Date
Page

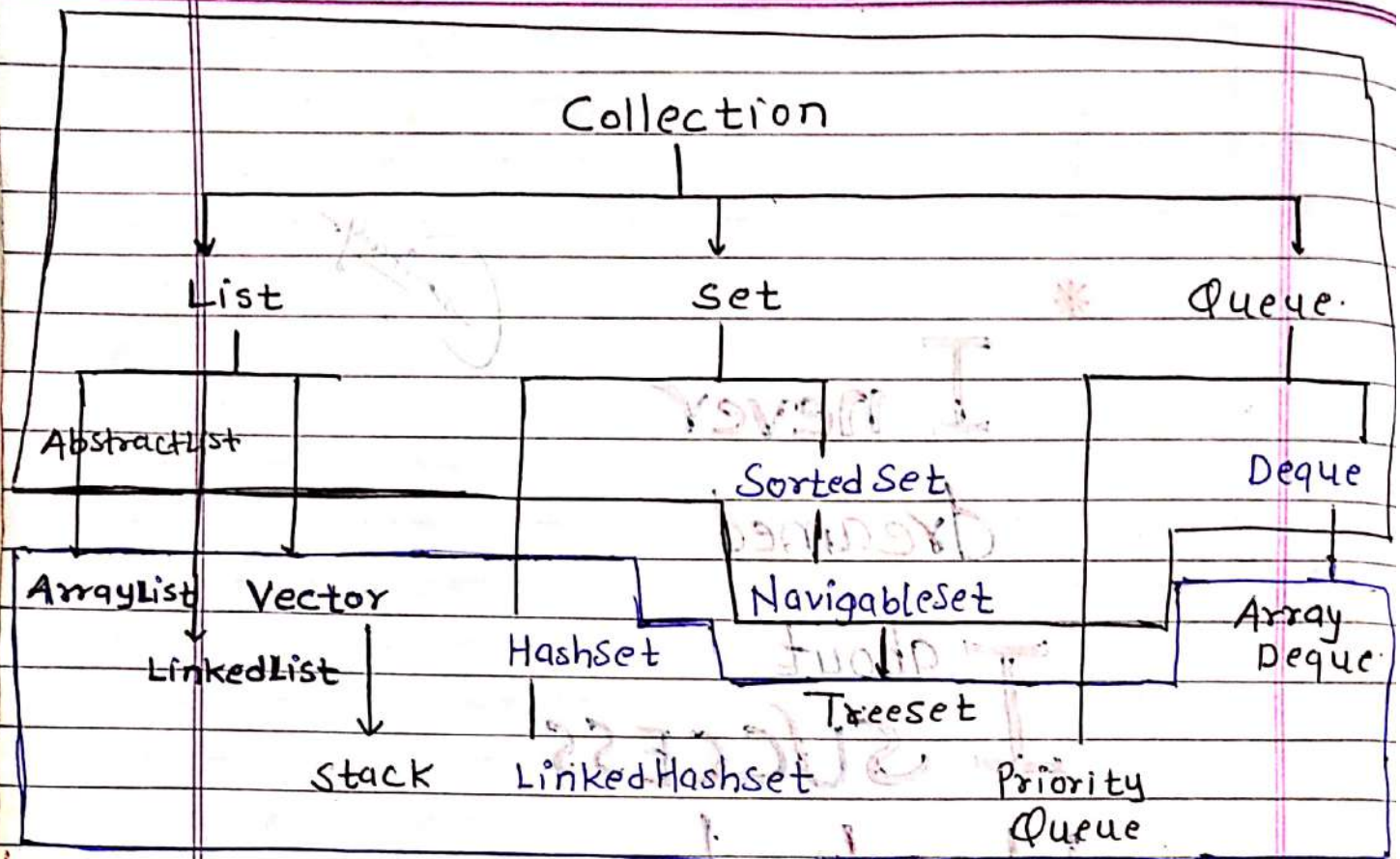
Srinivas

I never dreamed about SUCCESS Worked for it.

26/07/21

Srinivas

Interface



JDK

of method is factory method

immutable object kon create karte hai

List.of()
set.of()
map.of()

List.of(v1, v2, v3...)
Set.of(v1, v2, v3...)
map.of(k1, v1, k2, v2, k3, v3...)

Imp Interview Question

Collection Framework

Date Page

⇒ Types of Data structure:- 1. Primitive Data structure.

eg: int, char, long, float, double, boolean, byte.

Sequence: boolean, char, byte, short, int, long, float, double.

2. Non-Primitive Data structure.

→ String, Files, Arrays, Collection Framework (Collection, map)
① ②

⇒ Application of Arrays:-

1. Arrays are used to store multiple data with single variable name which reduces the number of variables to be declared.

2. Arrays are used to develop some algorithms like Bubble sort, Insertion sort, Selection sort etc → ex-setrg

3. Arrays can be used to perform matrix operations.

4. Arrays can be used to perform matrix operations.

5. Arrays can be used for CPU scheduling

⑥ Arrays are used to implement data structure for Example stack, ArrayList, Queues etc

⇒ Point to remember for Array & Collection Framework :-

1. → Array is Java language feature inbuilt support provided by sun Microsystems.

⇓

Array basically, Array Java ka inbuilt feature hota hai Jo Java ne allready banaya hua hai ⇒ we have to develop algorithms to sort or insert or delete etc

⇒ Collections are API features

⇓

collection Jo hote hai API features hote hai. API ka kya matlab hai enho ne pridefined kuch set banaye hue bhai kis chij ke Class and Interface ke or un classes and Interfaces ke Andher ~~classes and Interfaces~~ bahut saare method hai Jinko directly use karna ~~padta hai~~ hota hai

Agar hum logo collection framework use kar rhe hai na to hum khudh se koi logic path lagane ki need nhi hai

⇒ It provides predefined classes and interfaces and methods by which we can easily iterate or delete or sort the elements.

2: → Array can store primitive (int, char etc) and non-primitive (objects)
↓ array bhi hai Jo objects store karwa skta hai

⇒ Collection Framework can store only non-primitive data type (objects)

Wrapper class = (Integer i = new Integer(rollno1);
↳ [101] object hai

(3) → Array can store only homogeneous data types i.e array can store only similar type of data.

⇒ Collection Framework can store heterogeneous data i.e we can store different type of data.

eg:- class HomeLoan
{
}
class CarLoan
{
— — —
}
class BankApp
{
main()
{
HomeLoan hl1 = new HomeLoan();
CarLoan cl1 = new CarLoan();
}
}

Agar ye object create hua → reference (hl1)

| hL1 | CL1 | hL2 |

ye store nhi hoga (CL1)

ye store kyu ki ho Jayega dusre data types agya hai

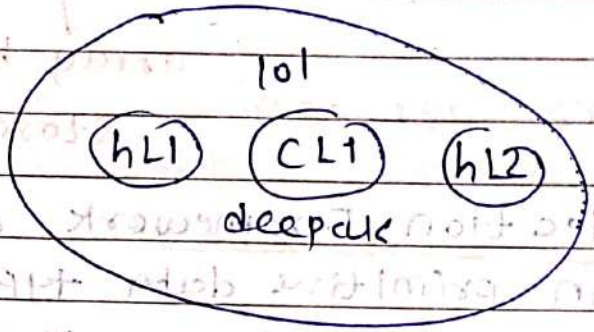
ye store ho Jayega

Array Eg.

| hl1 | cl1 | |

kyu ki dusre data aya store nhi hoga

Collection eg:



(4) → ~~Array~~ The size of an array cannot be increased or decreased according to our requirement at runtime.

→ The size of collection can be increased or decreased according to our needs.

(5) → Array are not good with respect to memory.

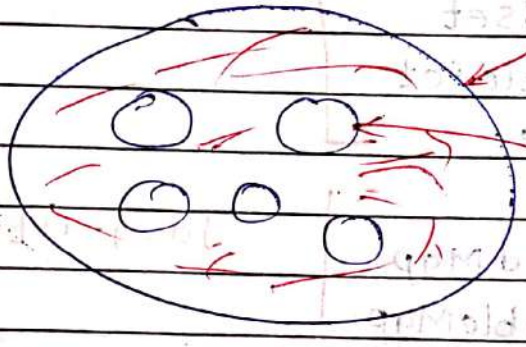
→ 'Collection framework are very good with respect to memory'

(6) → Arrays are good by performance wise

→ Collection are not good by performance wise.

* What is Collection Framework ?

Ans: Collection Framework consists of 2 words i.e Collection and Framework.



= Collection → collection wah ek single entity ya phir single objects ko represent karta hai jiske under multiple data store ho skta hai

⇒ Defination: Collection is a single entity or an object which contains multiple data.

⇒ Framework → represents the Library.

→ Collection framework is the set of classes and interfaces that implement commonly reusable collection data structure.

→ Collection framework contains 2 main parts :-

✓ 1. Java.util.collection.
✓ 2. Java.util.Map.

Map →: "9 key interfaces" of collection. Framework.

1. Collection
2. List
3. set
4. Sortedset
5. Navigableset
6. Queue

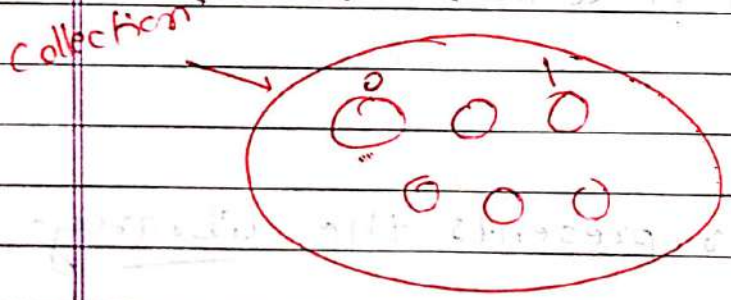
Java.util.~~Map~~ Collection

7. Map :
8. SortedMap
9. NavigableMap

Java.util.Map.

* Hierarchy of collection Framework :-

→ In Collection, we can store the data directly. but in Map we can store the data in key-value pairs



Map

Key	Value
101	deepak
102	amit
103	rahul

root Interface is collection.
↓

27/07/21

Collection:

Collection:-

→ collection is an interface which is present in Java .util package.

→ syntax: public interface Collection<E> extends Iterable <E> {-}

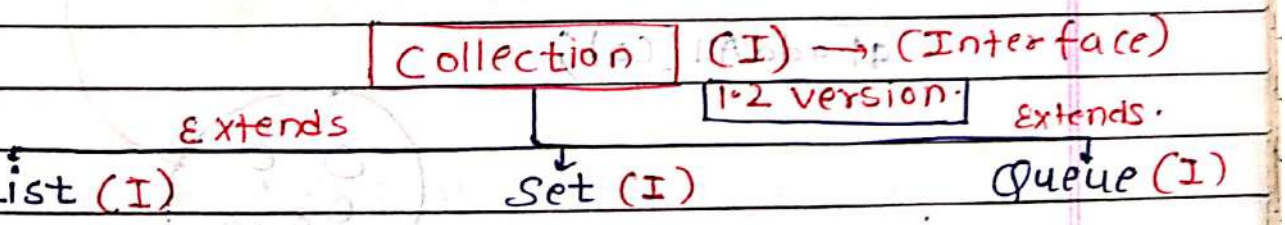
→ collection was introduced in JDK 1.2 version.

→ Collection is an object which is used to represent a group of individual objects as a single unit.

→ Collection interface is the root interface of collection Framework.

→ There is no concretee class which implements the collection interface. directly but there are interfaces which inherit the collection interface.
i.e List, Set & Queue.

→ Hierarchy of collection interface:-



Collection Interface ka object create nhi kar skate hai

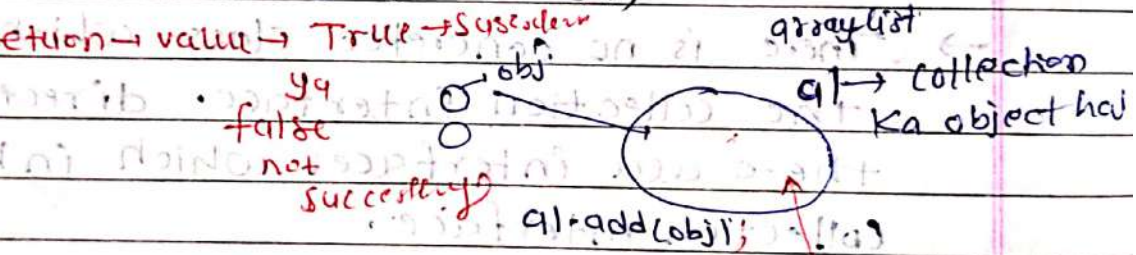
→ Collection Interface Jo hai banaya kya hai?

Jitne bhi common ~~hai~~ methods hai (kis chij ke liye) hai ek collection Interface ya phir classes ke liye collectionframework Interface and class ke liye wah sare ke sare common method hai collection ke Adher banaye gye hai

→ Collection interface contains most common methods which are applicable for any collection object.

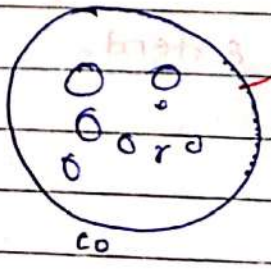
→ Methods of collection Interface :-

1. boolean add(object obj);



2. boolean addAll (Collection c);

al.addAll(co);



Agar App log ek collection ke object ko ~~(~~ dusre collection me store karwana chahte ho to addAll(collection c) method ko use karlega

Interface ke Andher body. nhi hoti hai

{ }

3. boolean remore (object obj)

al.remove(xyz);

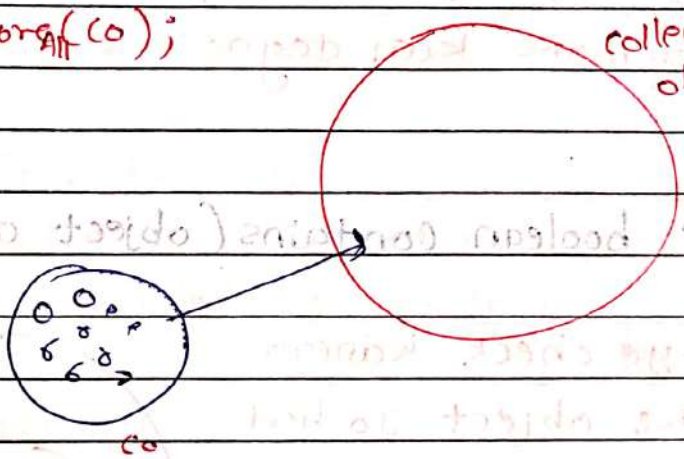
collection object al

xyz

Collection object me se remove karwaan ka kam karta hai

4. boolean removeAll (Collection c);

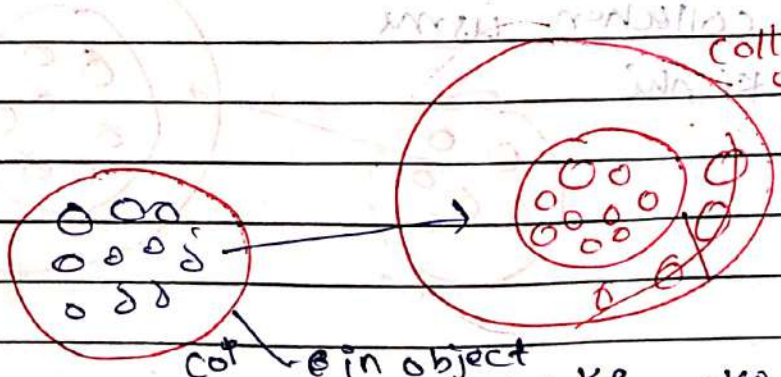
al.removeAll(co);



5. default boolean removeIf (-) { - }

Agar kisi bhi method ek Age default laga de to usKi body ~~ban jati hai~~ or usKi Interface ke undher create kar skate hai

6. boolean retainAll (Collection c);

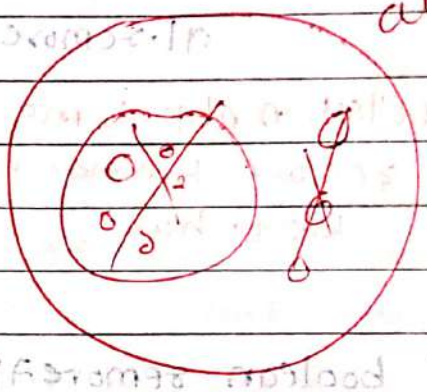


co ~~e~~ in object ko bhhod ke baki sabko remove kar dege.

Collection is golobal nature ka hota hai

7. void clear();

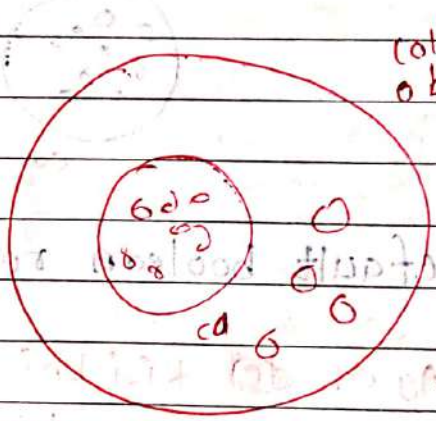
Collection ke andher jitne bhi object hai wah pura ke pura objecto empty kar dega.



Sabse ke sare objecte ko remove kar dega.

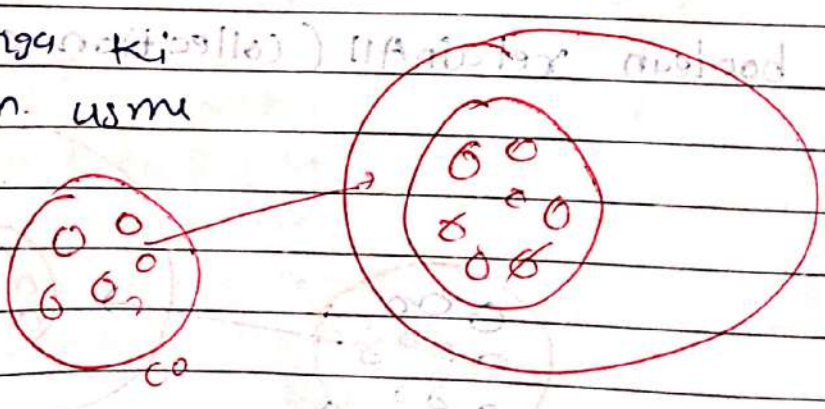
8. boolean contains(object obj);

→ ye check karega ke object jo hai wah us wale object me hai ki nhi hai to → true value return karega. nhi to false karega.



9. boolean containsAll (collection c);

ye check karega ki sare collection usme hai ki nhi



10) boolean isEmpty();

ql.isEmpty(); Empty check karwata hai hai to → return value. True karega nhi hai to → false karega.

11. int size();

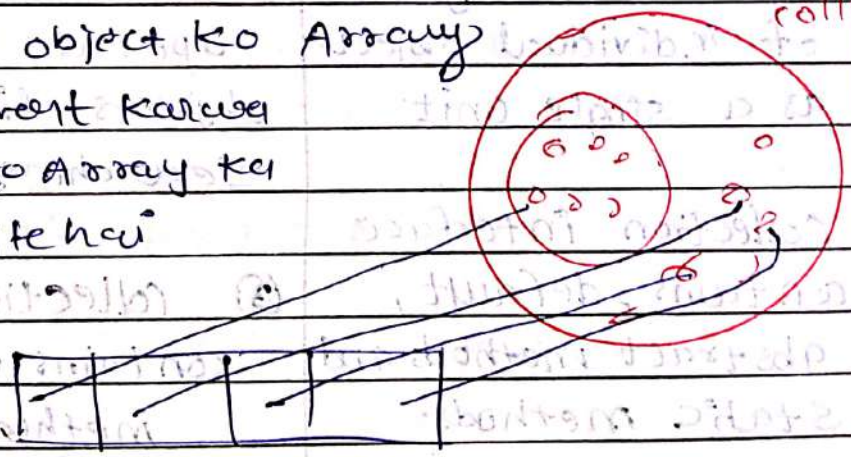
collection object ka size return karega.

12. Iterator iterator();

collection se Element ko get karwana hai to Iterator method use karte hai

13. object toArray();

collection object ko Array me convert karwa ~~to~~ to Array ka use karte hai



al.toArray();

14.) boolean equals (object obj.); objects pass karwate hai

15.) int hashCode ( );
object class ka method hai

What is difference between Collection & Collections.

Collection	Collections
1.) Collection is an Interface.	① Collections is a utility class.
2.) Collection is an object which is used to represent a group of individual objects as a single unit.	② Collections defines several utility methods that are used to operate on collection objects like sorting, searching etc.
③ Collection interface contains, default, abstract methods and static methods.	③ Collections class contains only static methods.

owx-kdzg-meq

* What is Utility class in Java?

Ans utility class is also known as helper class which cannot be instaintated इनस्टान्ट शियेट

↓

→ Jis class ka hum log object create nhi kar skate us class ko hum log utility class bolte hai.

→ Utility class contains only static methods.

→ Examples are, Arrays, collections.

⇒ How we can create utility class :-

1. declare the class as public and final.

Final, class Jo hai uska inherit nhi karne dega.

2. We have to declare private constructor to prevent object creation.

```
public final class Test
{
    private Test() { }
}
class main
{
    // Test t = new Test(); X nhi kar skta hai kyu ki private lag gya hai
}
```

3. class should contain only static method and does not contain abstract method.

```
public final class Test
{
    private Test(){}
    static void show()
    {
    }
}
class main
{
    Test t = new Test(); // error.
}
```

4. every method should have deep documentation.

= = = = = = = = =

What is Utility method ?

→ Utility methods perform common, often reused methods.

⇒ Utility methods are always static type.

Eg:- Examples are : Sorting, searching, methods, performing String manipulation, Methods connecting to database etc.

List (Interface) & Array List Class

⇒ List Interface :-

→ List is a interface ~~with~~ which is present in Java.util package.

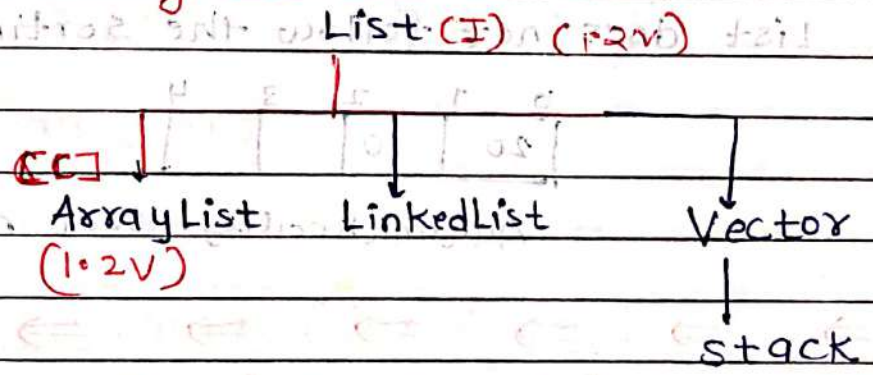
→ List is the child interface of collection interface.

→ Synta : public interface List extends collection { - }

→ List was introduced in JDK 1.2 version.

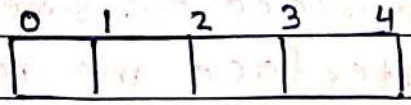
→

Hierarchy of List interface :-



→ Properties of list Interface :-

1. List is an index based Data structure. which means that first element will be inserted at '0' index position.



2. List can store different data types or hetrogeneous elements.

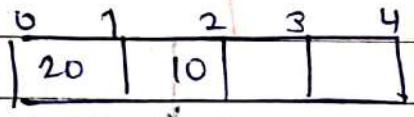
3. We can store dyplicate elements in the List.

10	10	null	null

4. We can store any number of null values in the List.

5. List follows the insertion order. which means the sequence in which we are inserting the elements, in the same sequence we can retrieve the elements.
Jai Jai App log List ke Andher elements provide karege. wase hi element ko usi sequence me element ko retrive karwa skte hai

6. List does not follow the sorting order.



Automatically sort nhi hoga.

⇒ ⇒ ⇒ ⇒ ⇒ ⇒ ⇒

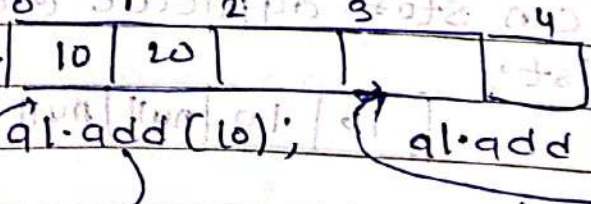
Methods of List Interface:-

1. List contains all the methods of collection interface.

↓

collection Interface ke same method. List Interface me bhi present hoge. kyu ki List Interface collection Interface ko inherits karta hai.

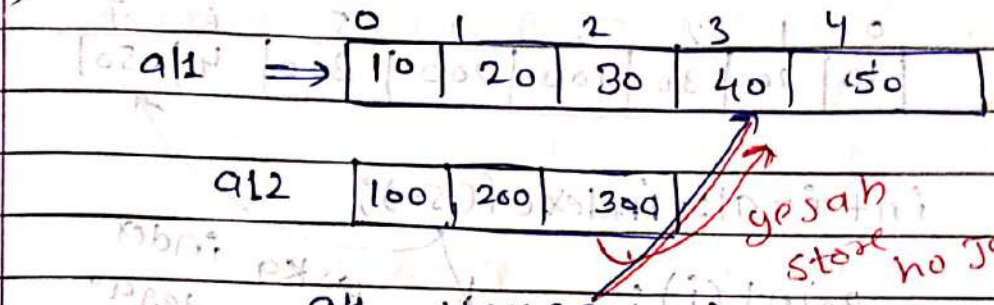
2. void add (int index, object obj);

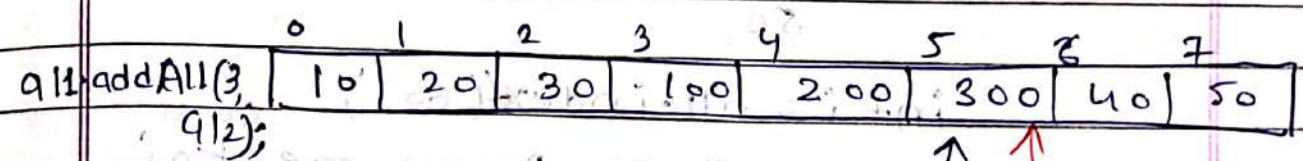


directly element insert karwa dega.

l.add (3, 50);

3) boolean addAll (int index, Collection c);





4) object get (int index);

object obj = all.get(5);
↓
300

5) object remove (int index)

object obj = all.remove (5);

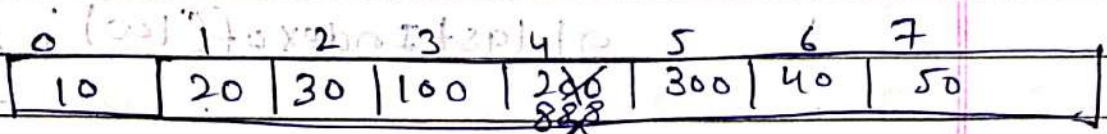
remove karega

object ke andhar store karwa dega.

6) object set (int index, object newobj);

//set method is used to replace the object at given index position.

all.set (4, 888);

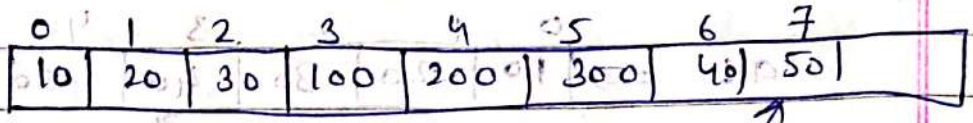


all.set (4, 888) set kar dega.

replace karwa dega.

// It will return the index position of provided object and if object is not found then it will return (-1)

7.) int indexOf(object obj);



int i = a11.indexOf(50);

print(i);

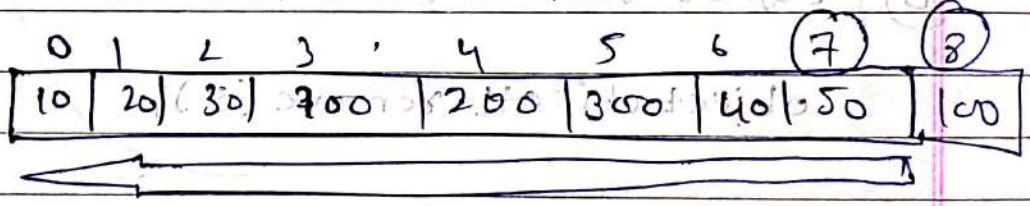
50,

esko index no. degaa

int i = a11.indexOf(60);

Agar Array me koi value nhi milti hai to wah (-1) return value karegaa.

8) int lastIndexOf(object obj);



int i = a11.indexOf(100)

print(i); // 3

a1.lastIndexOf(100); // 8

⇒ ArrayList :-

→ Arraylist is an implemented class of list interface which is present in Java.util package.

↓

Arraylist jo hai List interface ko inherit karti hai. ye saare methods hai jinki body nhi hoti hai.

Arraylist ke method ke Andher body banaya gya hai.

→ Syntax :- Public class ArrayList Extends AbstractList implements List, RandomAccess, Cloneable, Serializable.

→ The underline Data-structure of ArrayList is resizable array or growable array.

→ Arraylist was introduced in J.DK 1.2 version.

⇒ Properties of ArrayList :-

→ 1. ArrayList is an index based Data structure which means that first element will be inserted at '0' index position.

→ 2. ArrayList can store different data types elements or hetrogenieous elements.

→ 3. We can store duplicate elements in the ArrayList.

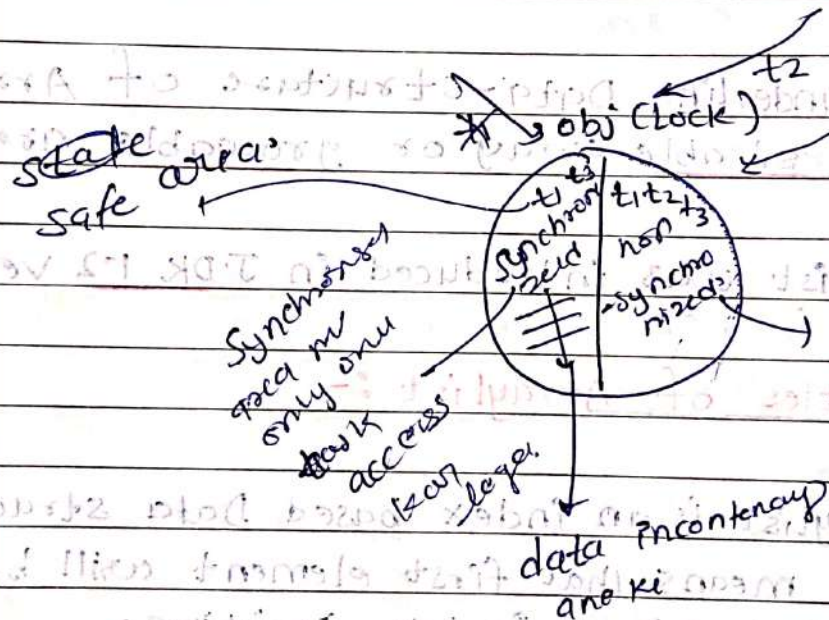
→ 4. We can store any number of null values in the ArrayList.

→ 5.) ArrayList follows the insertion order which means the sequence in which we are inserting the elements, in the same sequence we can retrieve the elements.

→ 6.) ArrayList does not follow the Sorting order.

(above properties are same as List interface)

→ 7.) ArrayList is non-synchronized collection because ArrayList does not contain any Synchronized method.



→ 8.) ArrayList allows more than one thread at one times

→ 9.) ArrayList allows parallel execution.

↓ All methods are Access kar sakta hai es liye parallel execution hai

→ 10) ArrayList reduces the execution time which in turn makes the Application fast.

↓

Application ki jo list hai wah fast ho jayega.

→ 11.) Arraylist is not threadsafe.

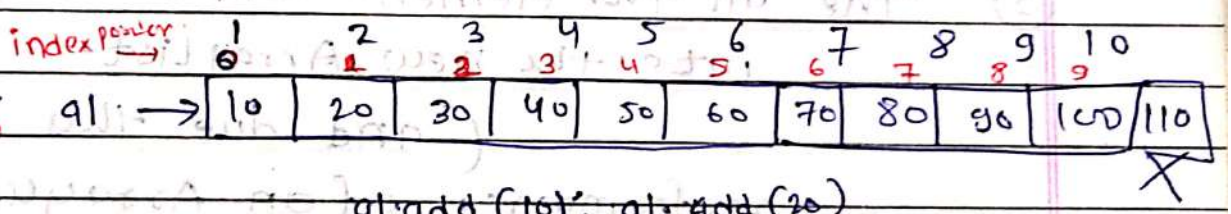
→ 12.) Arraylist does not guarantee for data inconsistency.

↓

Same ke same thread ek sath us data ko access kar skte hai Jiske karna wah Data inconsistency problem occure ho skti hai. — Arraylist me.

All method synchronized nahi hai

* Working of an ArrayList :-



Jab bhi humlog new ArrayList create karte hai usi time yaha par ek new Array list create ho jayega but uski capacity kitni rahegi Humesa 10 rahegi nahi kam hosi na jaida fix rahega.

al.add(10); al.add(20)
al.add(30) .... al.add(90)
al.add(100)
al.add(110)

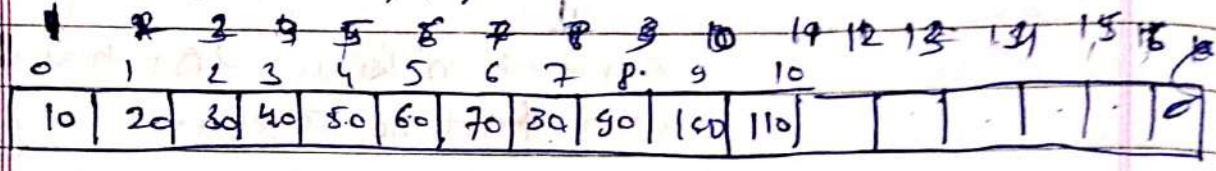
ArrayList al = new ArrayList();

Constructor.

(1) when we create default ArrayList, a new ArrayList with initial capacity 10 is created. (but size is 0)

(Current Capacity * 3 /2) + 1

(10 * 3 /2) + 1 = 16



(16 x 3 /2) + 1) = 25

2) When the ArrayList capacity is full, a new ArrayList will be created with new capacity.

The new capacity is calculated by this formula :-

((Current Capacity * 3 /2) + 1

(3) The all the element will be copied into the new ArrayList.

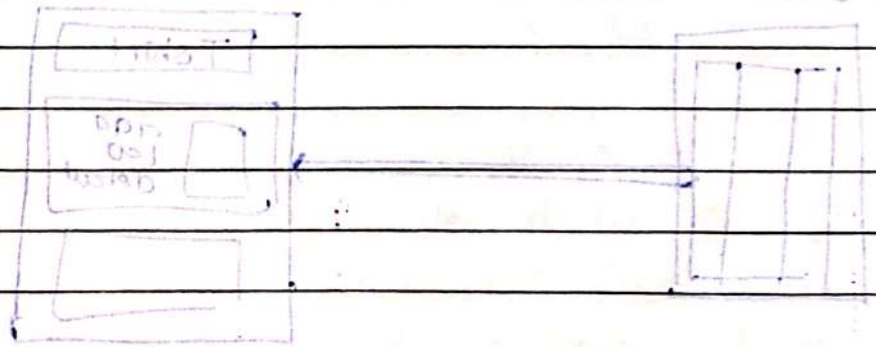
(and due this reason performance of an ArrayList decreases)

(4) Then due to garbage collection, old ArrayList object will be deleted.

(5) When new ArrayList is created automatically, then reference variable will point to the new ArrayList.

5. Then old ArrayList object will be not referenced by any reference and then garbage collection will delete that object.

Note:- There is no way by which we can find the capacity of an ArrayList.



* Contructors of ArrayList :-

1. ArrayList al = new ArrayList ();
   * In this arraylist, a collection object is created whose capacity is 10

2. ArrayList al = new ArrayList (int initialCapacity);
   * In this arrayList, an Arraylist object is created with provided initial Capacity.

3. ArrayList al = new ArrayList (Collection c);
   * in this arrayList, another collection object is copied into new arraylist object.

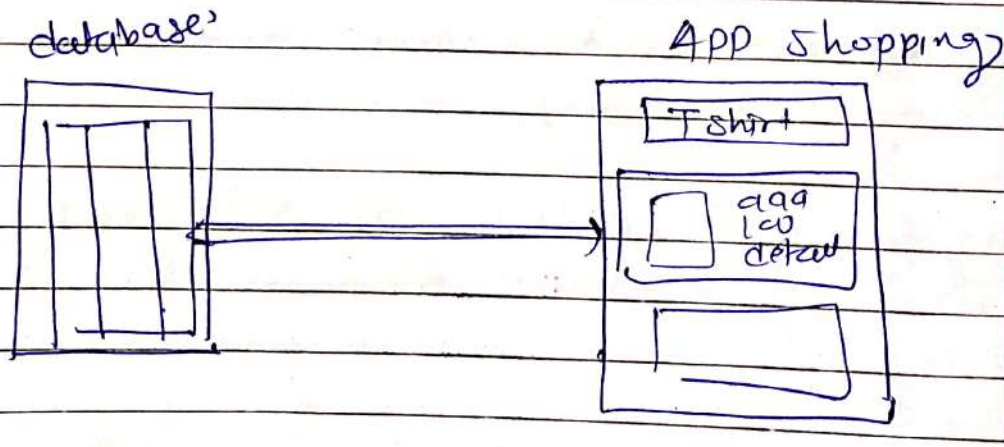
* Programs ArrayList.

→ When we should use ArrayList?

→ when we use retrival operation mostly

Eg: value ko get karwan
ye kam karta hai
App shopping

ArrayList



(retrival operation is fast in case of ArrayList because it implements Random Access interface).

→ When we should not use ArrayList?

→ When we have insertion or deletion operation, then we should not use Arraylist.

Jaha ArrayElement ko se shift karwana hai waha ArrayList ko use nhi karte hai → kyu ki performance slow ho jayegi

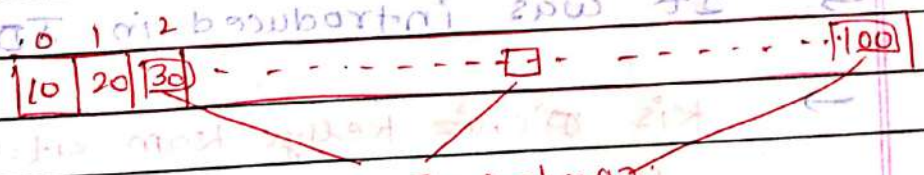
* RandomAccess interface

→ RandomAccess interface is a marker interface that means it does not contain any methods or field (variables)

↓

RandomAccess interface me koi contain ya method bhi hote hai.

→ Marker



Searching

Koi bhi element ko search karwa uski speed fast hogi

Same speed hoti hai kyu ki RandomAccess Interface ki wajah se.

→ The purpose of RandomAccess interface is to retrieve any random element in collection object at the same speed

For Eg:- We have collection object having 1 crore element, we have to search 3rd element or middle element or last element then it will search with the same speed.

→ There are only 2 classes which inherits the RandomAccess interface

1. ArrayList class.
2. Vector class

---

* Cloneable Interface:- hum log kisi bhi object ka clone bana sakte matlab same name ka dusra koi object create kar sakte hai

→ Cloneable interface is also a marker interface.

→ It was introduced in JDK 1.0 version.

→ kis chiz kelye kam ata hai
ye ek new object clone karwane ke liye kam ata hai
ye kisi bhi object ko clone karwane me kam ata hai

→ hum new operator use karne ki need nhi hai

→ directly clone karwa skta hai object ko

→ It is used to clone the objects without using the new keyword.

29/07/24.

~~LinkList~~ LinkedList

→ LinkedList is an implementation class of List interface which is present in Java.util package.

→ Syntax :- public class LinkedList extends AbstractSequentialList implement List, Deque, Cloneable, Serializable {- }

→ The underline data structure of LinkedList is Double LinkedList or Circular LinkedList.

→ LinkedList was introduced in JDK 1.2 version

* Properties of LinkedList :-

1. LinkedList is an index based ~~Data~~ data structure which means that first element will be inserted at '0' index position.

2.) LinkedList can store different data types or hetrogeneous elements.

3.) We can store duplicate elements in the LinkedList.

4.) We can store any number of null values in the LinkedList.

5.) LinkedList follows the insertion order which means the sequence in which, we are inserting the element, in the same sequence we can retrieve the elements.

6. LinkedList does not follow the sorting order. (same points as list interface)

7. LinkedList is non-synchronized collection because ArrayList does not contain any synchronized method.

8. Linked list allows more than ~~one~~ one thread at one time.

9. Linked List allows parallel excecution.

10. Linked List reduces the execution time which in turn makes the application fast.

11. Linked List is not thread safe.

12. Linked List does not gurantee for data consistency.

→ Working of LinkedList:-

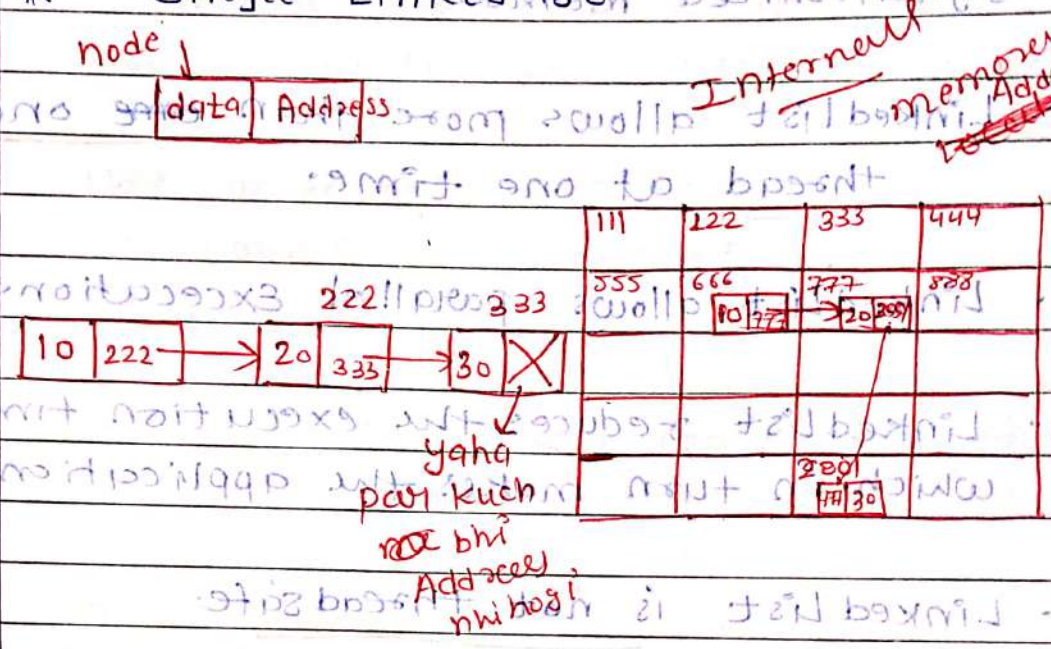
1. Types of LinkedList (all programming language):-

a. Single Linked List
b. Double Linked List
c. Circular Linked List.

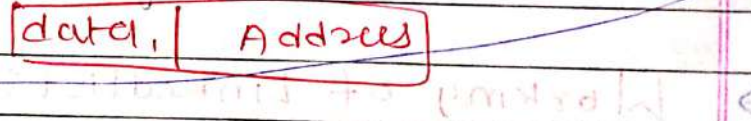
] Java base.

2. Linked List ek Linear data structure in which elements ek not stored in Contigeous memory locations.

* Single LinkedList:

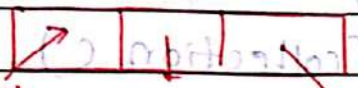


Linked List me Node create hote hai
us me data store hote hai

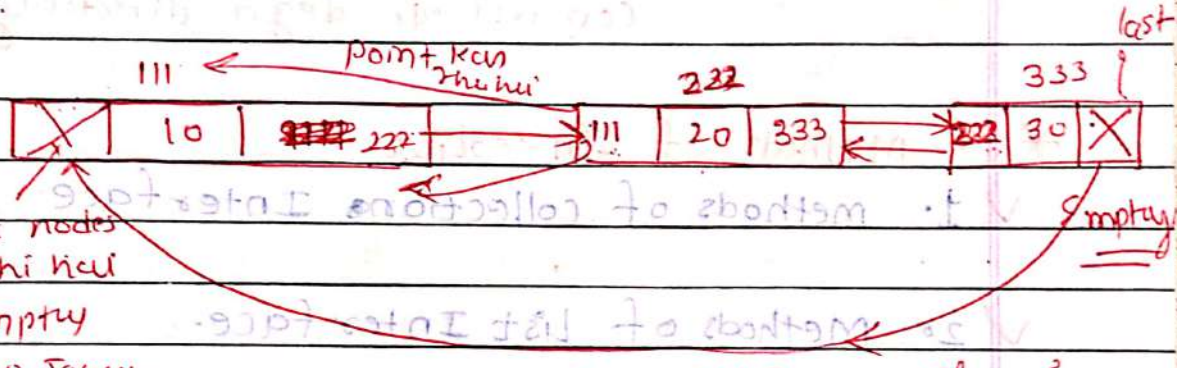


har ek node me Another hamare pas two parts store hote hai
that is data, and Address.

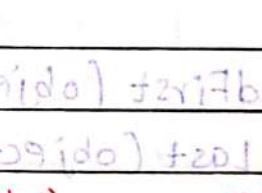
(2) double Linked List



Eg.



(3) Circular List:



(4) Linked List me capacity nhi hoti hai

↓

There is no capacity concept in linkedList like ArrayList.

* Constructors :-

1. public LinkedList () → default constructor only one node create karega.
2. public LinkedList (Collection C)
   ↓
   ArrayList ko LinkedList me convert kar dega directly.

* Methods of LinkedList

✓ 1. methods of collections Interface.

✓ 2. Methods of List Interface.

only Linked List.

3. public void addFirst (object obj)
4. public void addLast (object obj)

5. public object getfirst ()
6. public object getLast ()

7. public object removeFirst ()
8. public object removeLast ()

* When we should use LinkedList ?

Ans: LinkedList is best when we have to use insertion or deletion operations

↓

ye ArrayList me koi bhi element insert karwate the usme shift karna pdta tha, magar LinkedList me magar yesa nhi hai ye do node ke bhich me ye ek node create ha, Jata hai

* When we should not use LinkedList?

Ans: LinkedList is worst in case of retrieval or searching operation (as LinkedList does not inherit ~~testa~~ RandomAccess Interface)

↓

Jab hame bar bar koi element retrive karwana hai ya phir hame koi bhi Element search karwana hai to hum log Linked list ka use nhi karge.

* What is difference between ArrayList & LinkedList?

	ArrayList	LinkedList
1	Arraylist underline data structure is dynamic array or resizable array ~~linkedlist~~.	1. LinkedList underline data structure is double linked list or Circular linked list.
2	ArrayList stores the elements in contigeous memory locations.	2. LinkedList does not store the elements in contigeous memory locations.

3. ArrayList acts as List.	3. LinkedList can acts as List or Deque.
4. ArrayList is good in case of retrieval operation.	4. LinkedList is good in case of insertion and deletion operations.
5. ArrayList is worst in case of insertion or deletion operation.	5. LinkedList is worst in case of retrival operation.

* Vector (legacy class)

→ Vector is an implementation class of List interfaces which is present in Java.util package.

→ Syntax:- public class Vector extends Abstractlist implements List, RandomAccess, eloneable, Sericlizable {}

→ The underline data structure of vector is resizable array or growable array.

→ Vector was introduced in JDK 1.0 version.

→ Vector class is also know as lagacy class. (lagacy class kya hoti hai Jo ki pahle se hi pyrane version ke Andher thi wah class Jo pyrane version ke Andher thi but Jab wah new version ke Andher introduce hui to dubara se to wah restructared ya phir re-engineered ki gyi.)

→ (Legacy class is the class which was formed in previous version and was restructured or re-engineered in new version)

→

* Properties of Vector :

→ Vector is an index based Data structure which means that first element will be inserted at '0' index position.

→ Vector can store different data types elements or hetrogeneous elements.

→ We can store duplicate elements in the Vector

→ We can store any number of null values in the vector.

→ Vector follows the insertion order which means the sequence in which we are inserting the Elements, in the same sequence we can retrieve the element.

→ Vector does not follow the sorting order.

→ Vector is synchronized collection because vector contains many synchronized method.

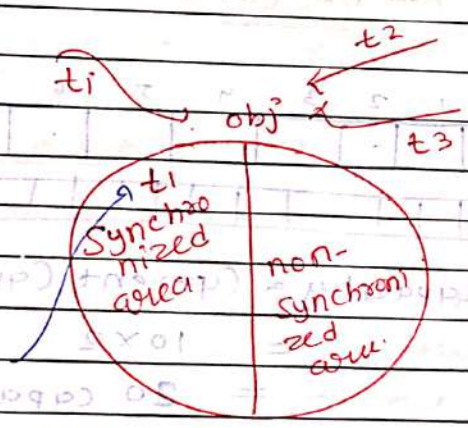
→ Vector does not allow more than one thread at one time.

→ Vector does not allow parallel execution or vector allows sequential execution.

→ Vector increases the execution time which in turn makes the application slow.

→ Vector is threadsafe.

→ Vector gyrantee for data consistency.



→ Agar hamare pass method synchronized hote hai' us case me shirf ata time ek hi thread yaha par. aa payega.
kyu ki eske pass lock ~~kota~~ hai hoga' baki' thread ke pass lock nhi hoga. wah es wale area ko access nhi kar paayege.

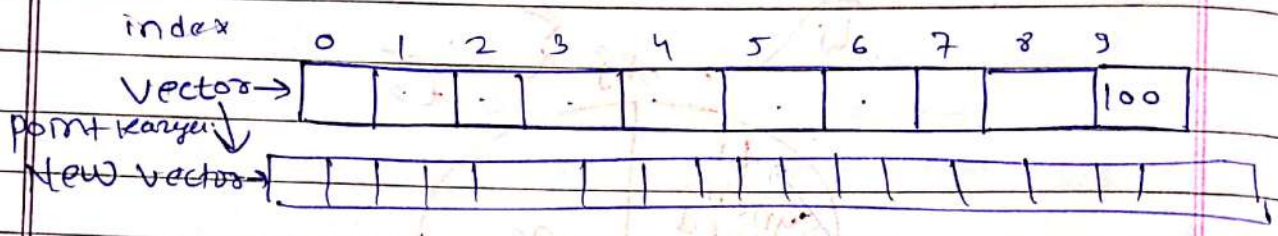
→ multiple thread wah ek sath es wale area ko access nhi kar skte

→ Agar ek sath sare ke sare persion access nhi kar rhe hai to ese kya ho jayega. execute speed jo hai wah slow ho jaigegi' Apprcation slow ho jayegi

→ Synchronized area me multiple thread access nhi nahi deta. to thread sefe ho jayega'

→ Data consistency ki ~~guran~~ garanti deta hai ~~kaise~~. kyu data ~~ya~~ ya ha par ek hi time par. one thread access kar sakta hai.

* Working of Vector :-



New capacity = Current Capacity * 2;
= 10 × 2
= 20 capacity.

(1) When we create a vector, a vector of '10' initial capacity is create.
↓
Jab hum log vector create karege tab wah ek new vector create karega aur uski capacity pahle wale se jada ho jayega.

(2) When the vector is full, then new vector will be created automatically with new capacity = previous capacity * 2;

(3) When new vector is created, the reference variable will point to the new vector and garbage collector will delete the previous vector from the memory.

hum log Vector create kiya hai uski capacity 10 hai hai agar wah full ho jati hai tab and Agar usme humlog ~~new~~ element insert or karwana hai tab wah ek new vector create hoga and pahle wale vector ko pointed nhi karega but (new vector ko pointed karega. and pahle wale vector ko garbage collector delete kar dega.

→ when new vector is created then all the elements from old vector will copied to new vector and then the reference variable will point to the new vector and garbage collector will delete the previous vector from the memory.

Note:- In vector we can find the capacity.

Vector ke Andher capacity findout kar skte hai uske liye method bnaya gya hai already

Constructors of Vector:-

1. public Vector()

↓

ye ek vector create kar dega or eski capacity 10 hai agar vector full ho jata hai to new vector ki capacity 20 ho jaige

2. public Vector (int capacity)

⇓

yaha khud se capacity de skta hu magar app jitna capacity dega utna hi create hoga dusra vector nhi hoga.

3. public Vector (int capacity, int incremental-ratio)

↓ capacity

For Eg.: hum logo ne 100 provide kar diya uske bad new capacity kitni hogi 200 hogi' Agar 200 full hogya tab 400 capacity ho jaigi

(100, 5)

↓ ratio

apki bar ratio me change hoga

Jai ki 105 complete hogya nai tab new, hoga 110, 115, 120

capacity

4. public Vector (Collection c).

② Vector contains all the methods of list interface.

Vector contains all the method of collection interface.

* Methods:

1- public synchronized int capacity()

2. public synchronized void addElement(object obj)

3. public synchronized object firstElement()

4. public synchronized object LastElement()

5. public synchronized boolean removeElement (object obj)

6. public synchronized void removeElementAt (int index)

7. public synchronized void removeAllElement()

* When we should use Vector

Ans: We should use Vector in case of retrieval operations. (Vector inherits the RandomAccess interface)

→ When we should not use Vector

Ans: we should not use vector in case of insertion or deletion of elements.

→

* Difference between ArrayList & Vector?

Array List:	Vector:
(1) ArrayList was introduced in JDK 1.2 version.	(1) Vector was introduced in JDK 1.0 version.
(2) ArrayList is not a legacy class.	(2) Vector is a Legacy class.
(3) ArrayList is non-synchronized collection.	(3) Vector is synchronized collection.
(4) Arraylist allows more than one thread at one time.	(4) Vector does not allow more than one thread at one time.
(5) ArrayList allows the parallel Execution.	(5) Vector does not allow parallel Execution.
(6) Arraylist decreases the execution time which in turn makes the application fast.	(6) Vector increases the execution time which in turn makes the application slow.
(7) Arraylist is not thread safe.	(7) Vector is thread safe.
(8) Arraylist does not gyrantee for the Data consistency	(8) Vector gyrantee for data consistency
(9) In case of Arraylist newCapacity = (present capacity * 3/2)+1	(9) In case of vector new capacity = present capacity *2
(10) In Arraylist we cannot find the capacity	(10) In vector we can find the capacity.

Stack Legacy class

→ stack is the child class of vector class present in java.util package

→ Syntax: public class stack extends Vector {- }

→ stack was introduced in JDK 1.0 version.

→ stack is also known as legacy class.
(Legacy class is the class which was re-engineered or restructured in new version).

→

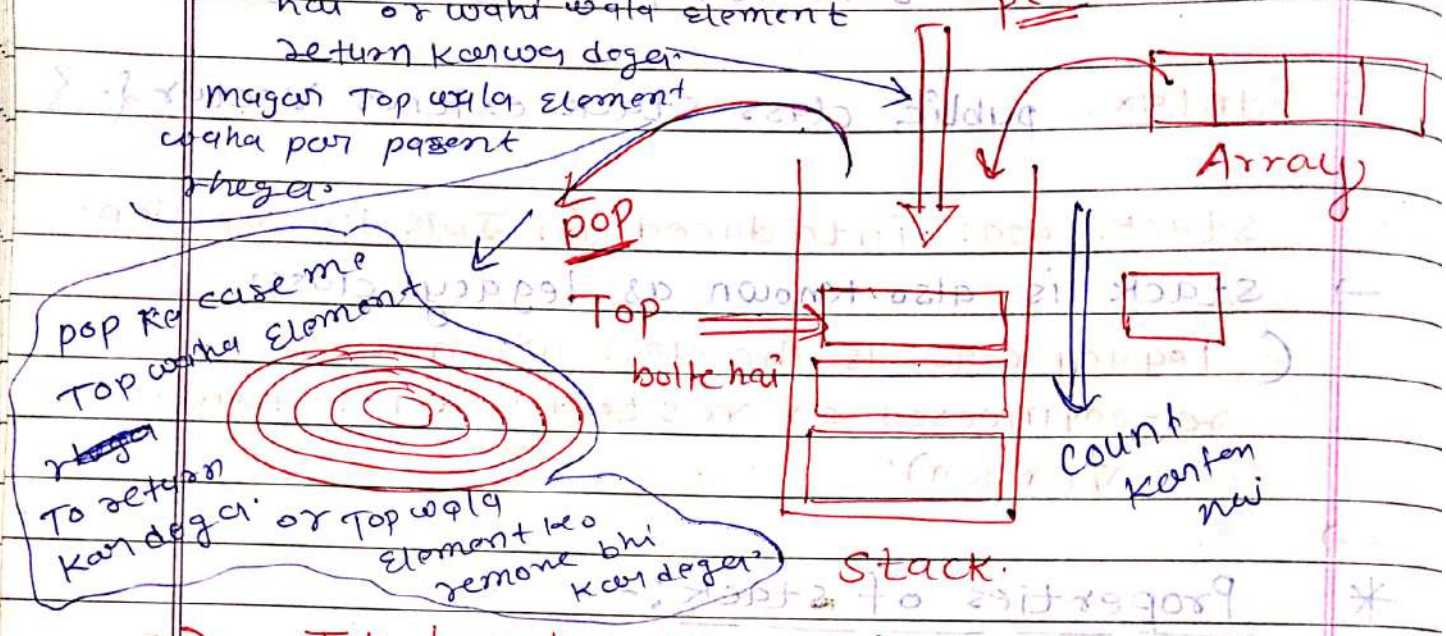
* Properties of stack:-

1. stack class was specially designed for Last In First Out (LIFO)
↓ we can change this algorithm according to our requirement.
↓ but ye wala algorithm ~~Amp~~ Apni requirement ke according change kar sakte hai

2. stack can be implemented using Array, ArrayList, LinkedList or Vector

3. stack is also index based data structure.

* Working for stacks



peek ke case me Top wala element dekhega kaun sa hai or wahi wala element return karwa dega magar Top wala element waha par present rhega.

→ Jab hum log Element ko stack ke Andher put kar wate hai to us wale part (ko) push bolte hai

→ Jab hum log Stack se Element ko retrive karwate hai ya phir get karwate hai to us wale part ko hum log kya bol dete hai pop

hum log stack ko upper se dekh rhe hai us part ko peek bolte hai

hum log upper se element ko peek karte hai

Last In FIRST OUT Ka matlab:

Jo bhi element bad me ~~ayega~~ hum log inserter karwayege wahi wala element sabse pahle retrive karwa sktehai

Last ~~In~~ FIRS OUT

Constructors:

1. public stack () {-}

Method:

1. Method of vector, List and collection.

2. public object push (object obj)
   ↓ Element insert karwa rhe hai

3. public Synchronized Object pop()
   ye ~~ho hai value~~ ko retrive karwa deta hai or value ko likal dega.

4. public synchronized object peek()

5. public synchronized int search (object obj)
   ↓ kon sa method search karna hai

6. public boolean empty()
   ye wala method check karega ki stack empty hai ya nhi

Cursors in Collection :-

→ Cursors :-

In Java, whenever we print the object reference, internally JVM will call toString() method of object class. In case of simple object it will print className @ reference but in case of prinq collection object it will print the element present in collection object.

↓

Arraylist me ata time saare ke saare element print karwa rha that but mujeh kya kam karna hai mmjeh ek sath Element ko print nhi karani mujeh individually print karwani hai tab humlog use karte hai Cursor ka

Agar hum log simple object ko print karna hai to output provide karta className @ reference.

→ When we print the collection object it will retrieve all the elements at one time but we want to retrieve the elements one by one then we have to use cursors.

* Types of Cursors :-

1. Enumeration
2. Iterator
3. ListIterator

⇒ 1. Enumeration :-

→ ① Enumeration is the cursor which is used to get the elements one by one from the collection object.

↓

Enumeration ek cursor hota hai Jo collection object se ek ek kar ke values ko print/get karwata hai

→ ③ Enumeration cursor is used only for legacy class.

↓

Enumeration cursor ko shrif legacy class ke liye hi use kiya gaha hai

② Enumeration was introduced in JDK 1.0 version.

* Steps "how to use" enumeration cursor :-

1. create Enumeration cursor object.

→ For this we have to use "public Enumeration elements ()" (this methods which is present in vector & stack Legacy class.)

2.) Read one by one all the elements from Enumeration Cursor.

→ public boolean hasMoreElements() }
→ public object nextElement () }

(~~these methods are present in~~ Enumeration interface)

→ Public class Test 2

{

public static void main (String [] args)

{

```
Vector v = new Vector();
v.addElement("aaa");
v.addElement("bbb");
v.addElement("ccc");
v.addElement("ddd");
```

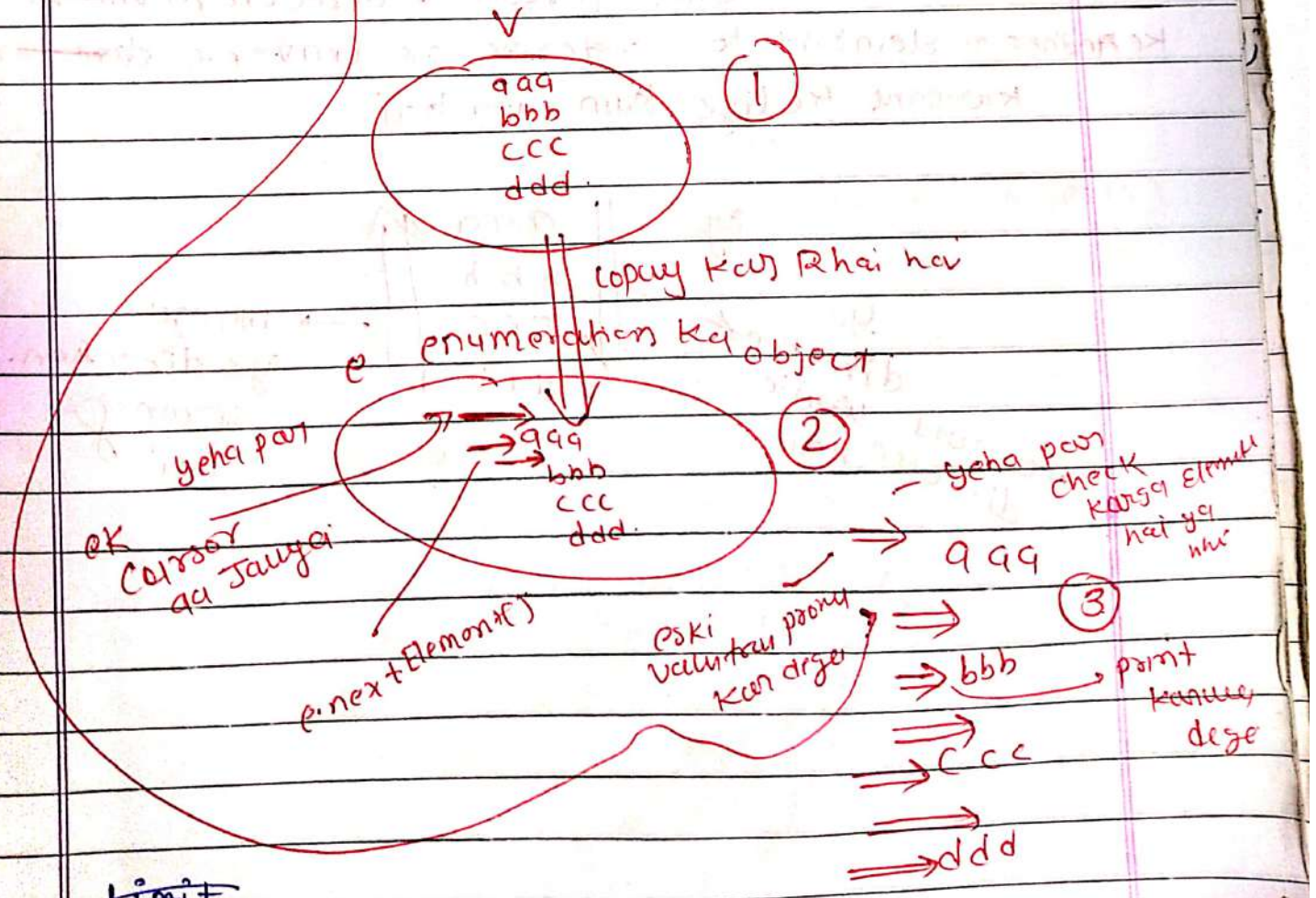
Step ①

```
Enumeration e = v.elements();
```

Step ②

```
While (e.hasMoreElements())
{
  S.y.pln (e.nextElement());
}
}
}
```

Step ③



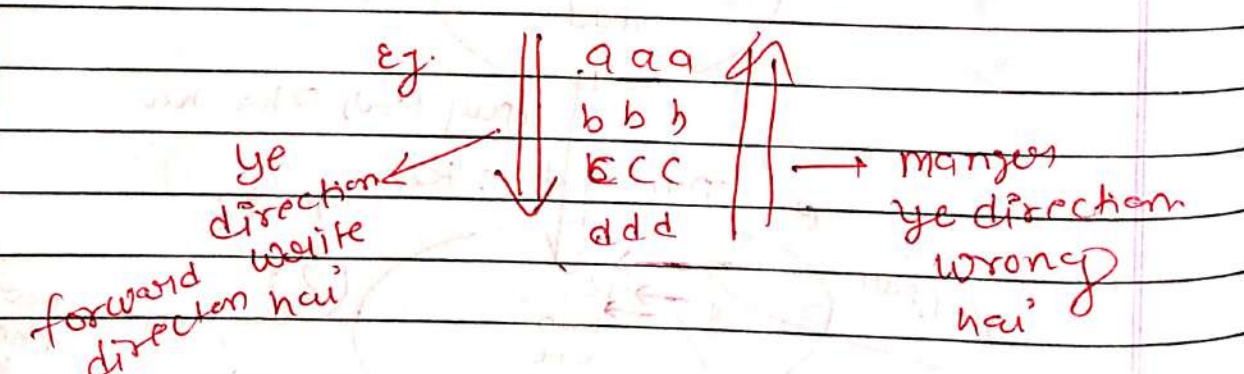
Limit

* Limitations of Enumeration:-

1. It can be used only with Legacy class and thus it is not universal cursor.
   ↓
   har ek ke sath use hote hai

2. By using enumeration cursor we can only perform read operation but not update or remove operation.
   ↓
   hum log eske through update and remove operation nhi kar skte hai

3. It can be used to traverse the elements only in forward direction.
   ↓
   ye sirf forward direction. ~~one hi~~ ke Another hi element ko retrive or traverse ~~direction~~ karwane ke liye kam ata hai



Iterator :-

* 1.) Iterator is a cursor which is used to get the elements one by one from the collection object.

⇓

values ko get karwane keliye kam aata hai

2.) It is universal cursor which means that we can use it with all collection objects.

3.) It can be used for read and remove operation.

4.) It was introduced in JDK 1.2 version.

* Steps "how to use" iterator cursor :-

1. create Iterator cursor object.
2. Read one by one all the elements from iterator cursor.

method → public Iterator iterator()
(this method is present in collection objects)

→ 1. public boolean hasNext()
2. public object next()
3. public void remove()
(these methods are present in Iterator interface)

remove karwana hai to

```
while(itr.hasNext())
{
    String element=(String)itr.next();

    if(element.equals("bbb"));
    {
        itr.remove();
    System.out.println("Element removed
                        successfully!!");
```

Limitations of Iterator Cursor :-

1. It can be used only for read and remove operation but not for replacement or addition operation.
2. It can be used to iterate the elements only in forward direction.

ListIterator :-

1. ListIterator is a cursor which is used to get the elements one by one from collection object.
2. ListIterator is bi-direction cursor which means it can be used to traverse the elements in forward or backword direction.

3. It can be used to read, remove, insert and replace operations.

4. It was introduced in JDK 1.2 version.

Steps "how to use" ListIterator cursor:-

1. create ListIterator cursor object :-
   → public ListIterator listIterator();
   (which is present in only List implementation classes)

2. Read one by one all the elements from ListIterator cursor :-
   → public boolean hasNext() } Elements ko forwards direction ke Andher traverse karwane ke liye)
   → public object next()
   → public int nextIndex()
   (above method are used to traverse the elements in forward direction)
   → public boolean hasPrevious() } Elements ko backword direction ke Andher traverse karwane ke liye.
   → public object previous()
   → public int previousIndex()
   (above method are used to traverse the elements in backword direction)
   → public void remove()
   → public void add(object obj)
   → public void set(object obj)

(above methods are used to remove, add and replace operation)

* Limitations of ListIterator cursor :-

1. It can be used only with List implemented classes thus it is not universal cursor.

* What is difference between Enumeration, Iterator & ListIterator.

Enumeration	Iterator	ListIterator
1. Enumeration can be used only with Legacy classes	1. Iterator can be used for any collection object.	1. ListIterator can be used only for List implemented classes
2. Enumeration can be used to traverse the elements only in forward direction.	2. Iterator can be used to traverse the elements only in forward direction.	3. ListIterator can be used to traverse the elements in forward and backword direction.
(3) Enumeration is used only for read operation.	(3) Iterator can be used for read and remove operation	(3) ListIterator can be used for read, remove, add and replace operations.
(4)		

Set (Interface) & Hashset

→ Set (Interface)

1. Set is an interface which is present in Java.util package.
2. Set is the child interface of Collection Interface.

* 3. Syntax: public interface Set extends Collection {- }

→ Set was introduced in JDK 1.2 version.

* → Hierarchy of set Interface.

* Properties of Set interface :-

1. Set is not an index based data structure, it stores the elements as per elements hashcode values.
   ↓
   ~~unique~~ uniqe Integer value hota hai Jo Jaise hi object create hota hai ushi time Heap manager create karta hai
2. Set does not follow the insertion order (except LinkedHashset) → LinkedHashset insertion order ko folloo kerta hai
3. Set does not follow the sorting order (except Sortedset, Navigableset, Treeset) ye thino sorting order ko folloo karte hai

4. Set can store different data types or hetrogeneous elements (Except sorteset, Navigableset, Treeset) ye thino different data types ko store nhi karwa sktehai → magar same data types ko store karwa sktehai

5. We cannot store duplicate elements in set.

6. We can store only one Null value in set

* Difference between List & Set :-

	"List"	"set"
1.	List is index based data structure which means in List data is stored by using index position.	1) Set is not an index based data structure, it stores the data according to hashcode value of the elements.
2.	List allows duplicate elements.	2. Set does not allow duplicate elements.
3.	List can store any number of null values.	3. set can store only one null value.
4.	List follows the insertion order.	4. set does not follows the insertion order.
5.	In case of list we can use Iterator & ListIterator cursor	5. In case of set we can use only Iterator cursor.
6.	List is used in case of retrieveing the elements.	6. set is used when we does not wan to allow duplicacy.

* Methods of set interface :-

⇒ Approximately same methods as that of collection interface

---

* Hashset :-

1. Hashset is an implemented class of set interface which is present in java.util package.

2. Syntax: public class Hashset extends Abstractset implements Set, Cloneable, Serializable {}

3. The underline data structure of Hashset is Hashtable (Hashset is backed up by map)

↓

Hashset ho jai map ke according, ya phir Hashtable ke according kam karta hai yehi hai underline data structure.

4. Hashset was introduced in JDK 1.2 version.

*

* Properties of Hashset :-

1. Hashset is not an index based data structure, it stores the elements according to elements hashcode values.

2. Hashset can store different data types of hetrogenous elements.

3. Hashset cannot store the duplicate elements.

4. Hashset can store maximum only one null value.

5. Hashset does not follows the insertion order

6. Hashset does not follow the sorting order.

(Same properties as Set interface)

7. Hashset is non-synchronized collection because Hashset does not contain any synchronized method.

8. Hashset allows more than one thread at one time.

↓

Hashset jo hai woh jaida thread ko ek sath allows kar deta hai execution ke liye.

↓

9. Hashset allows the parallel execution.

10. Hashset reduces the execution time which in turn makes our Application fast

11. Hashset is not threadsafe.

12. Hashset does not gyarantee for data consistencyy.

Working of Hashset:-

* map ke Andher values ko kisse type se store karwate hai. map ke Andher hamesha key value pair ke form me data store karwate hai

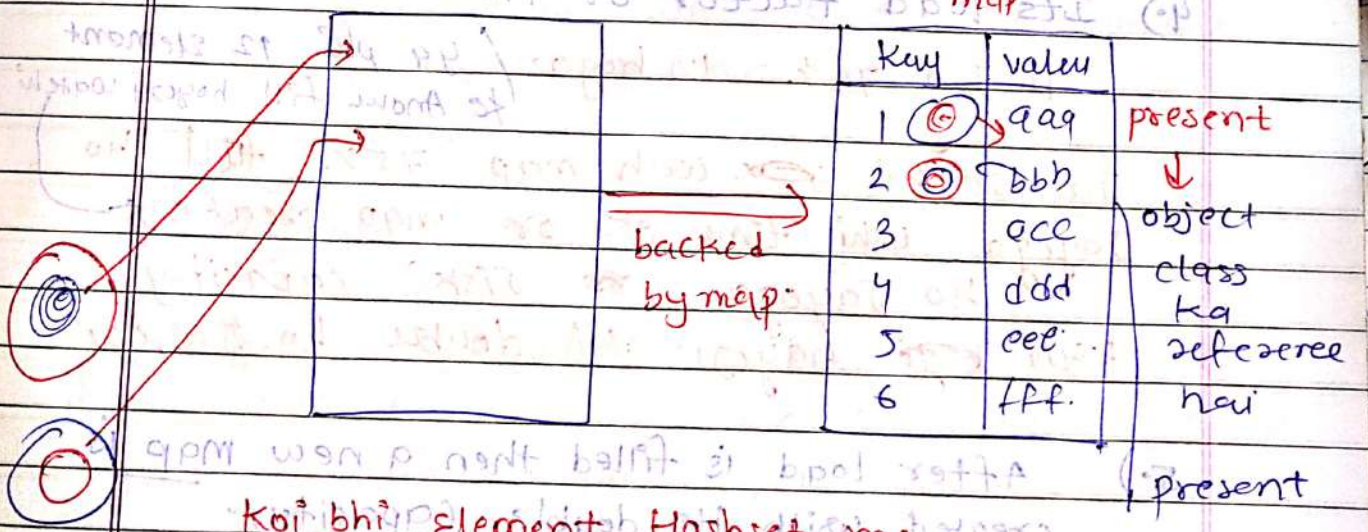
Map concept:

Key	Values
1	aaa
2	bbb
3	ccc
4	ddd
5	eee

Key is unique value.

Value me duplicate value store ho sakta hai

① Hashset internally works on the basiss of Hashtable (it internally backed up by map)



Koi bhi Element Hashset me store karte hai wah Map me jakar store hota hai jai ki waha Hashset me store hua. or map me key ke Andher store hokar value me present ho jata hai present ki object class ke reference hai

2. When we insert any element in Hashset it stores as a key inside the map and at value position PRESENT reference variable is stored which is the reference variable of object class.

3. Initial capacity of Hashset is 16 Element. (map ke bhi initial capacity 16 hoti hai)
16 maximum element store in Hashset

↓

* Hashset ke 16 capacity kyu hoti hai
map ke inital capacity 16 hoti hai
is esliy by default hashset ke bhi capacity 16 =

4.) Its load factor or fill ratio is 75%.
16 ka 75% 12 hoga. / ya phi 12 element ke Andar fill hoyega waise hi

↓

Jai se hi ~~75%~~ wah map 75% full ho Jayega ushi time ek or map create ho Jayega. Jiski capacity kitni ~~hogi~~ hayegi eski double ho Jayegi

5.) After load is filled then a new map is created with its double capacity.

* Constructors of Hashset:-

1. public Hashset();
2. public Hashset(Collection<? extends E> c)
   (object obj)
3. public Hashset(int intialCapacity, float loadFactor) → (10, 0.75f)
   ↓ 75 full ho thi kahi
   1 → 100%
   0.80 → 80%.
4. public Hashset(int initialCapacity).
   ↓
   initialCapacity provide kar skte hai.

* Methods of HashSet:-

= approx same methods as that of Collection or Set interface.

* Programs

* When We should use Hashset ?

Ans Hashset jo hai search operations ke liye bahut achha hota hai

Jobhi Data structure Hashcode value ke bases par kam karta hai na us case me searching operation bahut jaida achha hota hai

→ Hashset is good for searching or retriving operation.

* LinkedHashset (class):

→ LinkedHashset is the child class of Hashset which is present in Java.util package.

→

Syntax: public class LinkedHashset extends Hashset implements set, Cloneable, serializable {-}

→ The underline data structure of LinkedHashset is "Hashtable + LinkedList"

→ LinkedHashset was introduced in JDK 1.4 version. JSE. (1.4)

* Properties of LinkedHashset :-

1. All properties of LinkedHashset is same as Hashset except LinkedHashset follows the insertion order.

(same insertion order follow karta hai / or SAb same property hai)

→ Constructors of LinkedHashset :-

1. public LinkedHashset ()
2. public LinkedHashset (object obj)
3. public LinkedHashset (int initial capacity)
4. public LinkedHashset (int initialCapacity, float loadFactor);

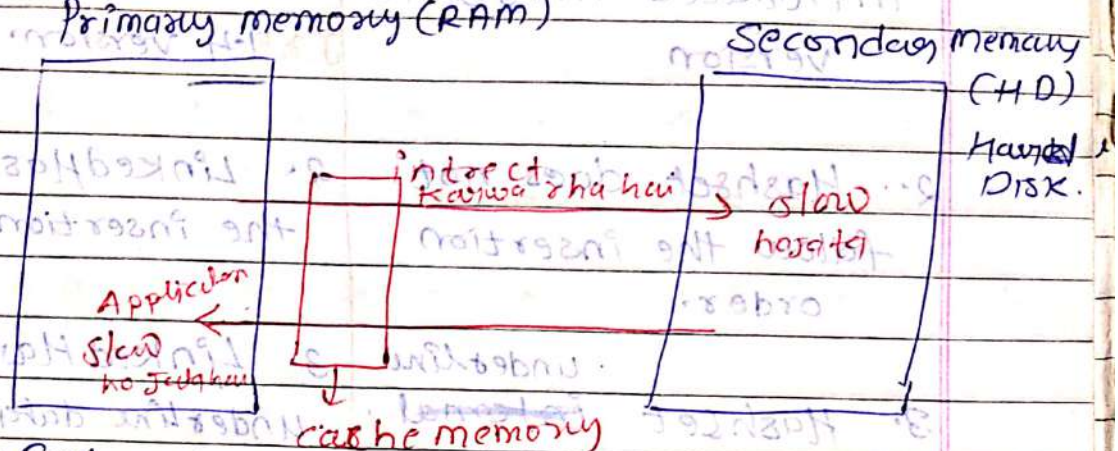
(Same Constructors as Hashset)

→ Methods of LinkedHashset:-

⇒ Same methods as Hashset.

* When we should use LinkedHashset :-

Ans



Property → Fast.
→ Costly

→ ~~We cannot~~ store duplicate elements.

→ It follows the insertion order.

1) If we have to create cache based applications then we can use LinkedHashset.

↓

LinkedList Jo hai cache based Application ke liye use hota hai

Difference between Hashset & LinkedHashset:-

⇒

Hashset	LinkedHashset
1. Hashset was introduced in JDK 1.2 version.	1. LinkedHashset was introduced in ~~JDK~~ J2SE 1.4 version.
2. Hashset does not follow the insertion order.	2. LinkedHashset follows the insertion order.
3. Hashset ~~internal~~ underline data structure is Hashtable.	3. LinkedHashset underline data structure is "Hashtable + LinkedList".

* Sortedset :-

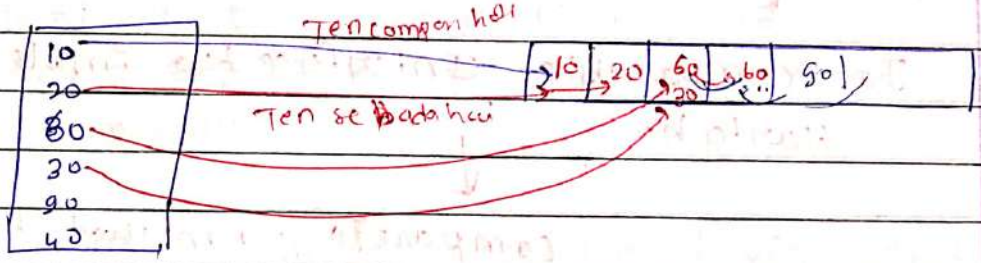
→ Sortedset is a child interface of set interface which is present in Java.util package.

→ Syntax: public interface Sortedset extends set{..}.

→ Sortedset was introduced in JDK 1.2 version.

* Properties of Sortedset :-

1. Sortedset is not an index based data structure.
2. Sortedset does not follows the insertion order.
3. Sortedset follow the sorting order.



When 2 objects are compared then three type of value are return:

0

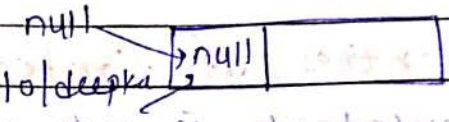
+1

-1

4. Sortedset can store same data types or homogeneous elements. (If we provide different data type element it will provide exception. i.e Java.lang.ClassCast Exception.

5. Sortedset cannot store the dyplicate elements

6. We should not store null value in SortedSet because ~~while~~ SortedSet follows the sorting order so while comparing the elements with null value, it will provides the Java.lang.NullPointerException.



7. SortedSet allows comparable objects.

All primitive data type jab comparable Interface ko inherit karte hai

To comparable Interface ko implements karta hai

↓

compareTo → method hota hai

by default, but if we insert non-comparable objects, then it will provide an exception.

8. SortedSet is non-synchronized collection.

bbb ddd
eske bich ke element or bbb se

* Method of Sortedset:

1. Sortedset<E> subset(E fromElement, E toElement);
2. Sortedset<E> headset(E toElement);
3. Sortedset<E> tailset(object fromElement);
4. object first();
5. object Last();
6.

Navigableset:-

→ Navigableset is the child interface of sortedset interface which is present in Java.util package.

* Syntax: public interface Navigableset extends Sortedset {-}

→ Navigableset was introduced in Java SE 6 version

* Properties of Navigable set:-

⇒ Navigableset has the same properties as that of Sortedset. but it provides some extra navigable methods.

* Methods

1. public NavigableSet descendingSet()
2. public object ceiling(object obj)
3. public object higher(object obj)
4. public object lower(object obj)
5. public object floor(object obj)
6. public object pollFirst()
7. public object pollLast()

TreeSet (class)

Date
Page

Treeset is the direct implementation for Navigableset interface. but it also provides the implementation for sortedset, set & Collection interface.

Syntax :- public class Treeset extends Abstractset implements Navigableset, Cloneable, Serializable {-}.

The Underline data structure of Treeset is "Balanced Tree"

Tree set was introduced in JDK 1.2 version

Properties of Treeset :-

1. Treeset is not index based data structure.
2. Treeset does not follows the insertion order
3. Treeset follows the sorting order. it may be default sorting order or customized sorting order.

↓

~~wah ho sakta hai ki default sorting ho ya phir hame Customized khud se Customized kiya ho.~~

Integers

→ 10, 20, 30, 40, 50

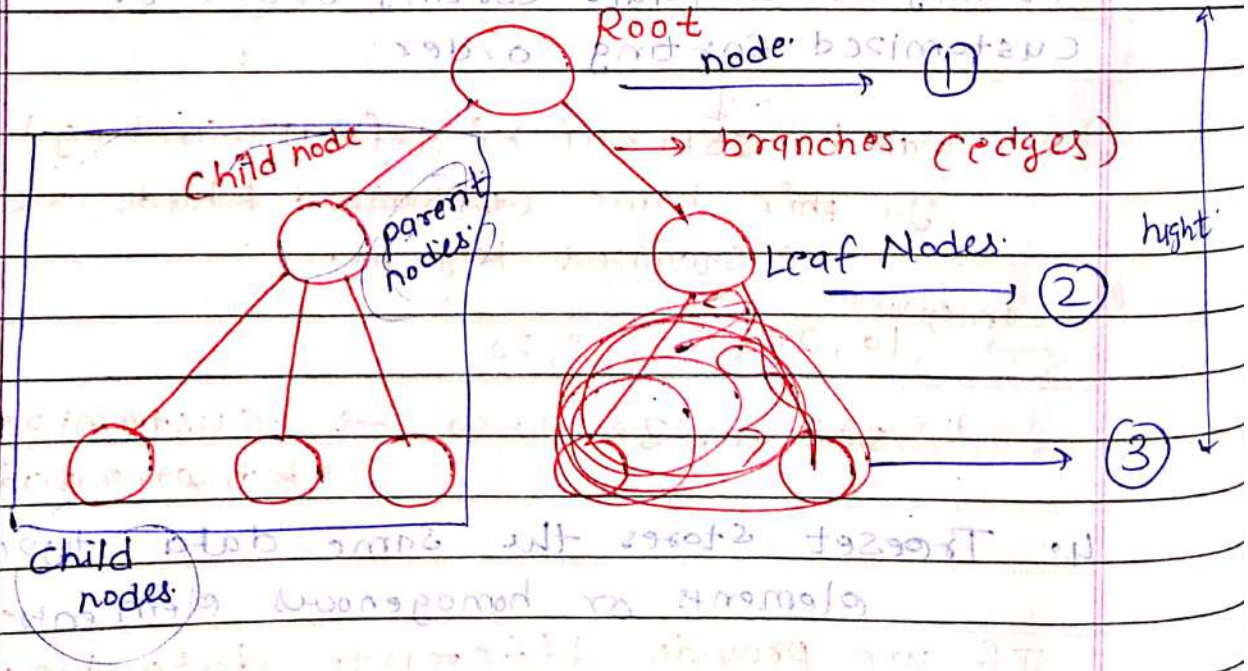
~~50, 40, 30, 20, 10~~ → Customized karwana padta hai

4. Treeset stores the same data type elements or homogenous elements.

If we provide different data types elements in Treeset then it will provide Java.lang.ClassCastException.

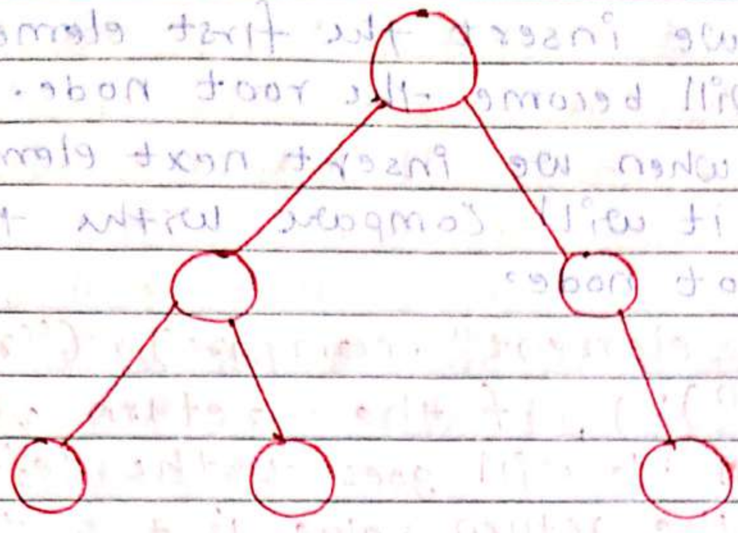
5. Treeset cannot store the duplicate elements
6. Treeset can store only one null value.
↓ wah bhi first me
7. Treeset is non-synchronized collection.
↓
~~kyu hota hai kyu ki Treeset ke Another koi bhi method Synchronized nhi hota hai~~
8. Treeset allows more than one thread at one time.
9. Treeset allaos the parallel execution.
10. Treeset reduces the execution time which makes our application fast.
11. Treeset is not threadsafe.
12. Treeset does not gurantee for data consistency.

* Working of Treeset:-

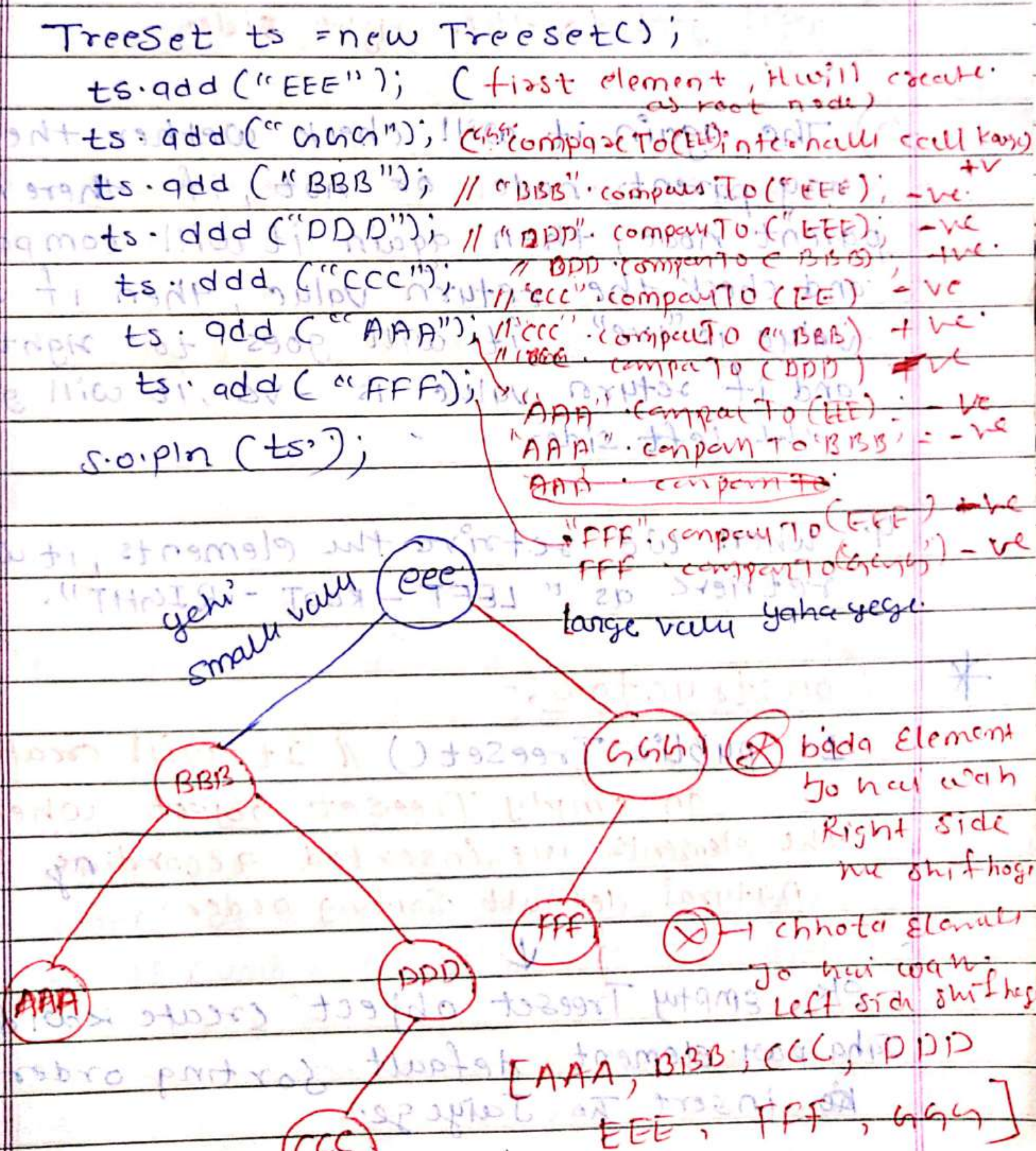


Treeset:

blanced Tree:



```
TreeSet ts = new Treeset();
ts.add("EEE");    (first element, it will create as root node)
ts.add("GGG");    ("GGG".compareTo("EEE") internally call karega)
ts.add("BBB");    // "BBB".compareTo("EEE"); -ve
ts.add("DDD");    // "DDD".compareTo("EEE"); -ve
                  // DDD .compareTo (BBB); +ve
ts.add("CCC");    // "CCC".compareTo (EEE) -ve
ts.add("AAA");    // "CCC".compareTo ("BBB) +ve
                  // "CCC" compareTo (DDD) -ve
ts.add("FFF");    "AAA".compareTo (EEE) - ve
                  "AAA".compareTo "BBB" - ve
                  ~~AAA . compareTo~~
                  "FFF".compareTo (EEE) +ve
                  FFF compareTo("GGG") - ve
s.o.pln (ts);
```



[AAA, BBB, CCC, DDD
EEE, FFF, GGG]

1. When we insert the first element, it will become the root node.
2. Then when we insert next element, it will compare with the root node.
("next element".compareTo("root element"));, if the return value is -ve, then it will goes to the left side and if the return value is +ve then it will goes to the right side.

3) The again it will check wether there is any parent node or not, if there is any parent node, then again it will compare and check the return value, then if return value is "+ve", it will goes to right side and if return value is -ve, it will goes to the left side.

4. When we retrive the elements, it will retrieve as "LEFT - ROOT - RIGHT".

* Constructors:-

1. public TreeSet() // It will create an Empty Treeset object where the elements are inserted according to natural default sorting order.

↓

ek empty Treeset object create kardega. Jaha par element default sorting order me insert ho jayege.

2. public Treeset (comparator comparator)
// it will create an empty Treeset object where the elements will be inserted according to the customized sorting order.
⇓
Agar much mera hi sorting order create karna hai to mai eska use karuga.

3. public Treeset (collection c)//
we can pass any other collection objects.
⇓
ya pr koi bhi collection objects pass karwa skte hai

4.) public Treeset (sortedset s)
⇓
sortedset ka koi bhi object pass karwa skte hai

Methods of Treeset:

1. Contains all the methods of collection, set, sortedset & Navigableset interface.

When we should use Treeset:-
when we want to store the ~~elements~~ large number of elements in sorting order.
⇓ bauhut sare element ko sorted order me store karwana ~~hai~~ to chahate hai to eska use kar skte hai and due to this retrival operation is fast

(14) Cases for "null" values insertion in Treeset:-

1. We can store only one null value.
2. We can insert null value only at First position: but if we insert the null value at second position then it will provide an Exception i-e Java.lang.NullPointerException'.

any other element then it will provide "Java.lang.NullPointerException"

3. NOTE: Until 1.6 version we can successfully insert the null value at First position but after 1-6 version we cannot store the null value even at First position.

1.6 version ke bad Null value use nhi kar skte.

kyu : baki ke element insert nhi hoge.

What is difference between Hashset, LinkedHashset & Treeset :-

Hashset	LinkedHashset	Treeset
1. Hashset underline data structure is "Hashtable".	1. LinkedHashset underline data structure is "Hashtable + LinkedList".	1. Treeset underline data structure is "Balanced Tree".
2. Hashset does not allow insertion order.	2. LinkedHashset allow the insertion order.	2. Treeset does not allow the insertion order.
3. Hashset does not allow the sorting order.	3) LinkedHashset does not allow sorting order.	3) Treeset allows the sorting order.
4.) Hashset allows the heterogenous objects.	4.) LinkedHashset allows the hetrogenous objects.	4.) Treeset does not allow the hetrogeneous objects.
5.) Hashset allows the null value insertion.	5) LinkedHashset allow the null value insertion.	5) Treeset allows the null value but at first position. (but this is applicable till Java 1.6 version).

Comparable & Comparator (Interfaces)

Comparable:

→ 1. Comparable is an interface which is present in Java.lang package.

→ 2. It contains only one method i.e. compareTo().

* Prototype: public int compareTo(Object obj)

obj1.compareTo(obj2);

ye three values return karta hai

+ve — if obj1 is greater than obj2

-ve — if obj2 is greater than obj1

0 — if obj1 is Equal to obj2.

→ Jitne bhi wrapper class hai and including String wah kya kam karte hai Comparable Interface ko inherit karte hai and compareTo method ko implement kiya hoga.

→ 3. String and all wrapper classes (Integer, Long, Float etc.) implements Comparable interface.

→ 4. Problems with Comparable Interface.

1. By implementing Comparable interface, the properties of original class will get changed.

2. By this way we can sort only for one entity for Example we can sort the student object either with name or rollno at one time.

* Comparator:-

→ 1. Comparator is an interface which is present in Java.util package.

2. Comparator interface contains 2 methods.

1. public int compare(object obj1, object ob2)
   ↓
   dono object ko yaha provides karwate hai

2. public boolean Equals (object obj)

* WAP to compare and add different elements according to length and alphabetical order.

eg. A, B, AA, BB, AAA, ~~AAAA~~ BBBB, AAAAA BBB

{ Customize karwana hai usliye comprator ka use karge. }

*

* What is difference between Comparable ~~Comparator interface~~ :

Comparable	Comparator
1) Comparable interface contains compareTo() method.	1) comparator interface contains compare() method.
2) Comparable interface is present in java.lang. package.	2) Comparator interface is present in java.util package.
3) By using comparable interface original class properties will get changed.	3) By using comparator interface original class properties will not be changed.
4) Comparable interface can sort only one entity at one time.	4) Comparator interface can sort multiple entities ~~at one time~~.
5) Comparable interface is used to implicit sorting	5) Comparator interface is used for customized sorting. ↙ Explicit.

Queue (Interface); PriorityQueue (class) & Deque & ArrayDeque.

* Queue

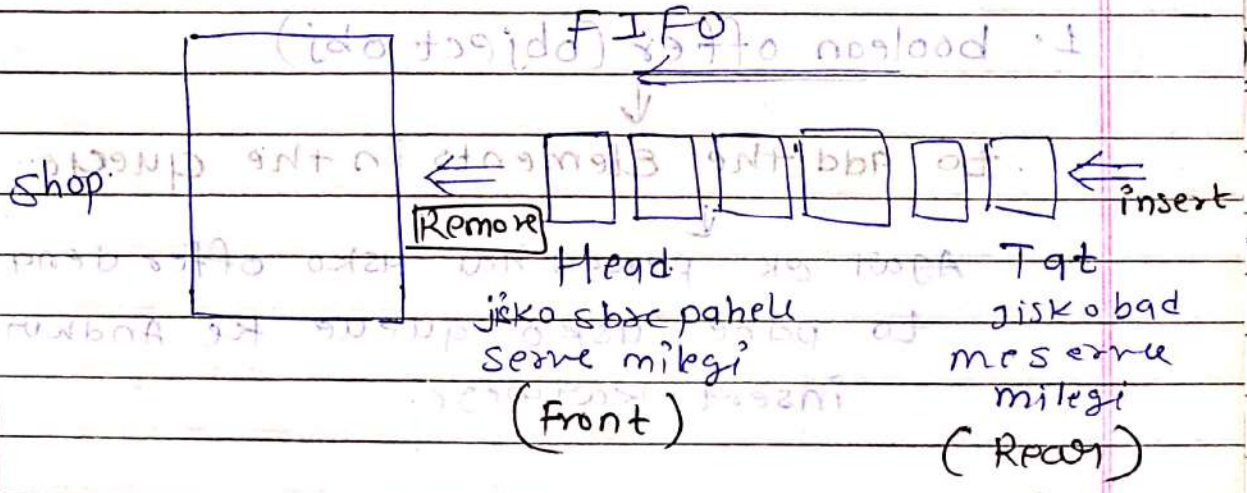
1. Queue is the child interface of Collection Interface.

* Syntax:- public interface Queue extends Collection {-}

2. Queue was introduced in JDK 1.5 version

3. Queue orders the elements in FIFO (First In First Out) manner., but we can change this algorithm according to our requirements.
↓
ye wala algorithm ko humlog change kar sktehai

4. Implementation of Queue:-



* Hierarchy of Queue:

* Properties of Queue:-

1. Queue does not follow the insertion orders.
2. Queue follows the sorting order.
3. Queue stores the same type elements or homogeneous elements. If we try to store different elements then it will throw an exception saying Java.lang.ClassCastExpho
4. Queue can store the duplicate elements.
5. Queue does not store any null value. If we try to store null value then it will throw an exception i.e. "Java!lang.NullPoint Exception.

* Methods of Queue:- method ki body nhi hai kyu ki interface

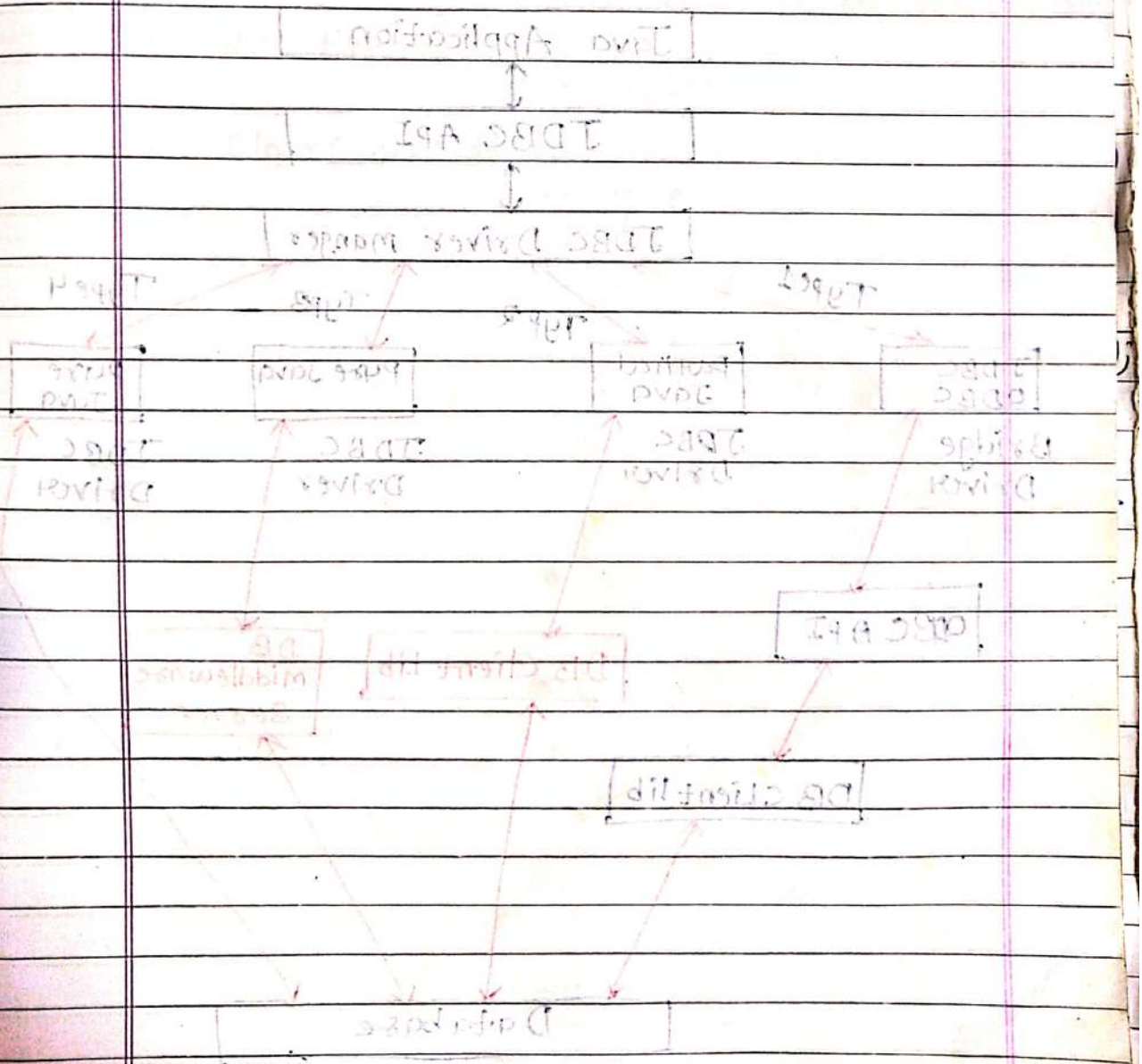
1. boolean offer(object obj)
   ↓
   to add the elements in the queue.
   ↓
   Agar ek person hai usko offer dena hai to pahe usko queue ke Andhar insert karwese.

2) object peek() → it will return the head element of the queue. If no Element is found in the queue it will return null value
3) object element() - It will return the head element of the queue.

If no element is found, it will throw an exception i.e "NosuchElementException"

4. object poll() → It is used to remove the head element and also it will return that element. If no element is found, then it will return null value.

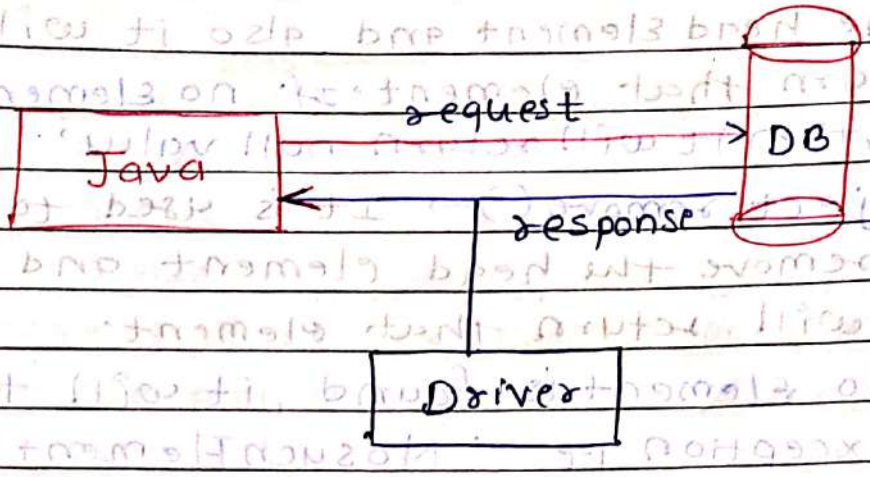
5. object remove() → It is used to remove the head element and also it will return that element.
If no element is found, it will throw an exception i.e "NoSuchElementException" (Java.util.)

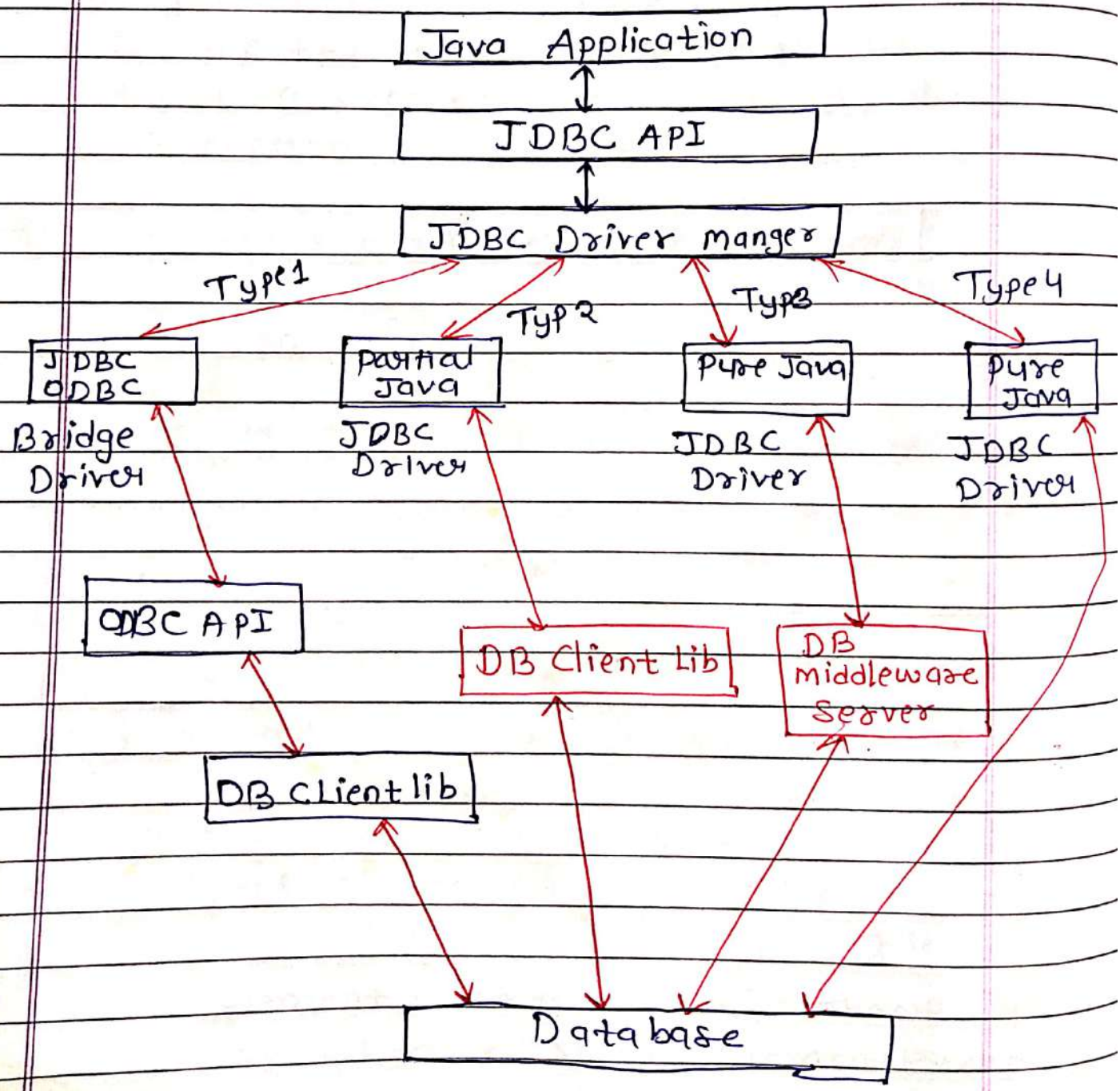


28/07/24

JDBC:

Java Database CONNECTIVITY:





7 Step to Connect With DB

1. import package :- Java.sql.*

2. Load & Register Driver
Class.forName("com.mysql.jdbc.Driver");

3. Establish Connection / create Connection.
Connection con = DriverManger.getConnection(url, user, password);
url="jdbc:mysql://localhost:3306/dbname"
mysql

4. Create Statement
Statement stmt = con.createstatement();

5. Execute Query
exceuteQuery()/exceuteUpdate()

6. Resultset
Resultset

7. CloseConnection.
con.close();

29/07/21

Date
Page

Data insert Kawana hai

available

Static — Statement(I) → compile atomic compile ho jata hai; static data ko allow karta hai, dynamic data ko nhi

'Single data insert kar skte hai'

static + dynamic — Preparedstatement(I) — precomplied ✓ — better hai user karne mei

multiple data insert kar skte hai

esme static + dynamic dono use kar skate.

user se data dena.

static + dynamic — Callablestatement(I)

function + stored procedure.

→ Connection ko bahar ~~insert~~ declear kar ~~resp~~ result dege.

prepared statement sab push.

or try catch me usko use karge or finally me usko closed kar dege.

(I) try catch local ~~variable~~ variable hai usme. finall use ~~kar~~ nhi kar sku

usko

02/09/2021

project: onlineBookStore

Book customer cart order

package

pojo (plain old java object)
private data members

Capsulation use karenge

getter setter
cons- non-param, param
toString()

package

dao (data access object)
Interface add, delete method ye sab yaha likhege.
class

package

Test
class logical part likhege. dao ke sath connection karenge.

utility
DBConnection → class create karenge. connect to Javacode

→ com.bookstore.pojo → package.

Book (class)

private int bookId;
String bookName;
String authorName;
String publisherName;
double rating;
double no-of-copies;
bookPrice;

* Book (table) (→ DBse me lege)

* BookDao (I) (Abstract method - lege)

boolean addBook() → All information add karna hai to (Book b); (Flamesh method lege)
boolean updateBook() → &
boolean deleteBook() → (int bookId)
List<Book> showAllBook(); → Jaha ko parameter nhi hoten hai
Book ~~searchBook()~~ → (int bookId);
Book searchBookById()
List<book> searchBookByName() → (String bookName);

* BookDaoImpl(C)

* BookTest(C)

Customer (pojo)

```
int custId;
String custName;
string custEmail;
string custPassword;
string custAddress;
```

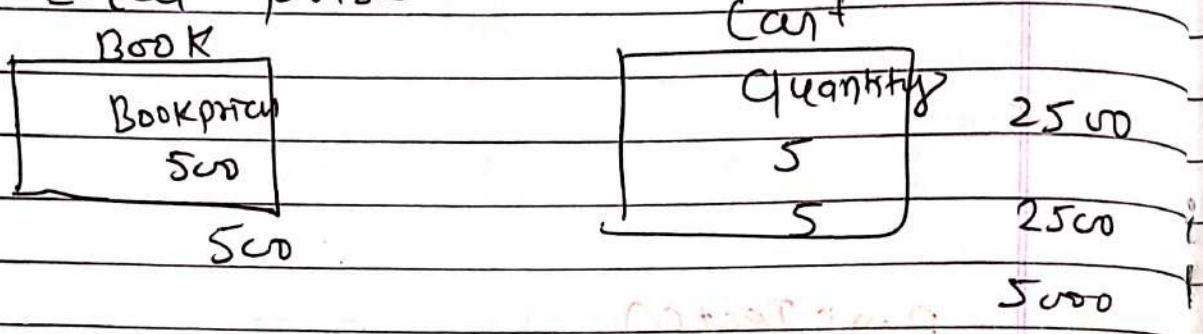
Customer Dao

```
boolean addCustomer(Customer c);
boolean updateCustomer(Customer c);
boolean deleteCustomer(int custId);
List<Customer> showAllCustomers();
Customer searchCustomerById(int custId);
List<Customer> searchCustomerByName(
        String custName);
```

Customer search (customerByEmail (String custEmail)',

order;

total price:



Select * from book b inner join cart ca on b.bookid = ca.bookid where custemail = ?;

JDK : 8.

1. StringJoiner
   do string ko join karne ke kam karta hai
2. Default Method & static method. → use same same line of code multiple classes.
   ek line ko multiple class me use kar skte hai
3. LocalDate date = LocalDate.now();
   System.out.println(date); pc me current Date batayega.

// off
LocalDate date1 = LocalDate.of(2012, 11, 23);
s.o.println(date1);

// LocalDate date2 = LocalDate.of(2010, Month.JANUARY, 17);

// Zone Area.
LocalDate date3 = LocalDate.now(ZoneId.of("Asia/Kolkata"));
s.o.p(Date3);

// Pure code me kar sal Day change

LocalDate, date4 = LocalDate.ofYearDay(2019, 5); (50), (35)
s.o.p(Date4);

// ofepochday by default date---
01/01/1970
LocalDate date5 = LocalDate.ofEpochDay(5);

// ofyearofDay ofsecondofDay

X Lambda Expression:

```
@FunctionalInterface
interface Vehicle {
    void run();
}

class Myclass {                    // imp vehicle
    main() {
    }
}

Vehicle v = new Vehicle() {
    void run() {
    }
};
```

multiplu interfaces in singer class

yes kar sakte hai