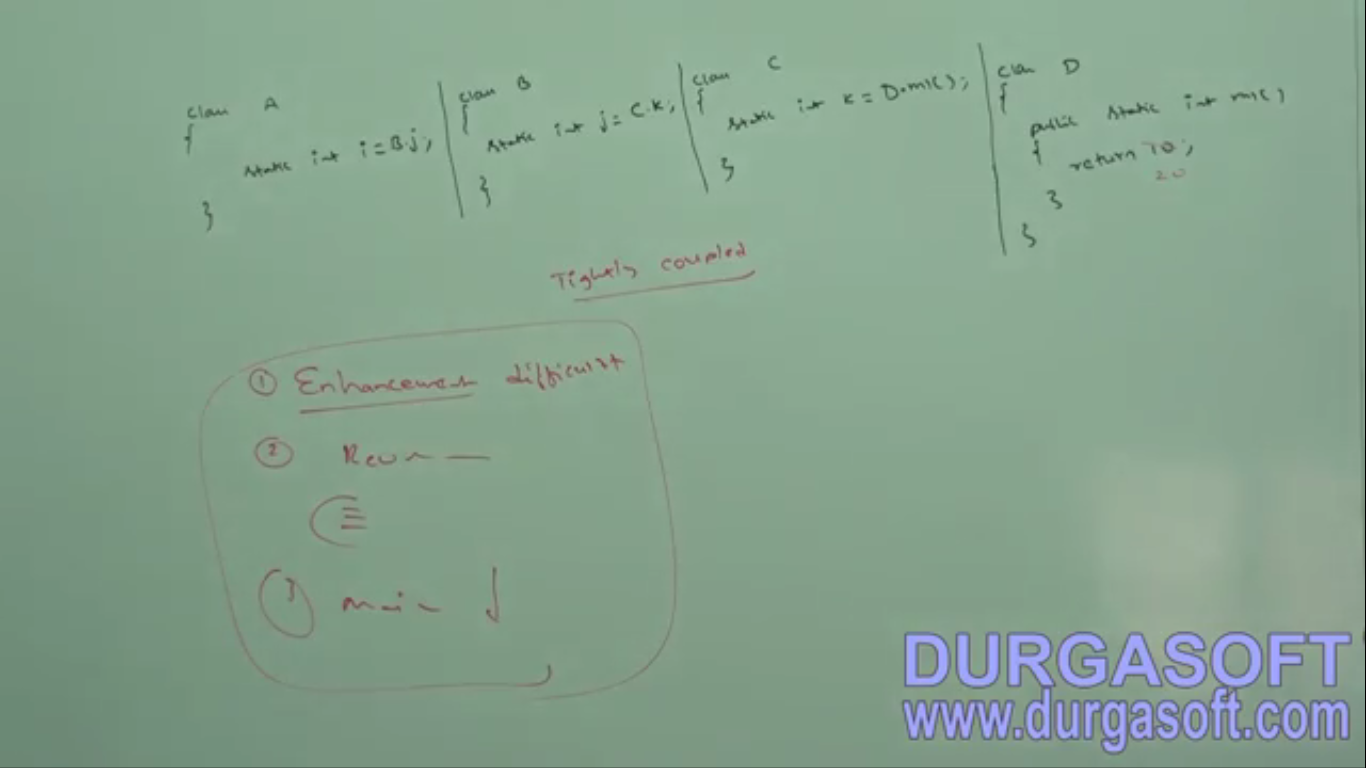
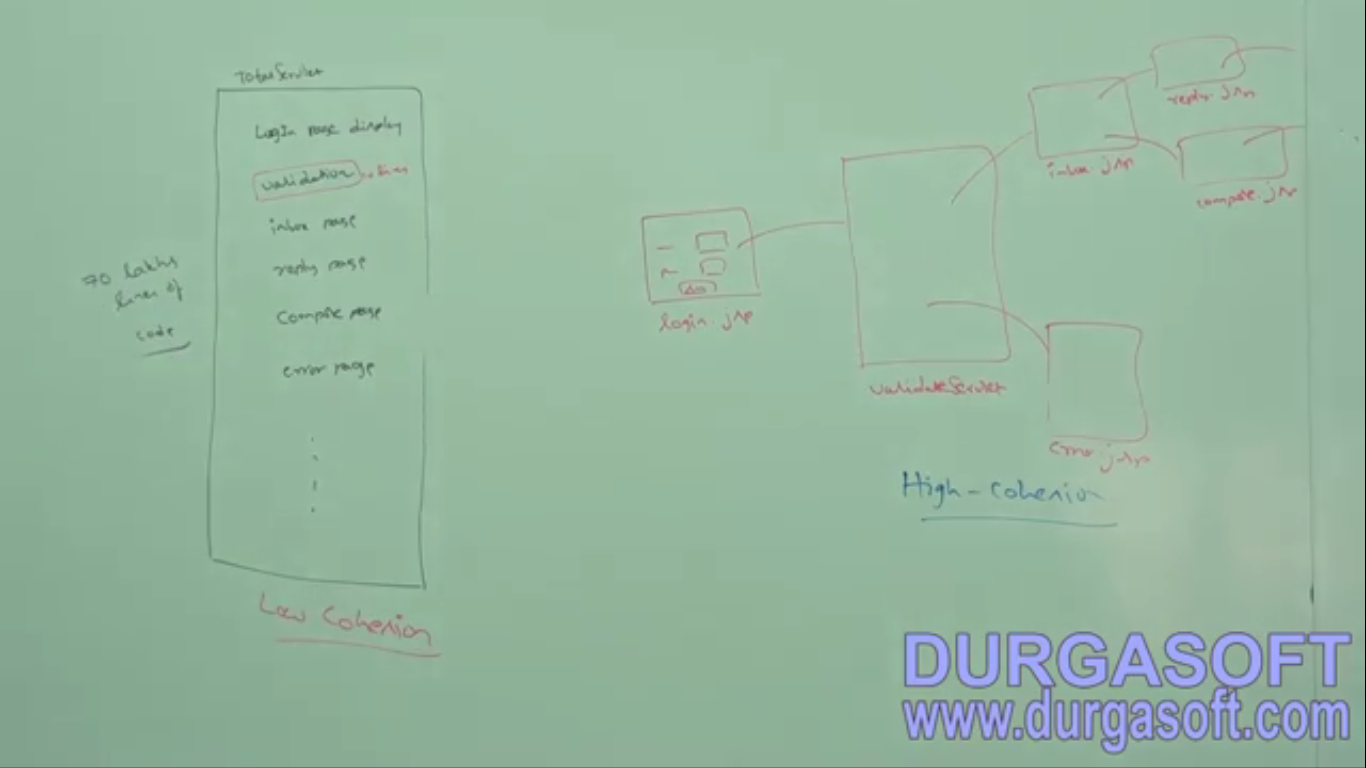
Coupling

1. **Definition**: The degree of dependency b/w the components is called **coupling**.   
   If dependency is more, then it’s considered as **Lightly Coupling**.   
   If dependency is less, then it’s considered as **Loosely Coupling**.

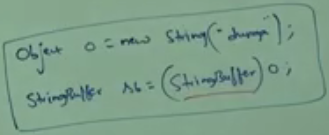
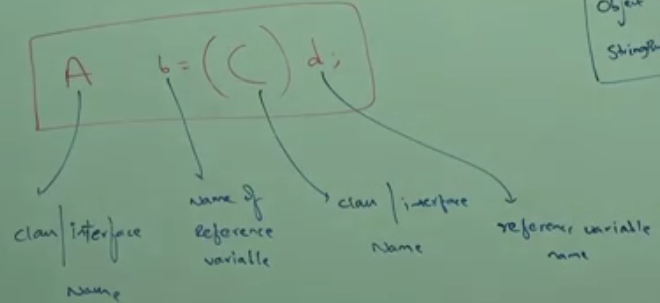
**Example**:  
NOTE: Tightly Coupling is not a good programming practice as because of serious disadvantages.  
Cohesive

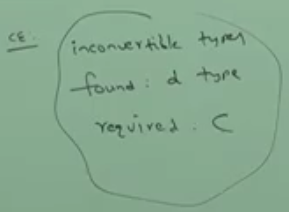
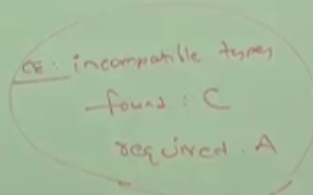
1. Enhancement becomes difficult, as changing one place affects other places too.
2. To reuse A component, B is required which requires C which requires D🡪 So much overhead
3. Enhancement is so difficult that to maintain it is so difficult.
4. For every component, a clear **well-defined functionality** is defined then that component is said to be following **high cohesion**.
5. **Example:**

**Problems**  
1. Enhancement   
2. Reusability  
3. Maintainability

1. d

Object Type Casting

1. **There are two points we should be aware of** 
   1. Parent reference can hold child reference 🡪 Object o = “String”;
   2. Interface reference can hold implemented class object🡪 Runnable r = new Thread();
2. **Let’s try to understand type casting**  
   Is the following valid typecasting **Let’s first understand the following**

* **How many activities are compiler doing in the above?**
  + **2**
    - **(C)d: Compiler will check if there is any parent-child relation b/w C and d (or same); otherwise we will get Compile Time Error.**
    - **A must be either same or Parent type of C otherwise Compile Time Error. You know while assigning reference, the collector type must be either same or parent.  
        
        
      In Short, Compiler check validity using reference type. ☺**
* **When typecasting, there are 3 activities performed by Compiler and JVM.**
* **How many activity does JVM perform in typecasting?**
  + JVM checks that is the runtime type of underlying object is either same or derived type of C otherwise we will get runtime exception saying 🡪 **ClassCastException**
  + **Jatin Remember**: From point number one from compiler rule above, we know that there is parent-child relationship b/w (C and actual object in d) or same. if same (the required operation most of the time) then no problem. if not then **there are two cases**
    - **Case01**: C is parent and actual object is child: Then there is no problem.
    - **Case02**: C is child and actual object is parent. Then there is problem. Actual object type can’t be cast to child.

1. **Exercise**:  
   