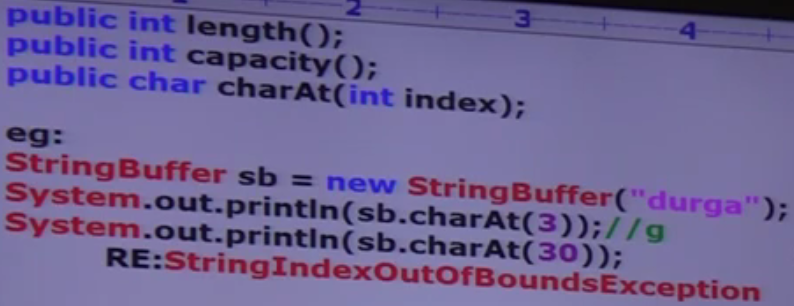
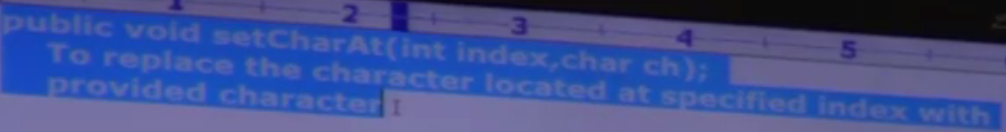
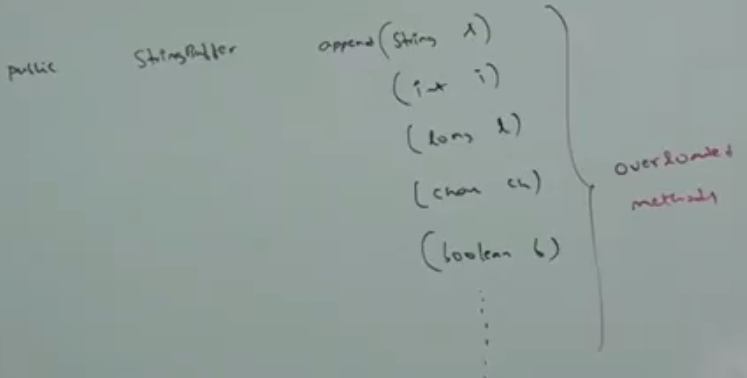
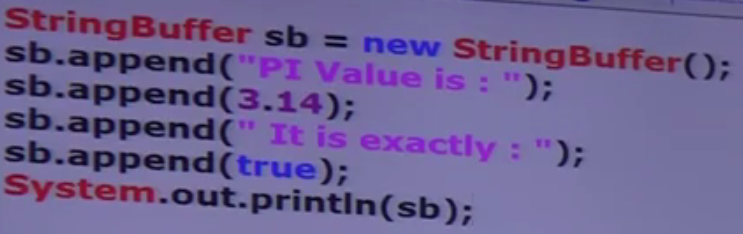
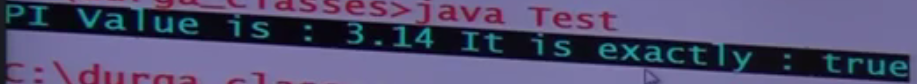
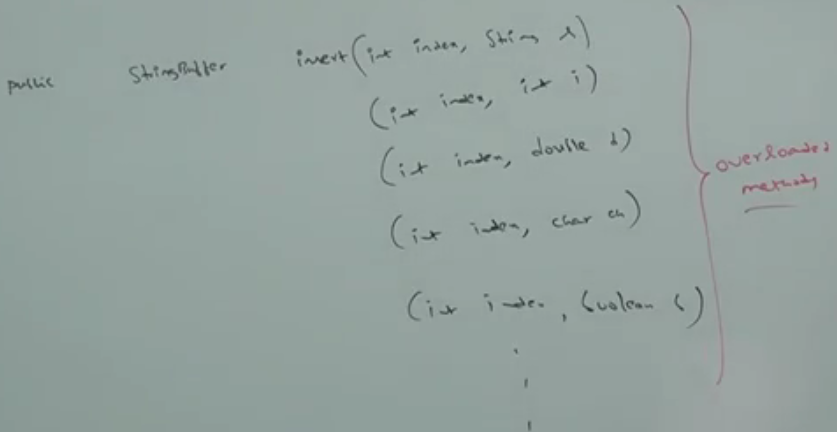
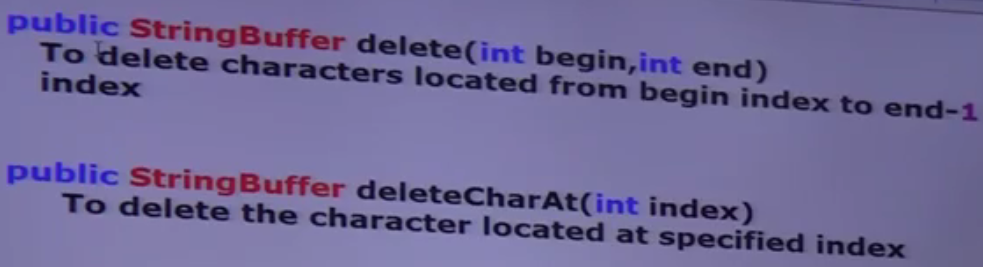
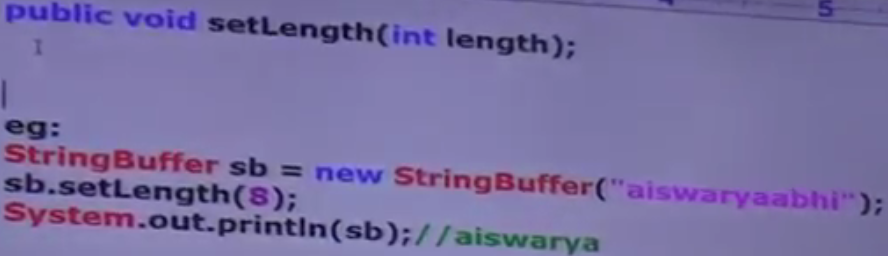
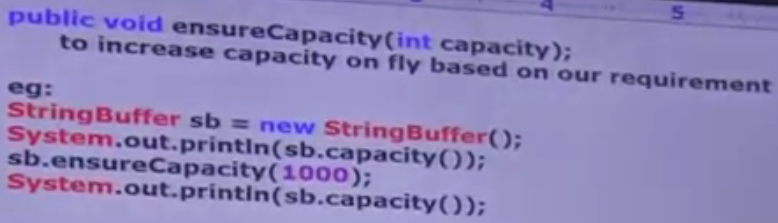
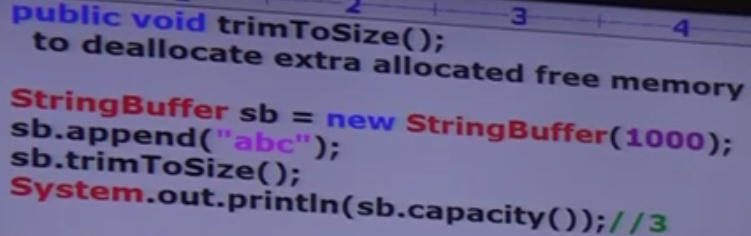
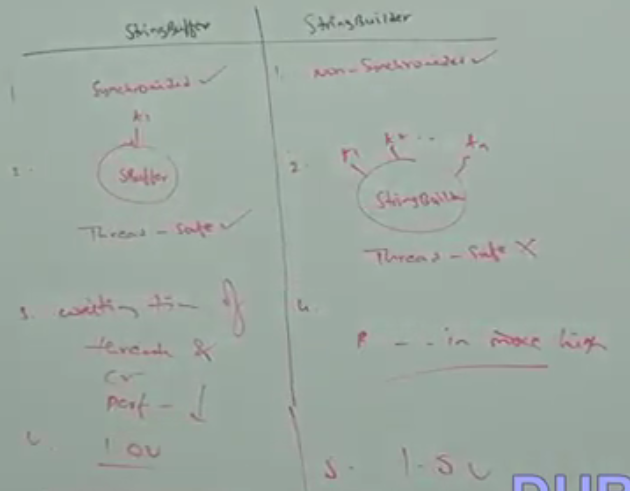
StringBuffer

1. If content is fixed and will not change frequently, then it’s recommended to go for **String** otherwise go for **StringBuffer**.
2. String: For every change a new change a new object is created affecting the system.
3. In case of StringBuffer, every change is performed on the existing object.
4. **Capacity**:
   1. **Default** : 16
   2. **17th Character entry**: New Capacity = (Current Capacity + 1) \* 2;
5. **Constructors**:
   1. new StringBuffer(): Creates empty StringBuffer object with default initial capacity (16). When StringBuffer reaches a max capacity, a new StringBuffer object is created with New Capacity = (Current Capacity + 1) \* 2;
   2. new StringBuffer(int initialCapacity)
   3. new StringBuffer(String);  
      **capacity** = string.length() + 16;
6. **Methods**:
   1. 
   2. 
   3. 
7. Example:  
     
   
8. 
9. 
10. 
11. 
12. 
13. 
14. d

StringBuilder

1. Every method present in StringBuffer is synchronized. Hence only one thread at a time is allowed to operate on StringBuffer object which may create performance problem. To handle this requirement, Sun People introduced StringBuilder concept in 1.5 version.
2. **String**builder is exactly same as StringBuffer except the following differences
   1. 
3. d

String vs. StringBuffer vs. StringBuilder

1. d

StringBuffer method Chainning