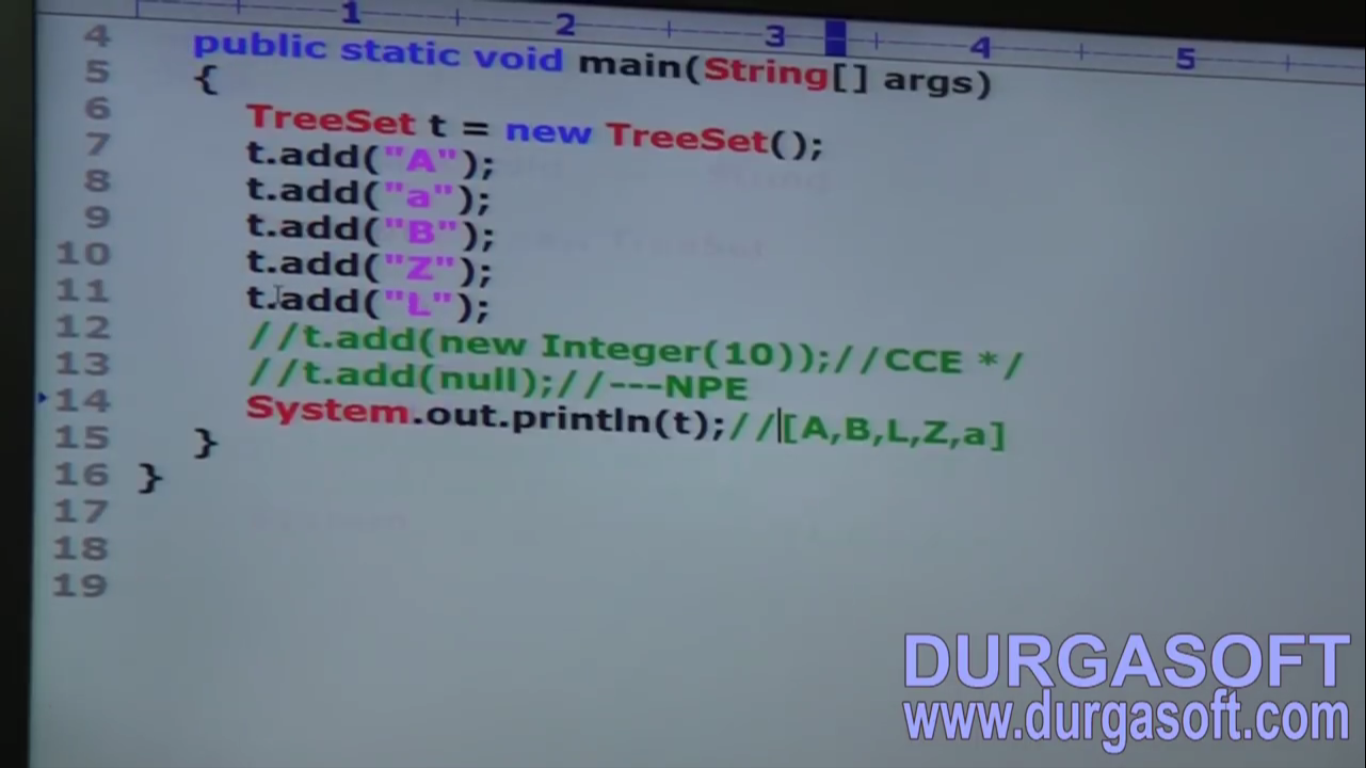
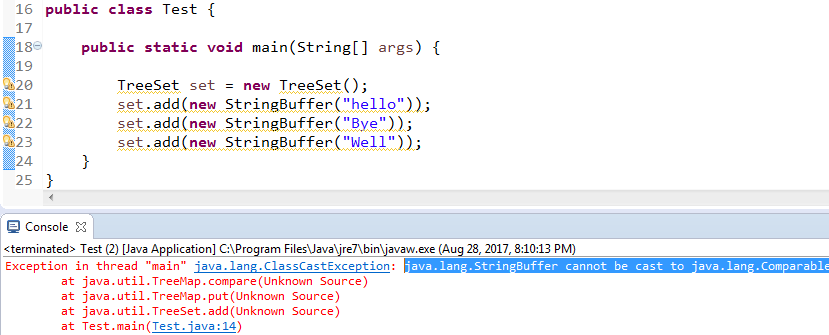
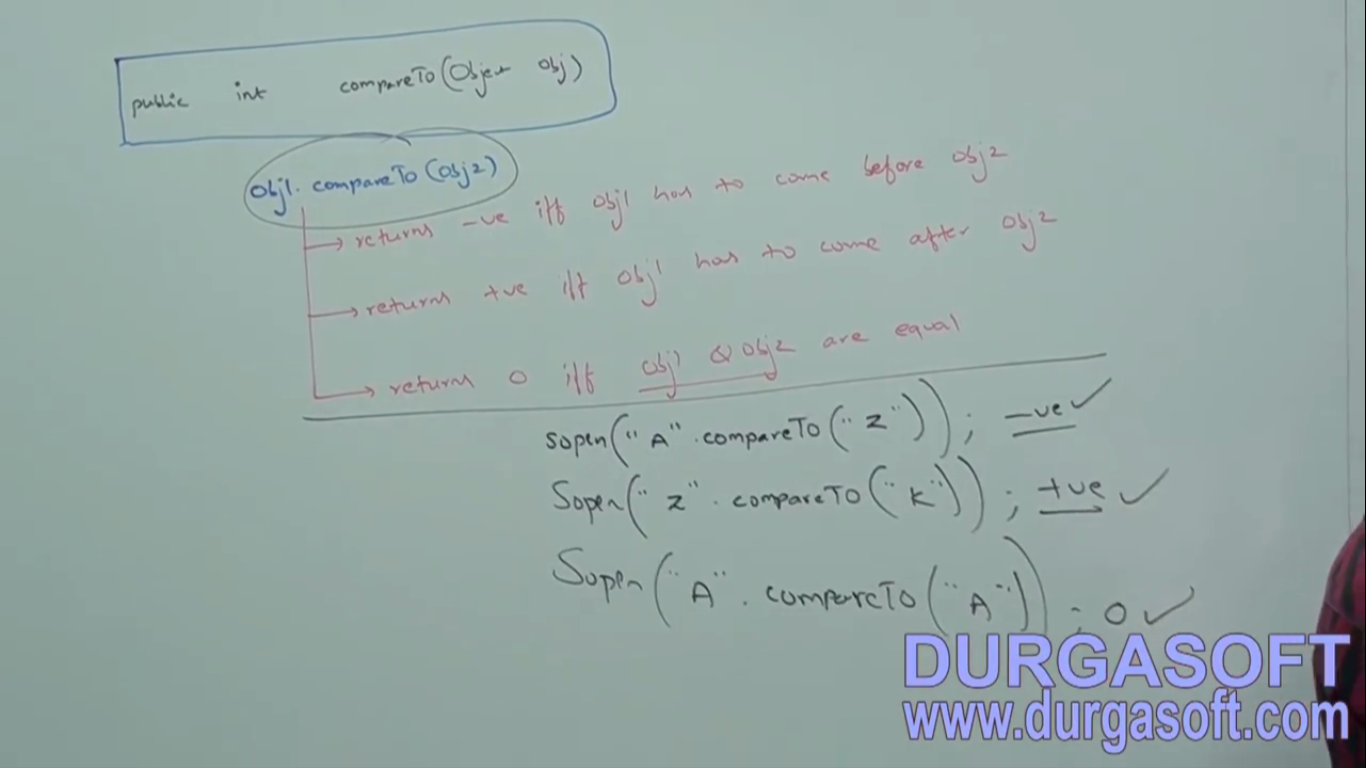
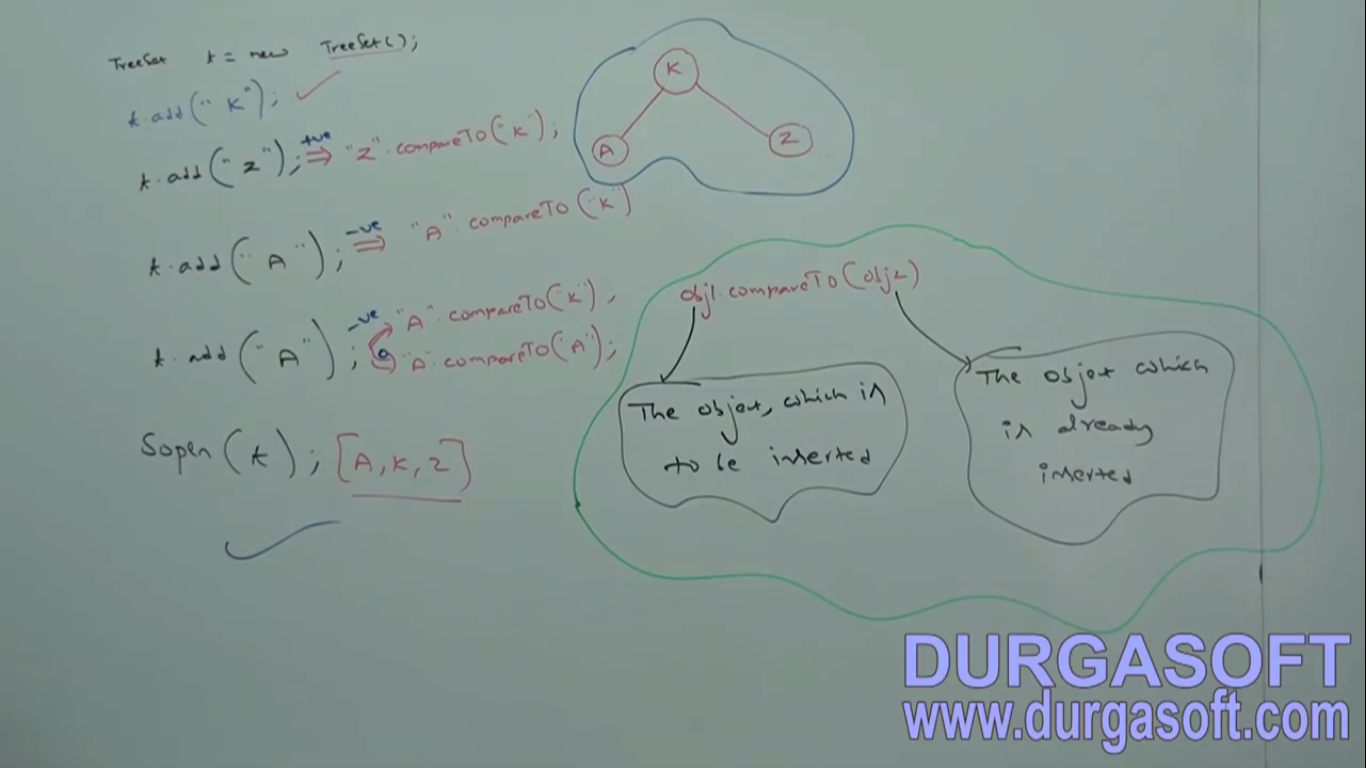
TreeSet(C)

1. **Properties**:
   1. **Underlying DS**: Balanced Tree
   2. **Duplicates:** Not Allowed
   3. **Insertion Order**: Not Preserved as objects will be inserted based on hash code.
   4. **Heterogeneous objects**: Not Allowed. Because objects are inserted according to some sorting order using comparable or comparator which converts passed object into same type if they are not same then ClassCastException is thrown.
   5. **Null Insertion**: Only once till 1.6v. In 1.8 not allowed 🡪 **NullPointerException**
2. **NOTE**: All objects will be inserted based on Some Sorting Order. It may be **default natural sorting order** or **customized sorting order**
3. **Constructor**:   
    **NOTE**: The same with TreeMap.
   1. **TreeSet()**:
      1. Where all objects will be inserted according to Default Natural Sorting Order.
   2. **TreeSet(Comparator)**:
      1. Creates an empty TreeSet object where objects will be inserted according to **Customized Sorting Order** specified by **Comparator object.**
   3. **TreeSet(Collection)**
   4. **TreeSet(SortedSet)**
4. **\*\*\*\*\*Remember: Jatin🡪** Default Natural Sorting Order means corresponding class implements **Comparable(I)**
5. **Example:**
6. **Defualt Natural Sorting**:
   1. Means the corresponding class should implement **Comparable(I)** otherwise **ClassCastException**
   2. When using Comparable (I) there is one rule.
      1. The objects should be homogenous.
7. d
8.   
   Added objects to TreeSet must be Comparable(I) when using default natural sorting order. Internally TreeSet tries to convert StringBuffer object into Comparable but StringBuffer doesn’t implement Comparable (I). Due to which we get **ClassCastException**

**NOTE**: String, All Wrapper classes implement Comparable (I)

1. d

Comparable (I)

1. **java.lang**
2. It contains only one method 🡪 compareTo(Object)
3. **public int compareTo(Object)**
   1. **Why Return type of compareTo() is int but not Boolean**
      1. **Because Boolean covers only two cases whereas in case of object comparisons, we have three cases**
         1. **Both equal**
         2. **Calling object is greater than passed object.**
         3. **Passed object is greater than calling object.**
      2. **d**
   2. **How to cover all three cases with Integer return type.  
      NOTE: greater means🡪 after, less means🡪 Before**
      1. **Negative: means calling object is less than passed one.**
      2. **0: means both are equal**
      3. **Positive: means the calling object is greater than passed one.**
   3. ****
4. **Internal Functionality of TreeSet  
   **
5. **d**

Comparator

1. **Application**: If Default Natural Sorting Order is not available or we are not satisfied with default natural sorting order, then we can go for customized sorting order by using Comparator.  
   