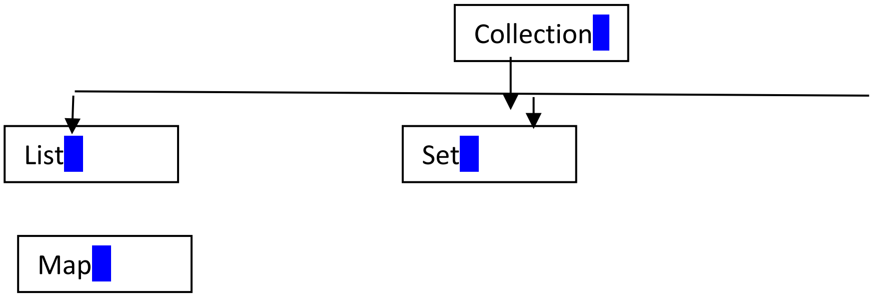
Java Collections. Are the data structures that holds the similar type of data.

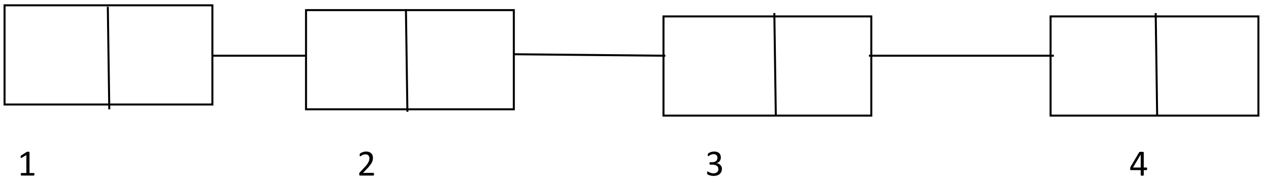
Collections => Class . Have utility methods

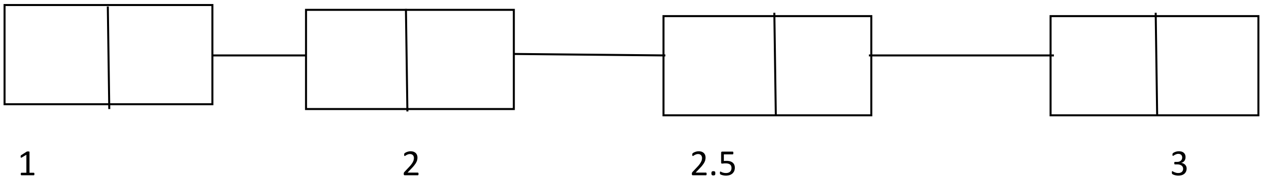
Collection => Interface

Both are defined in java.util package



- List
  - ArrayList
    - Resizable
    - Maintains insertion order
    - Allows the duplicate
    - Use Iterator to iterate or lambda function.
    - Write program to add any type of object in list.
  - Vector
    - Thread safe
  - Linkedlist
    - Node (value and address for next node)
    - First node = head





- Set
    - TreeSet
        - Does not allow duplicate data
        - Maintains sorted order (Ascending) for predefined types.
        - Comparator can be used to provide sorting order at runtime.
        - Must provide Comparable/Comparator for sorting logic in case of user defined type.
    - HashSet
        - Does not allow duplicate data.
        - Maintains random order.
- Map : Stores the data in key value pair
    - HashMap
        - <key ,value>
        - Does not allow duplicate key
        - Allow one null key
        - Is not sorted
        - Does not maintain insertion order
    - TreeMap

Importance of equals() & hashCode()

a. Used to override hashcode collision and duplication of object
b. Must override if we need to store the UserDefiend Object in Map and Set
c. Contract Between HashCode() & equals()
    i. Two different objects can have the same hash value
    ii. But vice-versa not true

Lambda

- Used to provide implementation on functional interface.

- @FunctionalInterface = interface with only one abstract method
- Helps in having less code.
- Syntax - (argument)-> { body }
  - Argument :- It can be empty or can have multiple arguments.
  - Arrow-token :- **(->)** used to link argument list to body
  - Body :- can have single line or multiple line.
  - Ex:- syntax
    - () -> {
      }
  - For iteration also can be used as for ex :- list.sort(obj -> …..)

1. Sorting
   a. Comparable<T>
      i. Provide implementation(logic) at class level
      ii. Can not have multiple sort sequence at runtime
      iii. compareTo (T obj)
   b. Comparator<T>
      i. Provide runtime implementation for sort sequence
      ii. Can have multiple implementations for sort sequence
      iii. Lambda can be used.
      iv. compare(T obj, Tobj1)

Streams

- Stream does not store elements.
- Do not change the original data
- Two types of operations
  - Intermediate
    - Map()
      - List.stream().map(x->x*x).collect(collectors.toList());
    - Filter()
      - List.stream().filter(x->x.startsWith(“A”)).collect(collectors.toList());
    - Sorted()
      - Write program and make use of sorted() in streams.
  - Terminate

- Collect()
    - Practice on getting data into set
- Foreach()
- Reduce()

- Optional
    - Make use with findFist()
    - Use isPresent() to avoid exception
    - Find list of employees grouping based on length of name > 3 and name starts with A