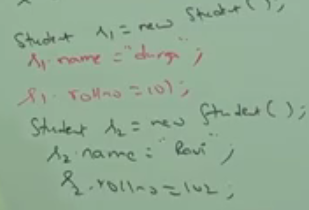
Constructor

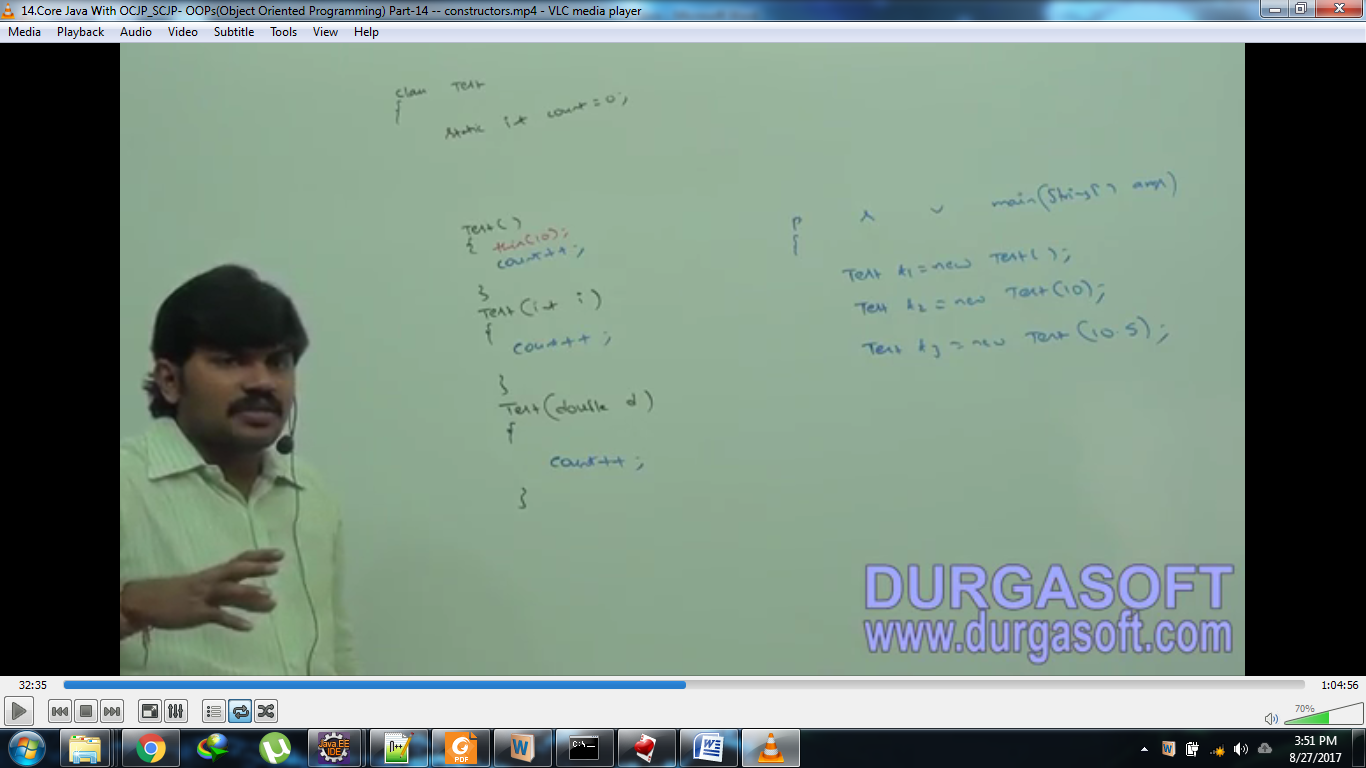
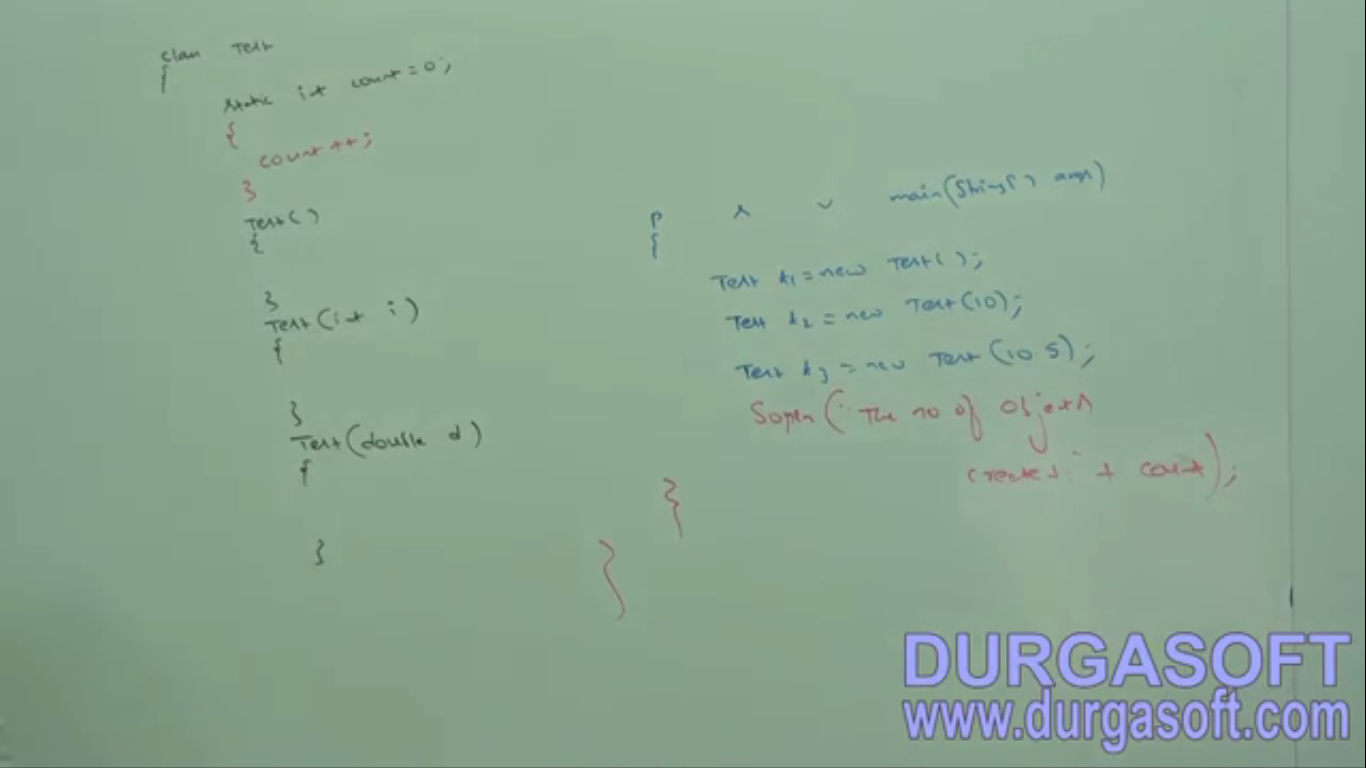
1. **Constructor**:
   1. Once we create an object, compulsory we should perform initialization only then the object is in position to respond properly.
2. **Places to perform initialization?**
   1. At the time of Declaration: But for all the objects, the state would be same.
   2. Instance Block**:** Again all objects would have same state as there is no way to pass arguments.
   3. assignment:   
        
      what if number of properties = 100, length of the code would increase. Second, properties are not directly available outside.
   4. **Constructor**: Whenever we are creating an object, some piece of the code will execute automatically to perform initialization of an object. This piece of the code is nothing but constructor. Hence, the main purpose of constructor is to initialize object initialization.
3. d

# Difference b/w Constructor and Instance block

|  |  |
| --- | --- |
| **Constructor** | **Instance block** |
| To initialize object | Other than initialization, if we want to perform some other activities such as  When object is created, do some update in DB,  Increment count how many objects are created etc. |
| Executed after instance block if any | Earlier than constructor. |
|  |  |

**NOTE**: Constructor and instance blocks have their own different purposes. Replace one concept with other, will not work always.

## Where instance block is better suited than constructor

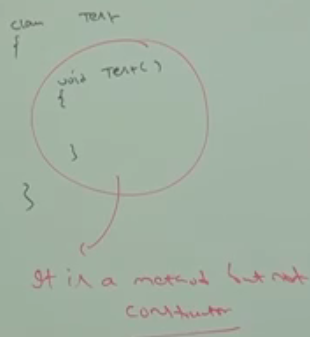
1. Keeping track of how many objects have been created.  
     
   **Using Instance block:**   
   

3 constructors. Put count++ in each constructor 🡪 Redundancy.   
Problem gets worst if we call one constructor from other constructor using this(args) now two times count++ will be executed.

# Rules: For constructors

1. Name of the class and that of constructor must match.
2. Return type concept is not applicable to constructor even void because we are not calling constructor it is called automatically.
3. The first line in every constructor must be either this([…]) or super([…]). If not given, then compiler will add super()

**NOTE**: if the return type of the constructor is specified, in that case, compiler considers that to be “non-constructor”.

* It’s possible to declare a method whose name is same as constructor.  
  
* **Only applicable modifiers for constructors**public, private, protected, <default>
* **Who is responsible to generate default constructor?**
  + **Compiler**
  + Compiler will generate constructor only in case, there is no constructor provider by programmer.