* **OverRide ‘equals()’ Rules:**
  + Should be symmetric. Meaning if x.equals(y) then y.equals(x) should also be true.
  + Reflexive: x.equals(x) should always be true.
  + Transitive: If x.equals(y) is true and y.equals(z) is true then x.equals(z) also should be true.
  + Should be consistent. x.equals(y) should be true as long as one has not changed its state.
  + Signature: public Boolean equals(Object obj).
* hashCode() Rules:
* **Object Comparison:**
  + Interface Comparable is in package lang and interface Comparator is in package util.
  + Comparable:
    - Comparable is used for natural ordering.
    - compareTo()
  + Comparator:
    - Comparator is for customized.
    - compare()
  + Comparable and Comparator shouldn’t be implemented by the same class.
* Collection Framework:
  + Jdk 1.0
    - Util
      * Vector
      * Hastable
      * Arrays:
        + Arrays are not dynamic in nature. They don’t grow of shrink.
        + They are not heterogeneous.
  + Collections(Java 2):
    - Like arrays are also data structures but are dynamic and heterogeneous.
    - Collections support objects of any class but do not support primitives. Primitives can be added by wrapping.
  + Any class extends the object class which has methods like .hashcode(), .equals(), .tostring().
  + Collection is sometimes called as Container.
  + It is typically used to store, retrieve, manipulate and transmit aggregrate data.
  + Framework has 3 things:
    - Interfaces:
      * They represent abstract data types.
      * List, Set, Map.
      * They allow to represent and manipulate collection objects irrespective of the details of the implementations.
    - Concrete Implementations:
      * They represent re-usable data structures.
      * Eleven Implementations are most widely used.
    - Algorithms:
      * They represent re-usable functionality.
      * Sort, shuffle,reverse,Binary-Search.
  + Class Heirarchy:
    - Collection
      * List
        + Vector.
        + ArrayList
        + LinkedList.
      * Set
        + HastSet: Fastest Implementation.
        + LinkedHashSet: Slower than but if elements are to be removed and added on regular basis this should be used.
      * Sorted Set
        + TreeSet: Slowest Implementation. Natural Ordering. Thus while using this method the objects of user defined classes should have a compareTo method defined and implement the interface comparable.
      * Queue
        + Linked List.
        + Priority Queue.
    - Map
      * HashTable
      * HashMap
      * LinkedHashMap
      * Sorted Map.
        + TreeMap.
      * Has Key: Object of any class whose hashtable() is overridden.
      * Has Value: Object of any class.
      * Keys of HashTable and HashMap donot have guarantee of order.
* Mathematical set Abstraction: