```
29/07/2022

Java Notes Day7  Weekend --> Association assignment --> 100 classes

Day 7
1. Collection         --> Innovate tree map
2. Generic classes

--------------------------------------------------------------------------
------------------------------------------
There are different types of bags
     > All inheriting from an abstract Bag super class

Iterable - Ability to travel
|Iterator iterator()   --> Book class --> add Book number, Book author,
Book number, Edition, No of pages, Book price
Collection       --> Serialize these (Books) via hashset
|                --> Innovate comparator interface with your uniquely
designed object - collections.sort - takes comparator as an object
----------------
|          |
List-Duplicate   Set-Unique
|          |
ArrayList
>PhoneLog  TreeSet >ChemicalElenent
LinkedList HashSet
>PhoneContact

--------------------------------------------------------------------------
------------------------------------------

ArrayList  - Call log in the phone, order cannot be swapped but members
can be deleted
           - very difficult to delete a member in between
           - continuously growing in one direction - only at the end
           - Like elevator - access only unidirectional

0 - -->emp object
1 - -->stu object
2 - -->fli object
3 - -->ba object
.
.

--------------------------------------------------------------------------
------------------------------------------

LinkedList - Like a train compartment
           - Like stairs - access from each member
           - Speed is not a concern(efficiency) - flexibility is the main
concern
           - Phonebook is a linked list - in the memory

head

node1 > 100
|data1     |200 |

node2 > 200
|data2     |300 |
node3 > 300
|data3     |400 |

node4 > 400
|data4     |... |


insert node5 between 2 and 3
      node5 > 500
      |data5     |400 |

node1 > 100
|data1     |200 |

node2 > 200
|data2     |500 |

node3 > 300
|data3     |400 |

node4 > 400
|data4     |... |

---------------------------------------------------------------------------
-------------------------------------------

Tree set   - chemical elements or unique keys like emlpoyee numbers
           - searching becomes easy, and sorted data is output
           - efficient to access and flexible to store

ROOT 50
|
-----------------
|           |
L 30        R 70
---------   ---------
|     |     |     |
L 20  R 40  L 65  R 80

IN ORDER - 50, 30, 70, 65, 80, 20, 40
OUT ORDER - 20, 30, 40, 50, 65, 70, 80 - L C R - this is the out order

---------------------------------------------------------------------------
-------------------------------------------

Hash set   - example a bookhelf consisting of different compartments
           - No definite in and out order
           - maintains a unique copy
           -

---------------------------------------------------------------------------
-------------------------------------------

Generic class
Component orthogonal space

                       DataType
                       |
                       |x - ButterMilk, FriedRice
```

```
                |
          ----------------------------------
          |y         |           |z
          Container  |           Algorithm
          |          |i          |
          Bottle          Iterator   drink();
          Plate      |           eat();
                     Straw
                     Spoon
```

```
Content determines the container

class StringValues
      container - String
      content - String x, y
      algoithm - void print()
      iterator - direct (In array we use for loop for iterating)

generic container - 98% code is similar
            - class Name <T or any data member>
            - T is a compile time decision
            - It is a type of anyType or an data type (it is of raw type)
            - Raw data type, the references to generic class must be
parametrized

Wrapper class - wrapper can be changed but the content within remains the
same

--------------------------------------------------------------------------
------------------------------------------

Iterator iterator()
{
   return arr;
}


interfaces -> Flower, Fragrance, Perfume
classes -> Rose, Lily, RoseFragrance, RosePerfume

Interface based coding

every time a method within an interface has return type of an interface,
an object must be created and returned in the implementing class
only a reference of the interface can be created and not the object of
the interface
the reference of the interface can only store the object of a class that
has implemented the interface

--------------------------------------------------------------------------
------------------------------------------

LinkedList<PhoneContact> arrLog = new LinkedList<PhoneContact>();
Iterator<PhoneContact> it = arrLog.iterator();

interface Iterator<T>
{
   Iterator iterator();
{
   return iterableRef;
```

```
}
}

class


Iterable - Ability to travel
|Iterator iterator()
Collection
|
-----------------
|          |
List-Duplicate   Set-Unique
|          |
ArrayList  TreeSet
LinkedList HashSet


Iterator iterator()
{

}


MyList mlist = new MyList();

Iterator i = mlist.iterator();
i.


=========================================
A
interface Iterable
{
      Iterator iterator();
}

class MyList implements Iterable
{
      Iterator iterator() {
            Iterator i = new ListIterator();
            return i;
      }
}
===========================================
B
interface Iterator
{
      boolean hasNext();
      Object next();
}
class ListIterator implements Iterator
{
      boolean hasNext() {

            return true/false;
      }
      Object next() {

            return obj;
```

```
        }
}
=======================================================
```

add true keyword comma after the file location to append