Sets and Maps

- Set – an interface for a collection of data
- A set does not necessarily have an order, nor does it contain duplicates
- Basic functions include adding, removing, locating, and listing elements
- The HashSet and TreeSet classes are two data structures that implement the Set interface
- Use HashSet when the order of its elements does not matter
- Elements of a HashSet must provide a hashCode method; many classes in the Java API already do
- Constructor
    - Set<Integer> aSet = new HashSet<Integer>();
- Adding and removing: aSet.add(12);, aSet.remove(12);
- aSet.contains(11) returns true if the element is contained in the set
- TreeSet sorts its elements, so the element type should either implement the Comparable interface or have a Comparator constructed with the set
- Constructor
    - Set<Integer> aSet = new TreeSet<Integer>();
- Use an iterator to go through the set
- Iterator<Integer> iterate = aSet.iterator();
- Since the elements are not ordered specifically, there is no previous() method, only next() and hasNext()

- Maps – keeps track of associations between keys and values
- A key cannot have more than one value associated with it, but multiple keys can lead to the same value
- A map can be a HashMap or a TreeMap
- Map<String, Integer> luckyNumber = new HashMap<String, Integer>();
- Map<String, Integer> luckyNumber = new TreeMap<String, Integer>();
- luckyNumber.put("Alexa",13); adds an association to the map
- get() returns the value associated with the key
- luckyNumber.remove("Alexa"); removes both the key and associated value
- keySet() returns a set containing all the map's keys

- Hashing is the term used for quickly finding elements in a data structure, and does not use a linear search
- Hashing uses a hash table, which is used to implement sets and maps
- Hash function – computes the hash code (an integer value) from an object
- hashCode() returns an objects hash code
- The hash code is used in the same way as an index position is used in an array
- It is possible for collisions to occur, in which two or more objects have the same hash code
- Elements that collide are put into a bucket, a linked list of elements with the same position value
- With no or few collisions, the time for finding, removing, or adding elements takes O(1) time.
- Less than 30% capacity in the hash table is typically recommended to reduce collisions