Collection(I) vs Collections(Class)

Collection is a interface in java that contains many methods without body.

These many methods which are inside Collection interface gives so much of flexibility to develop java applications.
sorting, searching, deleting, ...

List Interface(Methods)

=> We should use list interface whenever we want insertion order to be preserved / maintained.
==> List interface allows duplicate elements to be stored.

Ex: Shopping Cart ==> Add a Laptop, Add a Mobile(3), Add a Book,.....50 items

Set Interface(Methods)

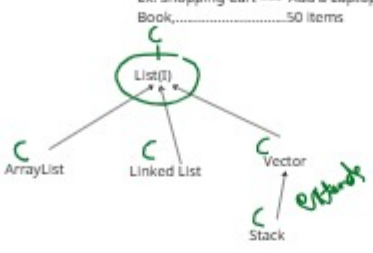
=> We should use set interface whenever we want insertion order not to be preserved / maintained.
==> Set interface not duplicate elements to be stored.

Ex: Playlist feature in your music app => Set Playlist (add 10 songs) ==> No duplicated

List Interface(Methods)

=> We should use list interface whenever we want insertion order to be preserved / maintained.
==> List interface allows duplicate elements to be stored.

Ex: Shopping Cart ==> Add a Laptop, Add a Mobile(3), Add a Book,.....50 items



All these classes which are implementing various interfaces, uses various data structures and algorithms

1) ArrayList

use data struture is Resizeable Array

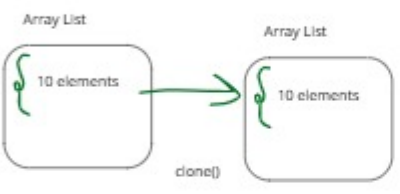
1) Insertion order is maintained.
2) Dupliacte elements are allowed.
3) Same tyope of data or different type of data.
4) Array list stores the data internally inside an RESIZEABLE ARRAY.

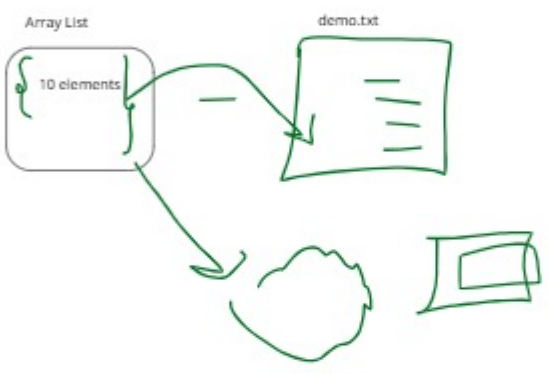
```
List al = new ArrayList(); ==> 10 block will be created
      al.add(10);
      al.add(20);
      al.add(30);
      al.add(20);
al.add(10);
 al.add(20);
      al.add(30);
      al.add(20);
al.add(40)
al.add("Hello")
al.add("Welcome")
```

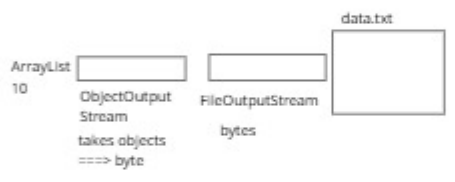
Old Cpaacity * 1.5 + 1
10 * 1.5 + 1 ==> 16

```
Class ArrayList implements Serializable, Cloenable, RandomAccess
{


}
```

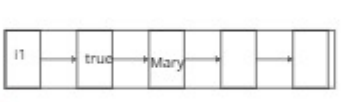


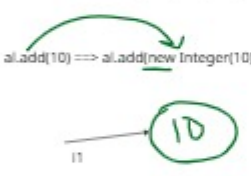




All the classes inside collection will store the data in object format

```
al.add(10)
al.add(true)
al.add("Mary")
```

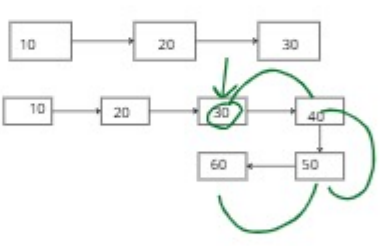




Wrapper Classes
Byte
Short
Integet
Float
Double
Boolean
Character
String
.....

converting primitive data in to a object formar

1) Maintain insertion order
2) Add duplicates



2) ArrayList is best used when we want to do read / retrieve operation.
3) ArrayList is worst if we want to perform insertion and dleetion opertion

Linked List

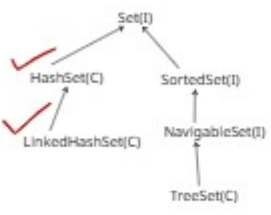
1) Maintain insertion order
2) Add duplicates
3) Linked List Internally uses Doubly Linked List
4) Linked List is usesful when the operation is insertion and deletion
5) Linked List is not useful when the operation is Reading

Vector
==> It is 99.99% same as ArrayList
==> ArrayList is not thread safe and vector is thread safe.

I want to create an application which should contain all the names starting with all the alphabtes

```
ArrayList<String> data = new ArrayList<String>();
data.add("A...");
data.add("A...");
data.add(100);
```

Set
=> No insertion order maintained
=> No duplicated allowed

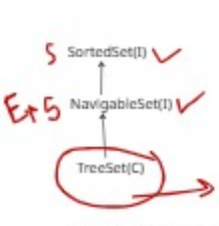


HashSet
=> It is a class
=> It cannot store duplicate elements
=> Insertion order is not maintained
=> It can store both homogeneous and heterogenous types of data.
=> HashSet uses HashMap(Dict) internally

```
HashSet hs = new HashSet();
hs.add(100);
hs.add(300);
```

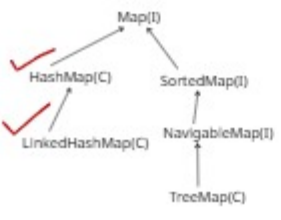
```
public HashSet()
{
map = new HashMap();
}
```

LinkedHashSet
=> It is a class
=> It cannot store duplicate elements
=> Insertion order is maintained
=> It can store both homogeneous and heterogenous types of data.
=> HashSet uses HashMap(Dict) internally



1) TreeSet stores the data in sorted order(Ascending Order).
2) Duplicate elements are not allowed.
3) Since TreeSet is also getting methods from NavigableSet, we can have navigation facility availbale.
4) TreeSet allows only Homogenoues data to be stored.

If we want the TreeSet to follow custom sorting order, then we need to use Comparator(I) ==> compare()



CharacterisDigit(ch) Checks if the character is a digit (0-9)
CharacterisLetter(ch) Checks if the character is a letter (a-z or A-Z)
CharacterisLetterOrDigit(ch) Checks if the character is a letter or digit
CharacterisUpperCase(ch) Checks if the character is uppercase
CharacterisLowerCase(ch) Checks if the character is lowercase
CharactertoUpperCase(ch) Converts character to uppercase
CharactertoLowerCase(ch) Converts character to lowercase
CharacterisWhitespace(ch) Checks if the character is a whitespace

```
String name = "Raju"
```

Collection(I) vs Collections(Class)

Collection is a interface in java that contains many methods without body.

These many methods which are inside Collection interface gives so much of flexibility to develop java applications.
sorting, searching, deleting, ----------

List Interface(Methods)

=> We should use list interface whenever we want insertion order to be preserved / maintained.
==> List interface allows duplicate elements to be stored.

Ex: Shopping Cart ==> Add a Laptop, Add a Mobile(3), Add a Book,----------50 items

Set Interface(Methods)

=> We should use set interface whenever we want insertion order not to be preserved / maintained.
==> Set interface not duplicate elements to be stored.

Ex: Playlist feature in your music app => Set Playlist (add 10 songs) ==> No duplicated

List Interface(Methods)

=> We should use list interface whenever we want insertion order to be preserved / maintained.
==> List interface allows duplicate elements to be stored.

Ex: Shopping Cart ==> Add a Laptop, Add a Mobile(3), Add a Book,----------50 items

All these classes which are implementing various interfaces, uses various data structures and algorithms

1) ArrayList

use data structure is Resizeable Array

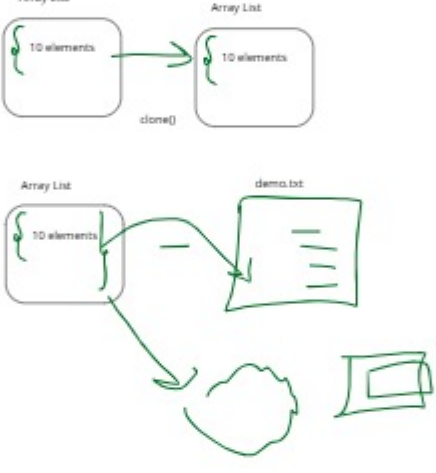
1) Insertion order is maintained.
2) Duplicate elements are allowed.
3) Same tyope of data or different type of data.
4) Array list stores the data internally inside an RESIZEABLE ARRAY.

```
List al = new ArrayList(); ==> 10 block will be created
      al.add(10);
      al.add(20);
      al.add(30);
      al.add(20);
al.add(10);
 al.add(20);
      al.add(30);
      al.add(20);
al.add(40)
al.add("Hello")
al.add("Welcome")

Old Cpasicity * 1.5 + 1
10 * 1.5 + 1 ==> 16
```

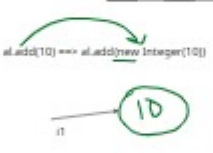
```
Class ArrayList implements Serializable, Cloneable, RandomAccess
{


}
```



All the classes inside collection will store the data in object format

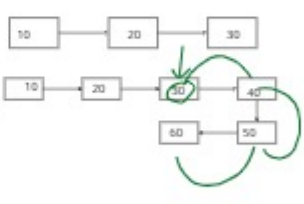
```
al.add(10)
al.add(true)
al.add("Mary")
```

Wrapper Classes
Byte
Short
Integet
Float
Double
Boolean
Character
String
----

converting primitive data in to a object formar

1) Maintain insertion order
2) Add duplicates



2) ArrayList is best used when we want to do read / retrieve operation.
3) ArrayList is worst if we want to perform insertion and dleetion opertion

Linked List

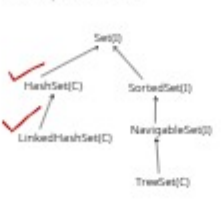
1) Maintain insertion order
2) Add duplicates
3) Linked List Internally uses Doubly Linked List
4) Linked List is usesful when the operation is insertion and deletion
5) Linked List is not useful when the operation is Reading

Vector
==> It is 99.99% same as ArrayList
==> ArrayList is not thread safe and vector is thread safe.

I want to create an application which should contain all the names starting with all the alphabtes

```
ArrayList<String> data = new ArrayList<String>();
data.add("A...");
data.add("A...");
data.add(100);
```

Set
== No insertion order maintained
== No duplicated allowed



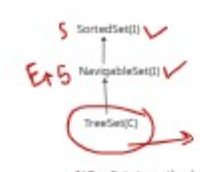
HashSet

== It is a class
== It cannot store duplicate elements
== Insertion order is not maintained
== It can store both homogeneous and heterogenous types of data.
== HashSet uses HashMap(Dict) internally

```
HashSet hs = new HashSet();
hs.add(100);
hs.add(300);
```

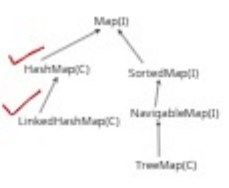
```
public HashSet()
{
map = new HashMap()
}
```

LinkedHashSet
== It is a class
== It cannot store duplicate elements
== Insertion order is maintained
== It can store both homogeneous and heterogenous types of data.
== HashSet uses HashMap(Dict) internally



1) TreeSet stores the data in sorted order(Ascending Order).
2) Duplicate elements are not allowed.
3) Since TreeSet is also getting methods from NavigableSet, we can have navigation facility availbale.
4) TreeSet allows only Homogenoues data to be stored.

If we want the TreeSet to follow custom sorting order, then we need to use Comparator(I) ==> compare()



Character.isDigit(ch) Checks if the character is a digit (0-9)
Character.isLetter(ch) Checks if the character is a letter (a-z or A-Z)
Character.isLetterOrDigit(ch) Checks if the character is a letter or digit
Character.isUpperCase(ch) Checks if the character is uppercase
Character.isLowerCase(ch) Checks if the character is lowercase
Character.toUpperCase(ch) Converts character to uppercase
Character.toLowerCase(ch) Converts character to lowercase
Character.isWhitespace(ch) Checks if the character is a whitespace

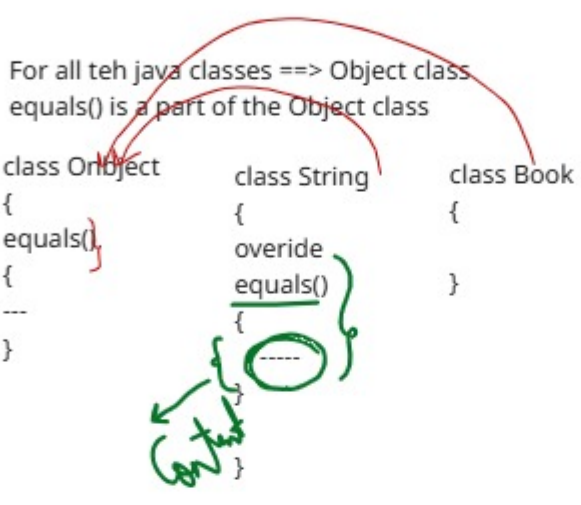
String name = "Raju"

Comparator

=> It is a interface.
=> Using comparator we can implement custom sorting order on any no of fields.

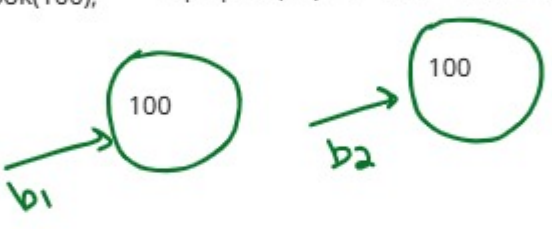
Comparable

=> It is a interface.
=> Using comparator we can implement custom sorting order on only one field.



Book b1 = new Book(100);
Book b2 = new Book(100);

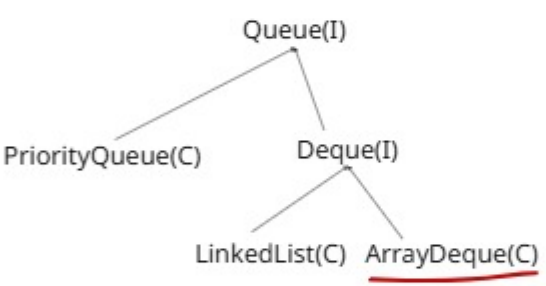
equals() ==> content
bq.equals(b2) ==> 100 == 100 ==> true



Hash Code
498931366

Hash Code
2060468723

HashSet ==> [b1, b2]

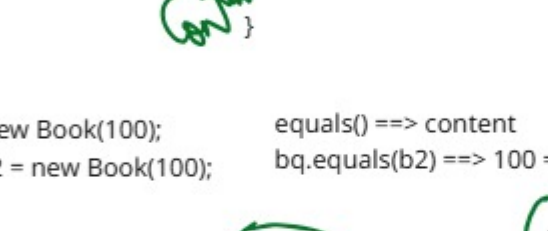


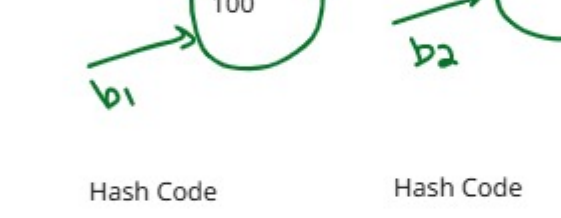
Queue(FIFO)
=> Scheduling tasks(The task that comes first, that task only should be taken care first).
=> Printing tasks(First Come First Server)
=> Tasks can be added in Queue always in the end.

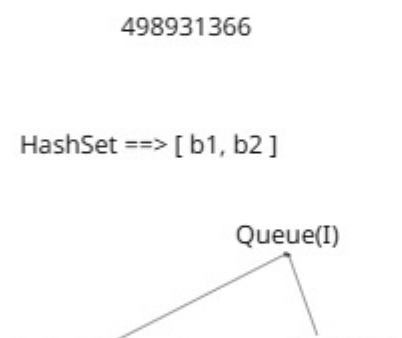
De(Double Ended)Queue
==> Tasks can be added in the last or evening the beginning

Priority Queue
=> It is a class that implements Queue interface.
=> This stores the data based on the priority
=> It interally uses min-heap
=> Priority Queue always keeps the element with most priority in the starting.

100, 50, 150, 25, 75, 125, 175







[25, 50, 125, 100, 75, 150, 175]

["stepsTaken": 1000, "CalrouesBurned": 30]