Collection

Q. What is a Collection?

Collection is a ready made implementation of data structure

Q.What is the purpose of Collection?

---

1. Ability to store any kind of data.
2. We can extends its size at run time as well as shrink its size at run time
   Means we can store infinite data in collection
3. Provide ready made implementation of data structure

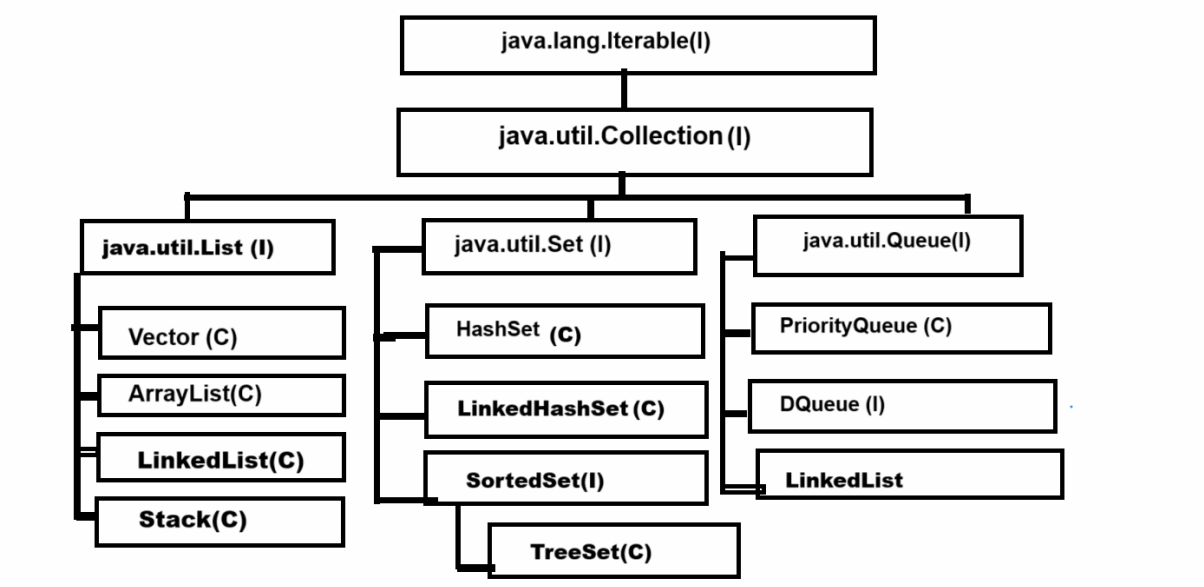
Note: if we think about array then we cannot modify its size at run time Or we need to write manual logic to extend the array size.
When we use array we need to manual logic of data structure like as sorting ,searching etc

Note: when we create array of Object class then we can store any kind of data in array

```
package org.techhub;
public class TestcollApplication {
	public static void main(String[] args) {
		Object obj [] = new Object[5];
		obj[0]=100;
		obj[1]=false;
		obj[2]=5.4f;
		obj[3]=new java.util.Date();
		obj[4]="good";
		for(int i=0; i<obj.length;i++) {
			System.out.println(obj[i]);
		}
	}
}
```

If we think about above code we have collection of different types of data using Object class array but there is limitation when we want to any data structure operation on above code we need to write manually like as searching, insertion of element , deletion of element etc
So Java Suggest we provide ready implementation of data structure to developer known as collection

If we want to work with Collection in JAVA we need to know the following classes and interface hierarchy



If we think about above diagram we have top most interface name as java.lang.Iterable

Q. Why is java.lang.Iterable the parent of all Collection?

The major goal of Iterable interface is to fetch data from collection and data fetching is common operation in all collections so Iterable is parent of all collection because Iterable interface contain iterator() method which return reference of Iterator interface and using Iterator interface we can fetch data from collection

```
interface Iterable
{  Iterator  iterator();
}
```

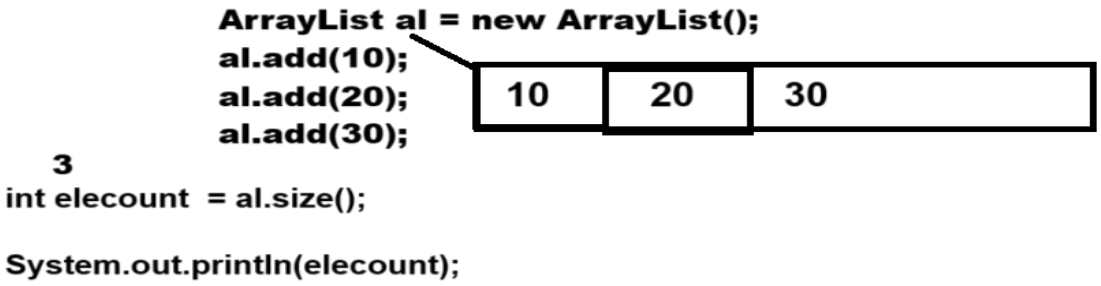
iterator() method return reference of Iterator interface and Iterator interface contain some inbuilt methods which is used for fetch data from collection like as

boolean hasNext() : this method check data present in collection or not if present return true otherwise return false
Object next() : this method can fetch data from collection and move cursor on next element
void remove() : this method can remove data from collection using iterator

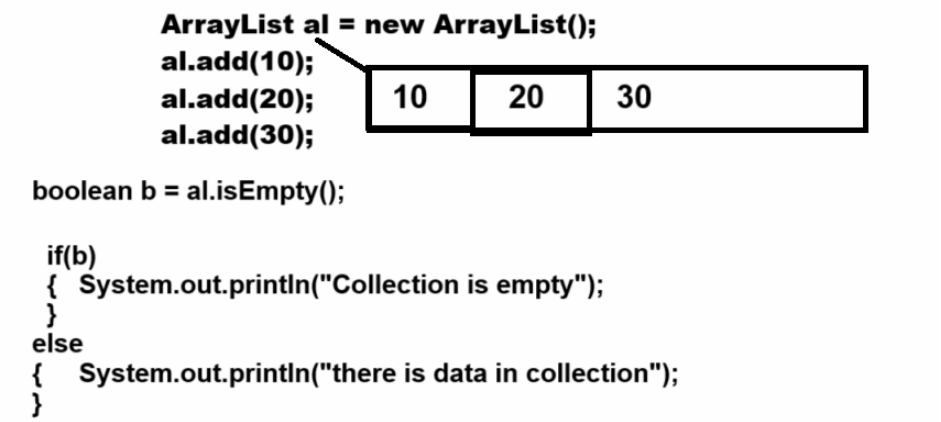
Collection interface

Collection interface contain common methods which is required for perform daily operation on collection like as adding element, removing element ,checking size etc

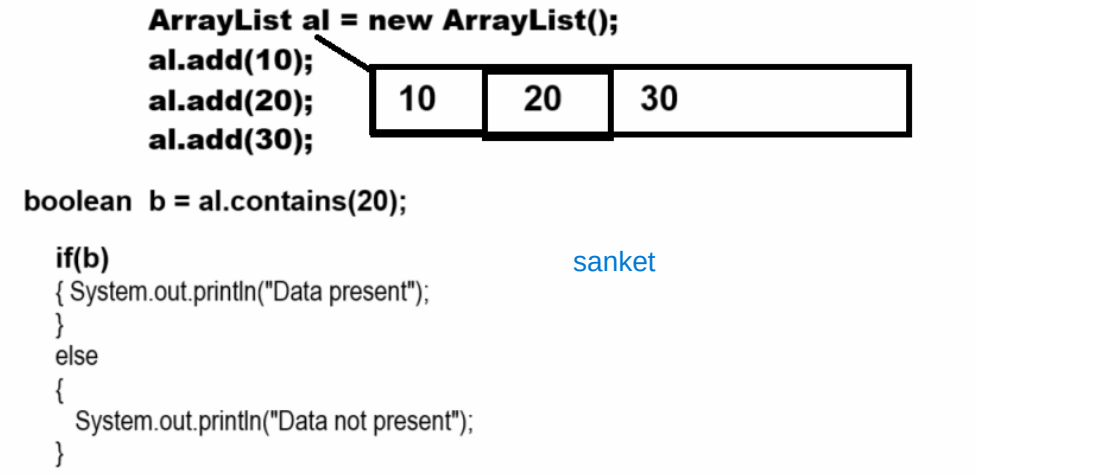
public abstract int size(): this method is used for return size of collection or return number of element present in collection



public abstract boolean isEmpty(): this method checks if the collection is empty or not if empty return true otherwise return false.



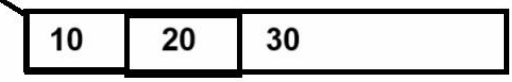
public abstract boolean contains(java.lang.Object): this method is used for checking data present in collection or not means normally we use this method for data searching purpose.



sanket

public abstract java.util.Iterator<E> iterator(): it helps us to fetch data from collection and it is a universal iterator.

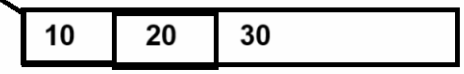
```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```




```
Iterator   i = al.iterator();

while (  i.hasNext() )
{
    Object   obj  = i.next();

  System.out.println(obj);

}
```

public abstract java.lang.Object[] toArray(): this method can convert any collection in object class array.

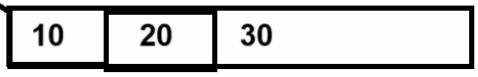
```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```




```
Object  obj []  = al.toArray();
for(int i=0; i<obj.length; i++)
{
  System.out.println(obj[i]);
}
```

public abstract boolean add(E) : this method can add a new element in collection and return true if element added otherwise return false.

```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```

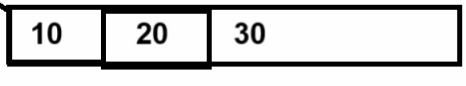



```
boolean  b = al.add(100);

if(b)
{
  System.out.println("element added ");
}
else
{
  System.out.println("Element not added");
}
```

public abstract boolean remove(java.lang.Object): remove element from collection and element removed successfully return true otherwise return false.

public abstract boolean containsAll(java.util.Collection<?>):

```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```




```
Collection c = new LinkedList();
c.add(10);
c.add(500);

boolean b= al.containsAll(c);
if(b)
{
  System.out.println("Element found");
}
else{
  System.out.println("element not found");

}
```

public abstract boolean addAll(java.util.Collection<? extends E>): this method helps us to add more than one element in collection at time.
public abstract boolean removeAll(java.util.Collection<?>): this method also helps us to remove more than one element at a time from collection.

There are three types of collection we have

List : List collection can store duplicate values or data and manage data by using indexing technique means we can say list collection is implementation of linear data structure internally.

Set : Set collection cannot allow duplicated data as well as can generate random data also provide data in sorted format using TreeSet collection etc
Means set collection is implementation of non linear and hash based data structure

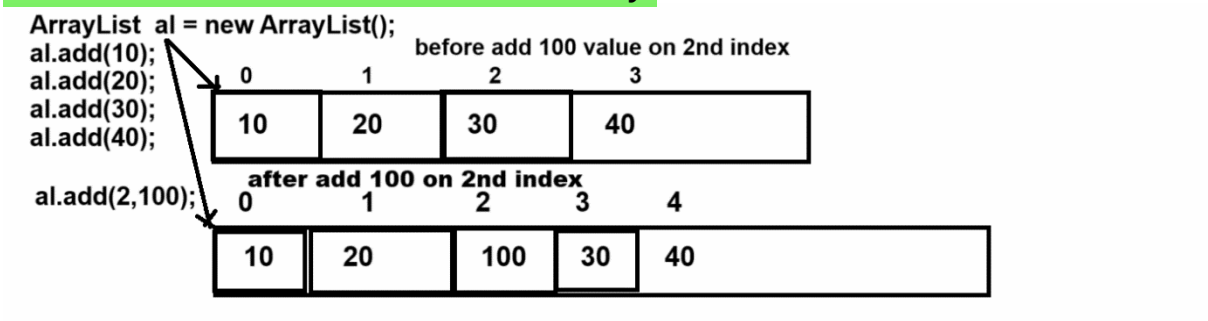
Queue : Queue in data structure allow to store duplicate values and can manage data using first in first out format and provide indexing to data management etc

Now we want to discuss about the List Collection

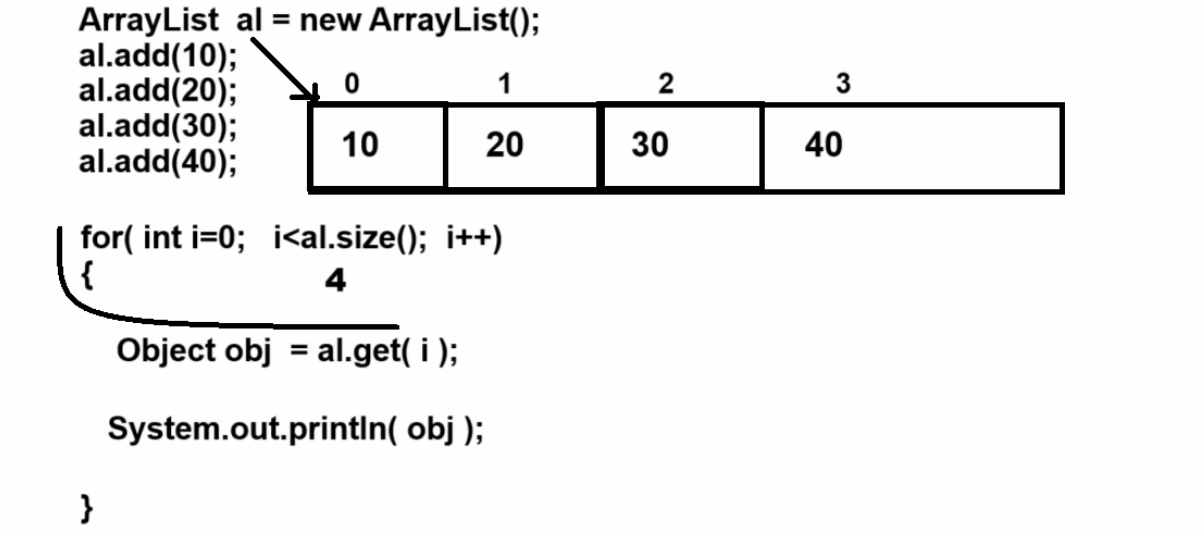
Methods of List Collection

List Collection contains all methods from Collection interface as well as List contains its own additional method also.

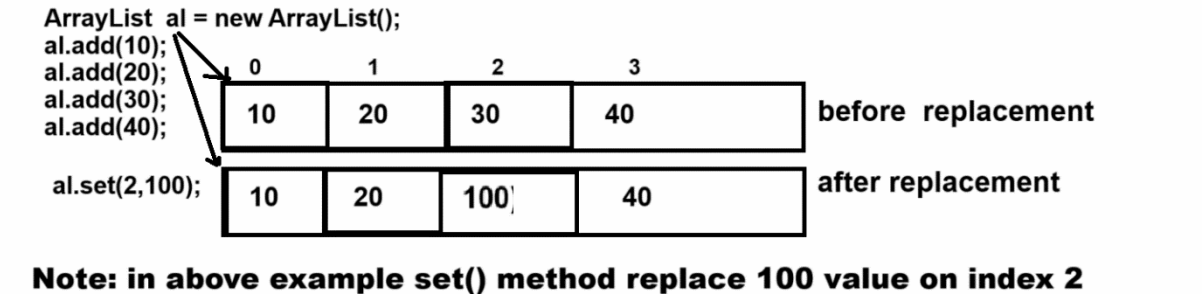
public abstract void add(int, E): this method is used for adding a new element on a specified index and moving previous values from that index onto the next index automatically.



public abstract E get(int): this method can return data from ArrayList collection using its index

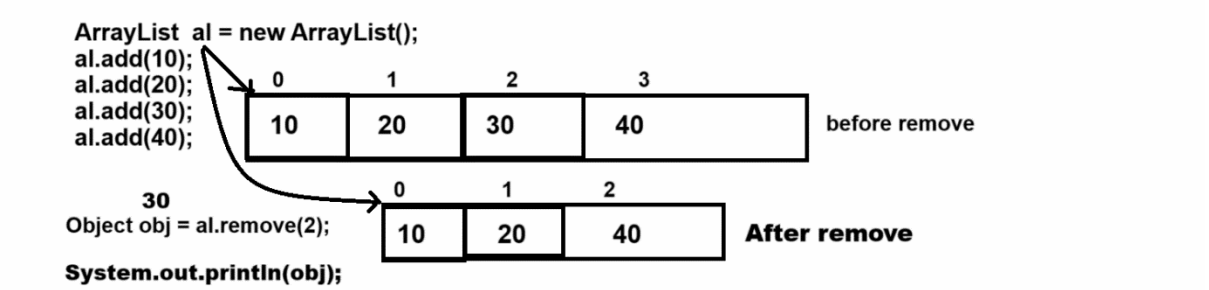


public abstract E set(int, E): this method is used to replace data on specified index.



Note: in above example set() method replace 100 value on index 2

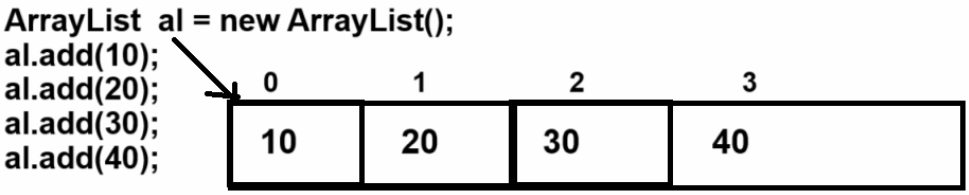
public abstract E remove(int): this method is used for remove data using its index and return removed value as output



public abstract int indexOf(java.lang.Object): this method can return index of particular element if element or value not found in collection return -1

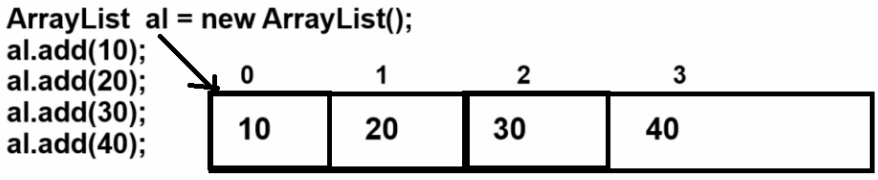
Note: this method can use for two purpose

a. Searching element from collection



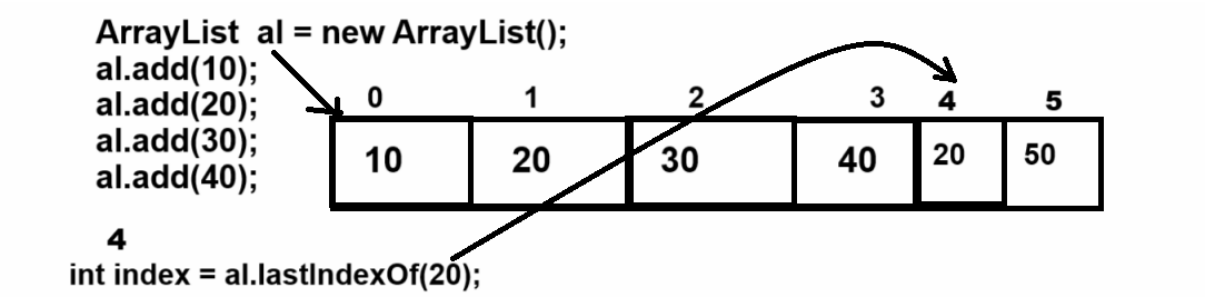

```
Scanner   xyz  = new Scanner(System.in);
System.out.println("enter value for search");
int value=xyz.nextInt(); //30

int index = al.indexOf(value);
 if(index!=-1)
 {  System.out.println("Data found");
 }
 else
 { System.out.println("Data not found");
 }
```

b. Before call remove method for avoid IndexOutOfBoundsException

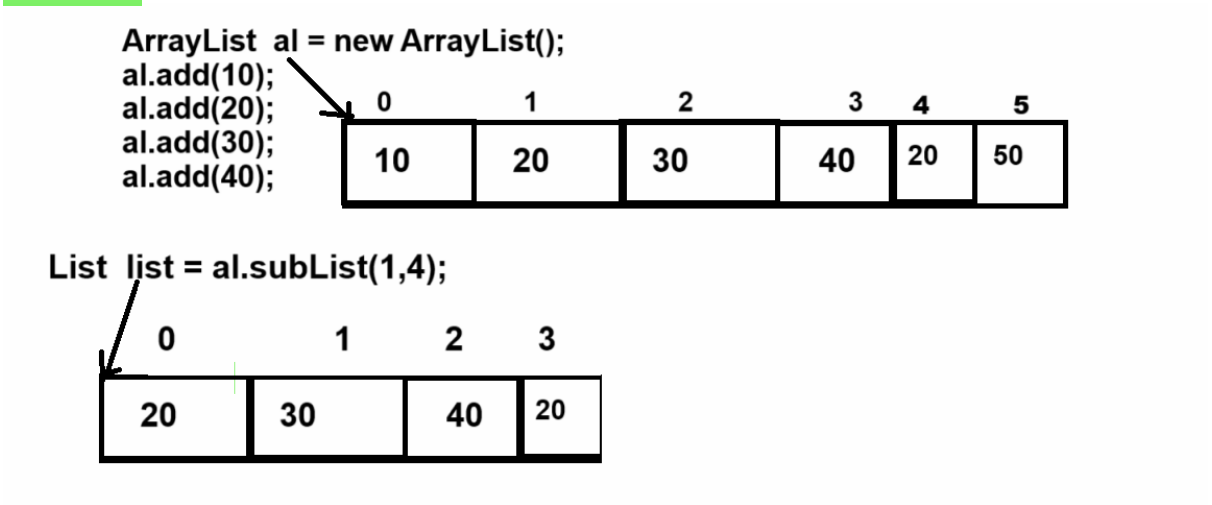



```
System.out.println("enter value for delete ");
int value = xyz.nextInt(); // 30
2 int index= al.indexOf(value); //30
if(index!=-1)
{
   al.remove(index);  //20
}
else
{
    System.out.println("value not found for remove or delete");
}
```

public abstract int lastIndexOf(java.lang.Object): this method return the last occurrence of index



public abstract java.util.List<E> subList(int, int): this method helps us extract the some specified portion from List collection between two indexes.



Vector class

Vector is a dynamic array internally and it is known legacy collection

Q. What is a legacy collection?

Legacy collection means a classes which is part of previous version of JAVA before collection but later they are added as part of collection called as legacy collection

Vector was added as part of the collection in JDK 1.2 before that vector known as dynamic array in JAVA.

Important points related with vector

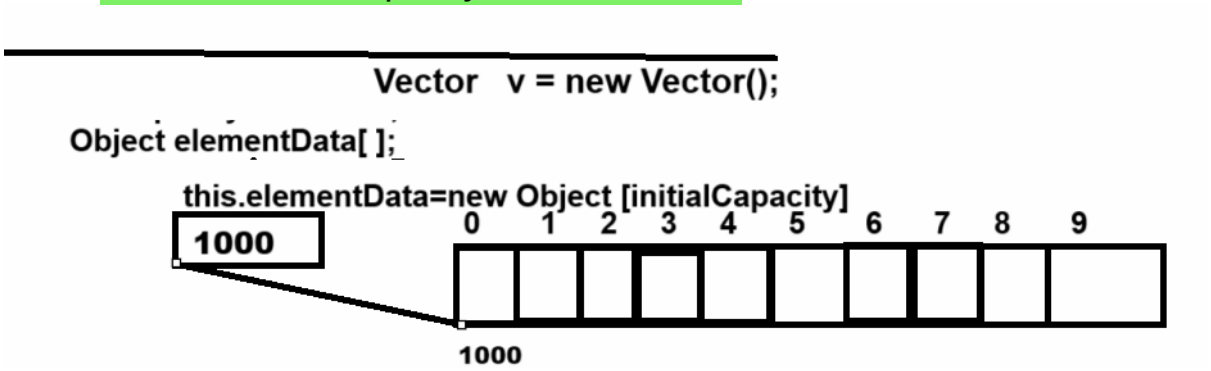
---

1. Vector is thread safe collection
2. Vector legacy collection

3. Default capacity of Vector is 10
4. Vector allocate memory double thant its current capacity so the threshold of Vector is 1.0 or 100%

If we want to work with Vector we have some constructor provided by Vector to us

1. **Vector():** When we use this constructor creating Vector object then internally we get array of Object class with initial capacity 10 Means default capacity of Vector is 10.



If we want to cross verify the default capacity of a vector is 10 then we have a method name as int capacity() using this method we can check the capacity of the vector.

```
1  package org.techhub;
2  import java.util.*;
3  public class TestVectorApp {
4
5      public static void main(String[] args) {
6          Vector v = new Vector();
7          int capacity =v.capacity();
8          System.out.println("Initial Capacity of vector is "+capacity);
9      }
10 }
11
```

```
Initial Capacity of vector is 10
```

Important point: when we store the value more than the current capacity of a vector then the vector occupies double memory than its current capacity.
Example: if we create an object of Vector using default constructor then the default capacity of Vector is 10 and if we try to add the 11th element in vector then its new capacity is 20 and if we try to add 21th element then the new capacity is 40 and so on.

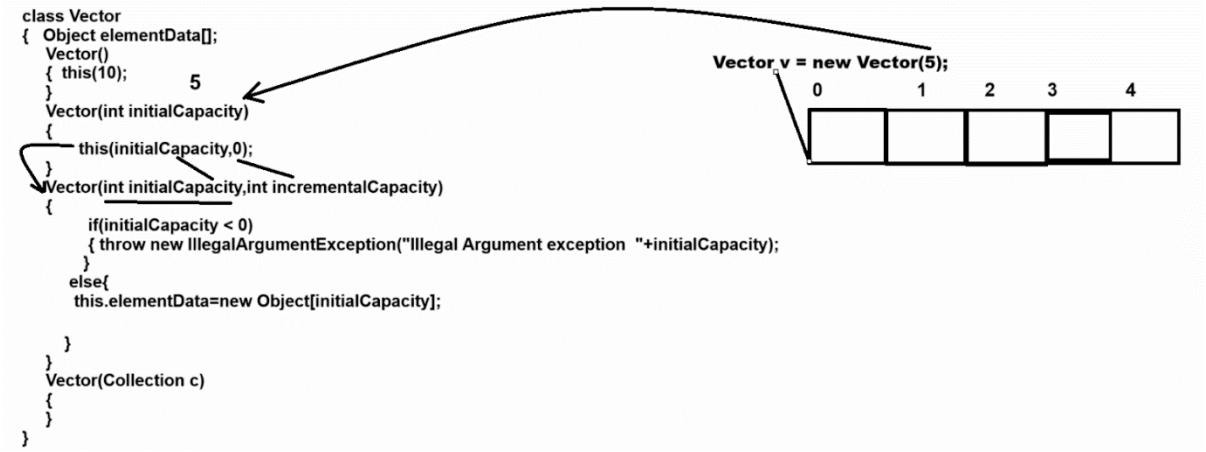
```java
package org.techhub;
import java.util.*;
public class TestVectorApp {

    public static void main(String[] args) {
         Vector v = new Vector();
         int capacity =v.capacity();
         System.out.println("Initial Capacity of vector is "+capacity);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         System.out.println("Size of vector is "+v.size());
         capacity =v.capacity();
         System.out.println("After Capacity cross "+capacity);

    }
}
```

Output

```
Initial Capacity of vector is 10
Size of vector is 12
After Capacity cross 20
```

Note: user can set initial capacity of vector according to his requirement.

2. Vector(int initialCapacity) : this constructor can create a vector with initial capacity as per the user choice**.**



Note: if we set initial capacity of vector according to user choice and if vector cross its current capacity then vector allocate memory double than its current capacity

Note: if we want to decide the incremental capacity of Vector according to user choice not by default double then we have a constructor with two parameters.

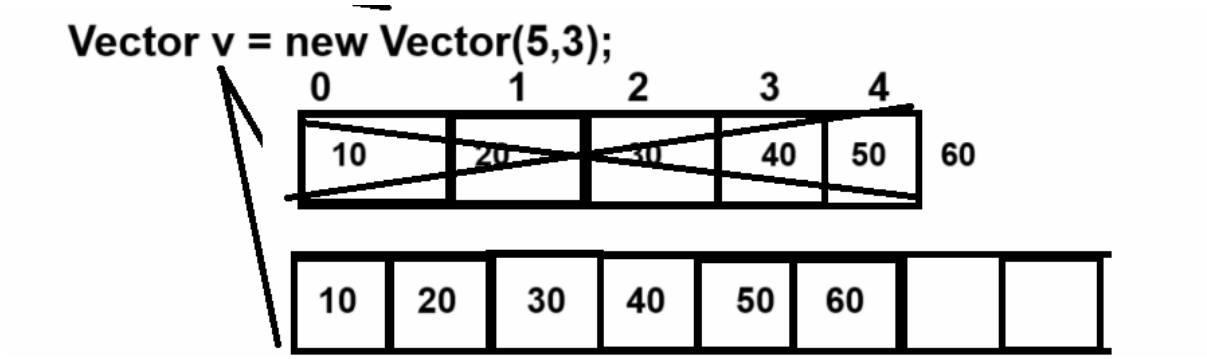
3. Vector(int initialCapacity,int incrementalCapacity)

Parameter Details

---

int initialCapacity: this parameter is used for set initial capacity of vector.

int incrementalCapacity: this parameter is used for decide the incremental capacity of vector



Note: if we think about above constructor we create vector object using Vector v=new Vector(5,3) here we set initial capacity 5 and we set incremental capacity 3 means when we try to add 6th element in vector then vector increase its capacity by 3 blocks only

4. Vector(Collection) : this constructor helps us to copy data from another collection and store it in a Vector object.

```
package org.techhub;
import java.util.*;
public class TestVectorApplication {

	public static void main(String[] args) {
		ArrayList al = new ArrayList();
		al.add(10);
		al.add(20);
		al.add(30);
		al.add(40);
		Vector v = new Vector(al);
		System.out.println(v);
	}
}
```

Example by using Vector

Case 1: Add New Element in Vector
Case 2: View All Elements
Case 3: Count number of element
Case 4: Search element by contains
Case 5: Search Element by index

Case 6: delete element by using index
Case 7: Fetch elements using get() method
Case 8: SubList
Case 9: Remove by element

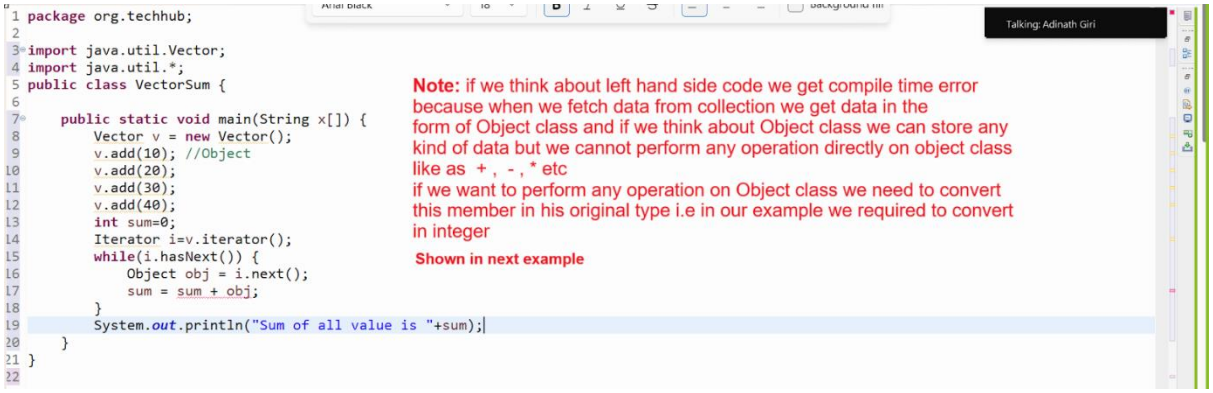
Example with source code

```
package org.techhub;
import java.util.*;
public class TestVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		do {
		Scanner xyz = new Scanner(System.in);
		System.out.println("1: Add Element");
		System.out.println("2: View All  ");
		System.out.println("3: Count number of element ");
		System.out.println("4:Search element by contains method");
		System.out.println("5:Search Element by index");
		System.out.println("6:Delete by using its index");
		System.out.println("7:Fetch elements by using get method");
		System.out.println("8: SubList");
		System.out.println("9: Remove element by value");
		System.out.println("Enter your choice");
			int choice = xyz.nextInt();
			switch (choice) {
			case 1:
				System.out.println("Enter data in vector");
				int value = xyz.nextInt();
				boolean b = v.add(value);
				if (b) {
					System.out.println("element added");
				} else {
					System.out.println("element not added");
				}
				break;
			case 2:
				Iterator i = v.iterator();
				while (i.hasNext()) {
					Object obj = i.next();
					System.out.println(obj);
				}
				break;
```

```
		case 3:
System.out.println("Number of element in vector " + v.size());
			break;
		case 4:
			System.out.println("Enter value for search ");
			value = xyz.nextInt();
			b = v.contains(value);
			if (b) {
				System.out.println("Value found");
			} else {
				System.out.println("Value not found");
			}
			break;
		case 5:
			System.out.println("Enter value for search ");
			value = xyz.nextInt();
			int index = v.indexOf(value);
			if (index != -1) {
				System.out.println("Data found");
			} else {
				System.out.println("Data not found");
			}
			break;
		case 6:
			System.out.println("Enter value for delete ");
			value = xyz.nextInt();
			index = v.indexOf(value);
			if (index != -1) {
				Object obj = v.remove(index);
				System.out.println("Data Deleted " + obj);

			} else {
				System.out.println("Data not found");
			}
			break;
		case 7:
			for (int k = 0; k < v.size(); k++) {
				Object obj = v.get(k);
				System.out.println(obj);
			}
			break;
		case 8:
		System.out.println("enter start index and end index");
```

```
                        int startIndex = xyz.nextInt();
                        int endIndex = xyz.nextInt();
                        if (startIndex >= 0 && endIndex < v.size()) {
                                List list = v.subList(startIndex, endIndex);
                                for (Object obj : list) {
                                        System.out.print(obj + "\t");
                                }
                        }
                        break;
                case 9:
                        System.out.println("Enter value for delete ");
                        value = xyz.nextInt();
                        v.remove((Object)value);
                        break;
                case 10:
                        System.exit(0);
                        break;
                default:
                        System.out.println("Wrong choice");
                }
        } while (true);
    }
}
```

Example: WAP to create vector and store 5 values init and calculate its sum




```
package org.techhub;
import java.util.Vector;
import java.util.*;
public class VectorSum {
public static void main(String x[]) {
                Vector v = new Vector();
                v.add(10); //Object
                v.add(20);
```

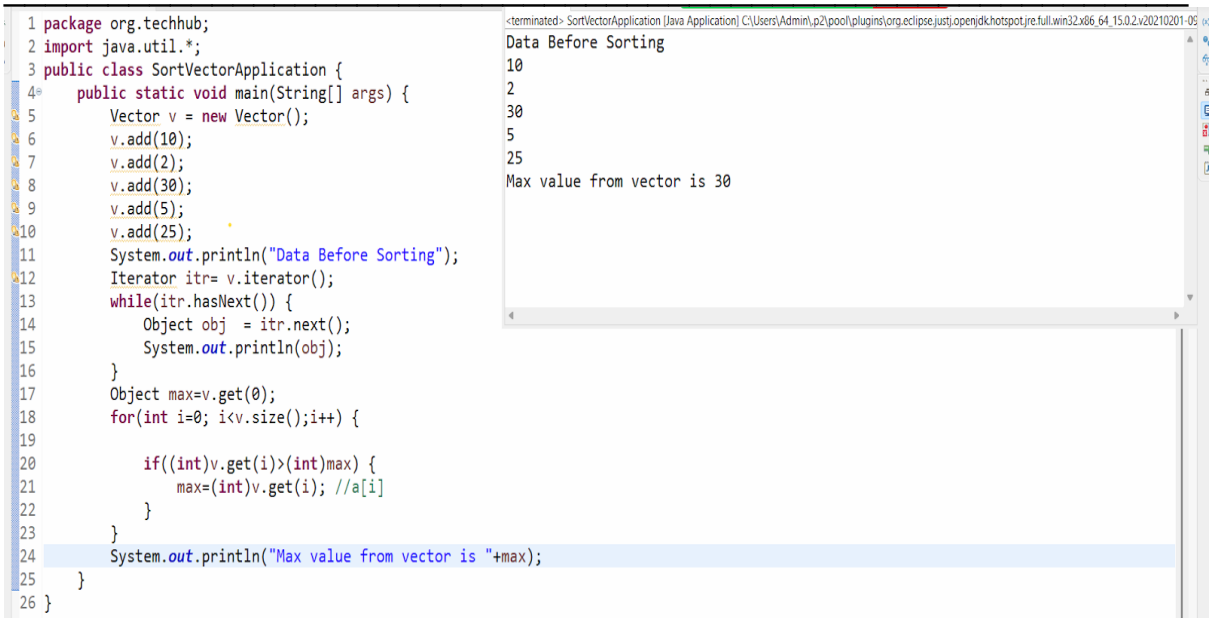
```
		v.add(30);
		v.add(40);
		int sum=0;
		Iterator i=v.iterator();
		while(i.hasNext()) {
			Object obj = i.next();
			sum = sum + (int)obj;
		}
		System.out.println("Sum of all value is "+sum);
	}
}
```

Example: WAP to create Vector and store 5 integer values in it and perform sorting operations on it.

```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		v.add(10);
		v.add(2);
		v.add(30);
		v.add(5);
		v.add(25);
		System.out.println("Data Before Sorting");
		Iterator itr= v.iterator();
		while(itr.hasNext()) {
			Object obj  = itr.next();
			System.out.println(obj);
		}
		//perform sorting
		for(int i=0; i<v.size();i++) {
			for(int j=(i+1); j<v.size();j++) {

				Object prev=v.get(i);
				Object next=v.get(j);
				if((int)prev>(int)next) {
					v.set(i, next);
					v.set(j, prev);
				}
			}
```

```
            }

            System.out.println("Data After Sorting");
              itr= v.iterator();
            while(itr.hasNext()) {
                    Object obj  = itr.next();
                    System.out.println(obj);
            }
        }
}
```

Example: WAP to create Vector and store 5 values in it and find the max value from Vector without using Collections.max()



```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        v.add(10);
        v.add(2);
        v.add(30);
        v.add(5);
        v.add(25);
        System.out.println("Data Before Sorting");
        Iterator itr= v.iterator();
        while(itr.hasNext()) {
            Object obj  = itr.next();
            System.out.println(obj);
        }
        Object max=v.get(0);
        for(int i=0; i<v.size();i++) {

            if((int)v.get(i)>(int)max) {
                max=(int)v.get(i); //a[i]
            }
        }
        System.out.println("Max value from vector is "+max);
    }
}
```

```
<terminated> SortVectorApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-05
Data Before Sorting
10
2
30
5
25
Max value from vector is 30
```

Example: WAP to store five words in Vector and create single string or line of statement using Vector

```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        v.add("Good");
        v.add("bad");
        v.add("abc");
        v.add("mno");
        v.add("pqr");

        String str="";
        Iterator i=v.iterator();
        while(i.hasNext()) {
            Object obj = i.next();
            str=str+" "+(String)obj;//obj.toString()
        }
        System.out.println("String is  "+str);
    }
}
```

```
<terminated> SortVectorApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\java
String is   Good bad abc mno pqr
```

How to store user defined objects in Collection

Q. Why do we need to store user-defined objects in collections?

When we want to add more than one element of different type as a single value in a collection then we can store user defined objects in the collection.

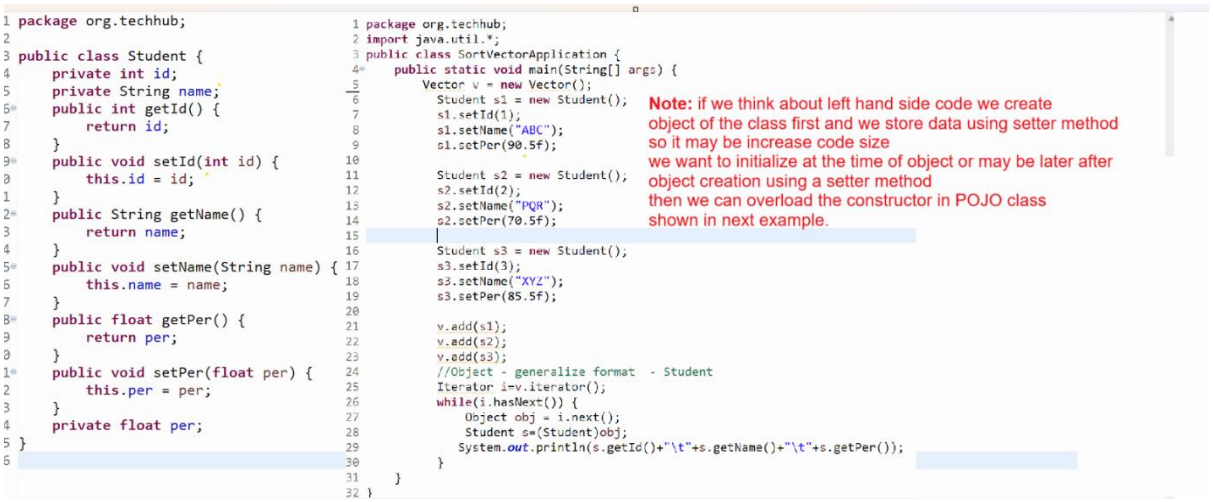
Example: we want to create a collection of students with data id,name and per here id has integer data type, name has string data type and per has float data type.

If we want to solve above problem then we can create POJO class name as Student with field id, name and per and store data in Student object and store in collection i.e in Vector according to our example and when we want to fetch data from collection then we required convert that data in Student object from Object class.

Example with source code

```
package org.techhub;
public class Student {
	private int id;
	private String name;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPer() {
		return per;
	}
	public void setPer(float per) {
```

```
		this.per = per;
	}
	private float per;
}
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		  Student s1 = new Student();
		  s1.setId(1);
		  s1.setName("ABC");
		  s1.setPer(90.5f);
		  Student s2 = new Student();
		  s2.setId(2);
		  s2.setName("PQR");
		  s2.setPer(70.5f);
		  Student s3 = new Student();
		  s3.setId(3);
		  s3.setName("XYZ");
		  s3.setPer(85.5f);

		  v.add(s1);
		  v.add(s2);
		  v.add(s3);
		  //Object - generalize format  - Student
		  Iterator i=v.iterator();
		  while(i.hasNext()) {
			Object obj = i.next();
			Student s=(Student)obj;

System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
		  }
	}
}
```



```
package org.techhub;

public class Student {
    private int id;
    private String name;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;
}
```

```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
          Student s1 = new Student();
          s1.setId(1);
          s1.setName("ABC");
          s1.setPer(90.5f);

          Student s2 = new Student();
          s2.setId(2);
          s2.setName("PQR");
          s2.setPer(70.5f);

          Student s3 = new Student();
          s3.setId(3);
          s3.setName("XYZ");
          s3.setPer(85.5f);

          v.add(s1);
          v.add(s2);
          v.add(s3);
          //Object - generalize format  - Student
          Iterator i=v.iterator();
          while(i.hasNext()) {
              Object obj = i.next();
              Student s=(Student)obj;
             System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
          }
    }
}
```

Note: if we think about left hand side code we create object of the class first and we store data using setter method so it may be increase code size
we want to initialize at the time of object or may be later after object creation using a setter method
then we can overload the constructor in POJO class shown in next example.

Example : Source code with parameterized constructor

```
package org.techhub;
public class Student {
        private int id;
        private String name;
        public Student() {

        }
        public Student(String name,int id,float per) {
                this.name=name;
                this.id=id;
                this.per=per;
        }
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public float getPer() {
                return per;
        }
        public void setPer(float per) {
                this.per = per;
        }
```

```
	private float per;
}
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		  Student s1 = new Student("ABC",1,90.5f);

		  Student s2 = new Student("PQR",2,70.5f);

		  Student s3 = new Student("XYZ",3,85.0f);

		  v.add(s1);
		  v.add(s2);
		  v.add(s3);
		  //Object - generalize format  - Student
		  Iterator i=v.iterator();
		  while(i.hasNext()) {
			Object obj = i.next();
			Student s=(Student)obj;
	System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
		  }
	}
}
Or
package org.techhub;

public class Student {
	private int id;
	private String name;
	public Student() {

	}
	public Student(String name,int id,float per) {
		this.name=name;
		this.id=id;
		this.per=per;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
```

```
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPer() {
		return per;
	}
	public void setPer(float per) {
		this.per = per;
	}
	private float per;
}

package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		  v.add(new Student("ABC",1,90.5f));
		  v.add(new Student("PQR",2,70.5f));
		  v.add(new Student("XYZ",3,85.0f));
		  Iterator i=v.iterator();
		  while(i.hasNext()) {
			Object obj = i.next();
			Student s=(Student)obj;

System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
		  }
	}
}
```

Example: WAP to create Vector and store five book objects in vector and search book using id from collection.

```
package org.techhub;
import java.util.*;
class Book{
	private int id;
	private String name;
```

```
	public Book() {

	}
	public Book(String name,int id,float price) {
		this.name=name;
		this.id=id;
		this.price=price;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPrice() {
		return price;
	}
	public void setPrice(float price) {
		this.price = price;
	}
	private float price;


}
public class BookApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		Book b1 = new Book("ABC",1,100);
		Book b2 = new Book("MNO",2,200);
		Book b3 = new Book("PQR",3,300);
		Book b4 = new Book("STV",4,400);

		v.add(b1);
		v.add(b2);
		v.add(b3);
		v.add(b4);
		System.out.println("Display book data");
```

```
		Iterator itr=v.iterator();
		while(itr.hasNext()) {
			Object obj = itr.next();
			Book b =(Book)obj;

	System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice());
		}
		Scanner xyz = new Scanner(System.in);
		System.out.println("Enter book id for search");
		int bid=xyz.nextInt();
		boolean flag=false;
		for(int i=0;i<v.size();i++) {

			Book b=(Book)v.get(i);
			if(b.getId()==bid) {
				flag=true;
				break;
			}
		}
		if(flag) {
			System.out.println("Book found");
		}
		else {
			System.out.println("Book not found");
		}
	}
}
```

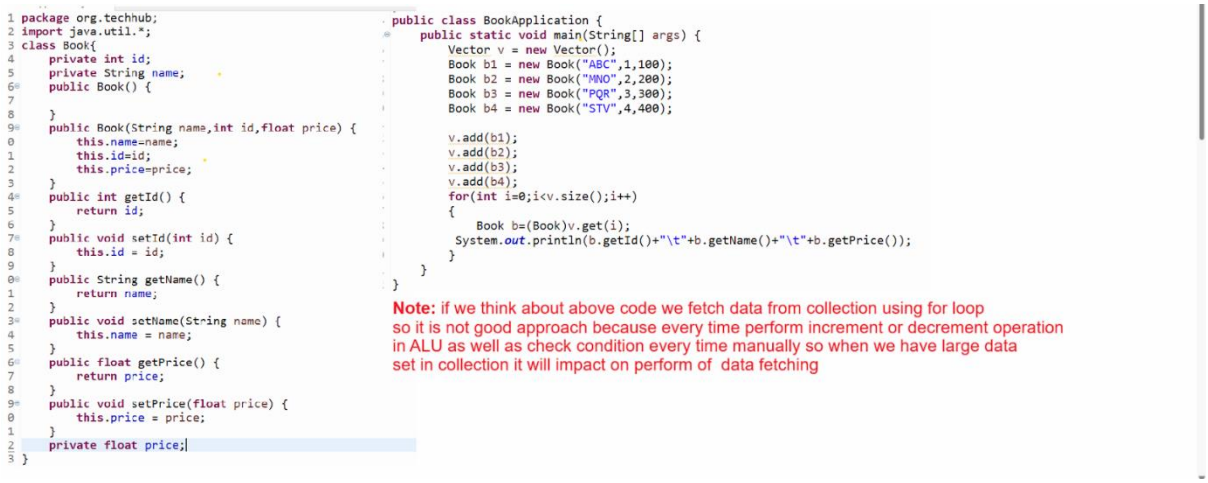
Cursor in Collection

Cursor are the some iterators which help us to fetch data from collection

Types of Cursor in Collection

---

1. **For loop :** but normally we fetch using for loop from collection who maintain the index like as List Collection, Queue collection etc



```java
package org.techhub;
import java.util.*;
class Book{
    private int id;
    private String name;
    public Book() {

    }
    public Book(String name,int id,float price) {
        this.name=name;
        this.id=id;
        this.price=price;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    private float price;
}
```

```java
public class BookApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        Book b1 = new Book("ABC",1,100);
        Book b2 = new Book("MNO",2,200);
        Book b3 = new Book("PQR",3,300);
        Book b4 = new Book("STV",4,400);

        v.add(b1);
        v.add(b2);
        v.add(b3);
        v.add(b4);
        for(int i=0;i<v.size();i++)
        {
            Book b=(Book)v.get(i);
         System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice());
        }
    }
}
```

Note: if we think about above code we fetch data from collection using for loop so it is not good approach because every time perform increment or decrement operation in ALU as well as check condition every time manually so when we have large data set in collection it will impact on perform of data fetching

2. Enumeration : Enumeration is a cursor in collection which helps us to fetch data from legacy collections only like as Vector etc

If we want to work with Enumeration or create reference of Enumeration we have elements() method of Collection
Syntax: Enumeration ref = collref.elements();

Note: Enumeration is read only cursor means using Enumeration we can fetch data from collection only not perform any other operation on Collection using Enumeration like as removing element or adding element etc

Methods of Enumeration

---

boolean hasMoreElements(): this method can help us to check data present in collection or not if present return true otherwise return false.

Object nextElement(): this method can fetch data from collection and move cursor on next element

```
package org.techhub;
import java.util.*;
class Book{
        private int id;
        private String name;
        public Book() {

        }
        public Book(String name,int id,float price) {
                this.name=name;
                this.id=id;
                this.price=price;
```

```
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPrice() {
		return price;
	}
	public void setPrice(float price) {
		this.price = price;
	}
	private float price;
}
public class BookApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		Book b1 = new Book("ABC",1,100);
		Book b2 = new Book("MNO",2,200);
		Book b3 = new Book("PQR",3,300);
		Book b4 = new Book("STV",4,400);

		v.add(b1);
		v.add(b2);
		v.add(b3);
		v.add(b4);
		Enumeration enm = v.elements();
		while(enm.hasMoreElements()) {
			Object ele=enm.nextElement();
			Book b=(Book)ele;

	System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice
());
		}
	}
}
```

3. Iterator : Iterator helps us to fetch data from collection and provide some inbuilt methods to fetch data from collection.

boolean hasNext(): this method checks if an element is present in collection or not if present return true otherwise return false.

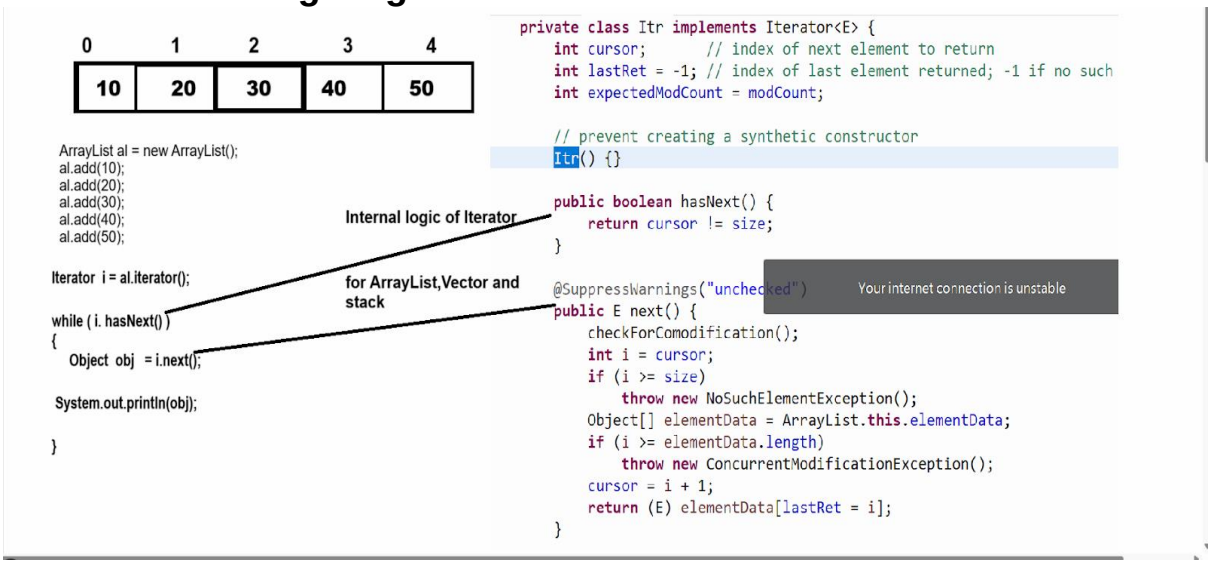
Object next(): this method fetches data from collection and moves the cursor on the next element.

void remove(): this method can remove data from collection at the time data traveling or traversing.

How does Iterator works internally with ArrayList,Vector,Stack and LinkedList?

ArrayList,Vector and Stack are the dynamic arrays internally i.e Object [ ] to store elements.
So if we think about Iterator with these three classes
The iterator maintains the index to track the current element position.
next() method return elementData[index++] and hasNext() check index!=size

Show in following diagram

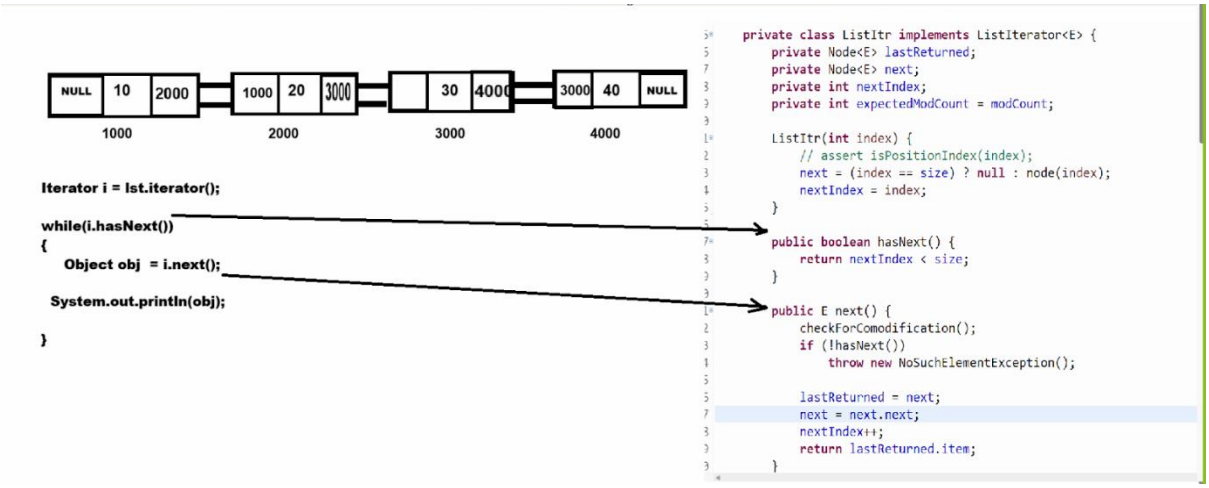


Internal working of Iterator for LinkedList

If we think about linked list in Java internally it is doubly LinkedList where each node holds references to its previous and next element as well as item or data for that purpose LinkedList has one inner class name as Node which contain three data i.e prev,next, E item

boolean hasNext(): method check the index<size in linked list for check data availability if index<size return true otherwise return false.

Object next(): this method can point one pointer on current node i.e lastReturn=next and move cursor on next node using ponter or reference of next node i.e next= next.next and increase index by 1 Index++ for move cursor on next node for comparing with size in hasNext() method and return data using lastReturn.item
Shown in following diagram



If we want to work with iterator we need to know two major important points

---

a. ModCount concept
b. Fail Fast Concept

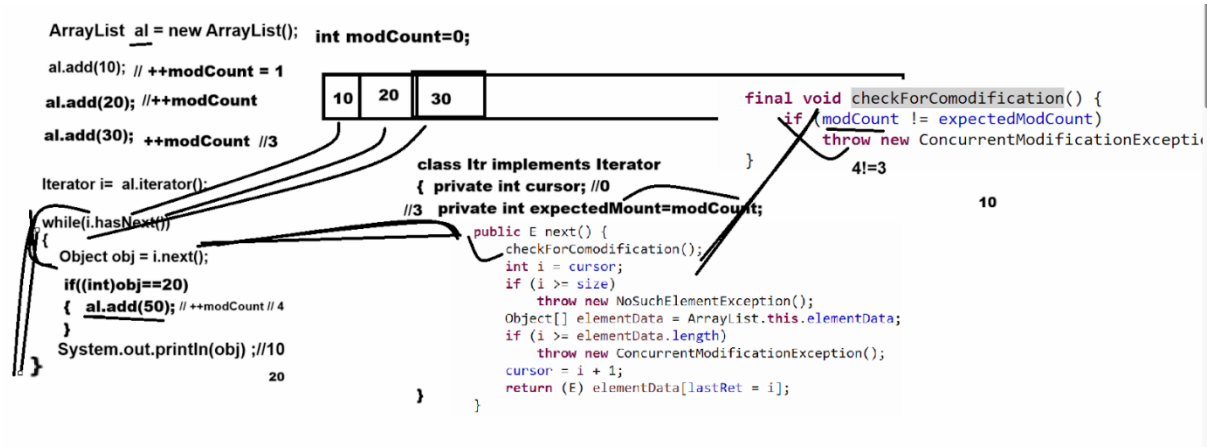
ModCount concept in Java Collection

ModCount stands for modification count. It is an integer field to use track number of structural modifications made to collection. It plays a crucial role in ensuring fail-fast behaviour of iterators by detecting concurrent modification duration iteration

ModCount integer field present in collection classes like as ArrayList, LinkedList , HashMap or Vector etc
It keep track of structural modification to the collection such as

1. Adding elements
2. Removing elements
3. Clearing the collection

Note: structural modification are operations that alter the collection size or its internal structure



Example with source

```
package org.techhub;

import java.util.Vector;
import java.util.*;
public class VectorSum {

	public static void main(String x[]) {
		ArrayList v  = new ArrayList();
		 v.add(10);
		 v.add(20);
		 v.add(30);
		 v.add(40);
		 Iterator i = v.iterator();
		 while(i.hasNext()) {
			Object obj = i.next();
			if((int)obj==30) {
				i.remove();
			}
		 }
		 System.out.println(v);
	}
}
```

4. ListIterator : ListIterator is the child interface of Iterator and it is used for travel the collection in forward direction as well as in backward direction and ListIterator only works with List Collection, not other means it is not an universal cursor.

How to create reference of ListIterator

---

boolean hasNext(): check element present in collection or not in forward direction traveling
boolean hasPrevious(): this method check element present in collection or not in backward direction
Object next(): this method can fetch data from collection and move cursor on next element
Object previous(): this method can fetch data from collection and move cursor previous element
void add(Object): add new element in collection at the time of data adding

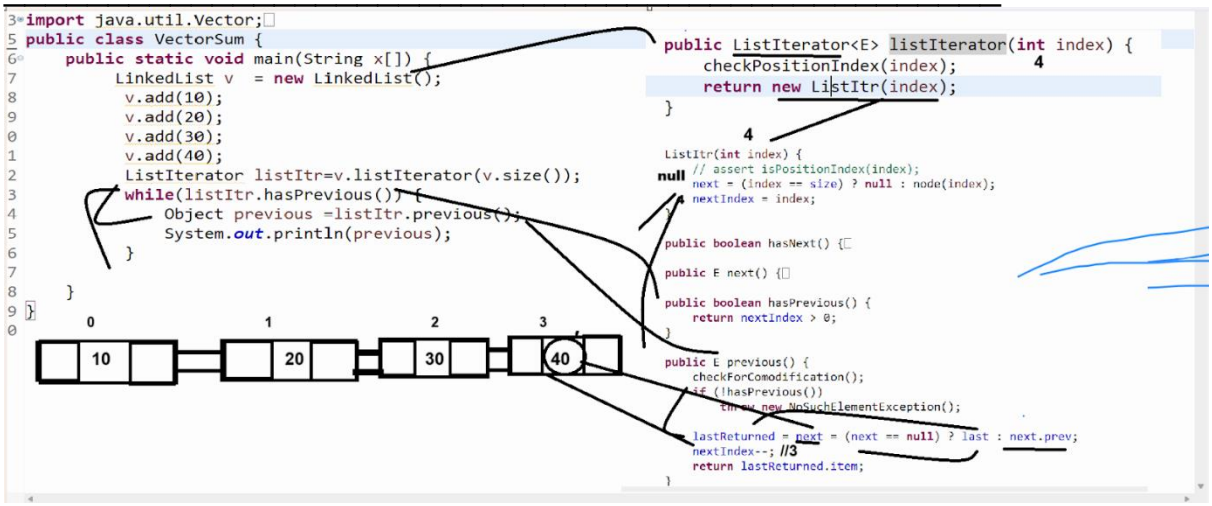
void remove(): this method can remove data from collection using ListIterator
void set(Object): this method can replace element in collection using ListIterator
Int previousIndex(): return the index of previous element
Int nextIndex(): return the index of next element

Example of ListIterator using Vector class

```
package org.techhub;

import java.util.Vector;
import java.util.*;
public class VectorSum {

        public static void main(String x[]) {
                Vector v  = new Vector();
                 v.add(10);
                 v.add(20);
                 v.add(30);
                 v.add(40);
                 ListIterator listItr=v.listIterator(v.size());
                 while(listItr.hasPrevious()) {
```

```
                Object previous =listItr.previous();
                System.out.println(previous);
            }

        }
}
```

Example of ListIterator using LinkedList



Q. What is the difference between Iterator and List Iterator?

a. **Travelling capabilities /Traversal capabilities**

Iterator: Travel collection or fetch data from collection using forward direction or in only one direction.
ListIterator: Travel collection or fetch data from collection using forward direction as well as using backward direction.

b. **Element Modification:**
Iterator: Iterator allow only removal operation at the time of travelling using remove() method
ListIterator: ListIterator can allo add(),remove(), set() - replace element in collection at the time of travelling.

c. **Index Access:**
Iterator: Iterator not provide method to access the index of collection at the time of traveling
ListIterator: ListIterator provides two methods to us: int previousIndex() , int nextIndex() for fetch index from collection at the time to traversing.

d. **Methods available**

Iterator: Iterator contain methods boolean hasNext(),Object next(), void remove()

ListIterator: ListIterator contain methods boolean hasNext(),boolean hasPrevious(),Object next(),Objext previous(), int previousIndex(),int nextIndex() , void add(E),void set(E),void remove() etc

e. **Performance Consideration**

1. For ArrayList both Iterator and ListIterator are the fast due to the index access
2. For LinkedList ListIterator is preferable because it avoids redundant travel when moving backward as LinkedList is optimized for bidirectional iteration.

How to create Fail safe collection

_____

If we want to create fail safe collection we have CopyOnWriteArrayList class

Basically CopyOnWriteArrayList is member of java.util.concurrent package and design to handle the concurrent modification like as fail safe collection means not throws ConcurrentModification Exception

```
package org.techhub;

import java.util.Vector;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.*;
public class VectorSum {
	public static void main(String x[]) {
		CopyOnWriteArrayList v=new CopyOnWriteArrayList();
		  v.add(10);
		  v.add(20);
		  v.add(30);
		  v.add(40);
		  Iterator i = v.iterator();
		  System.out.println("Before Adding "+v);
		  while(i.hasNext()) {
			Object obj = i.next();
			if((int)obj==30) {
				v.add(50);
			}
```

```
            }
            System.out.println("After adding  "+v);

        }
}
```

How CopyOnWriteArrayList works

---

1. **Snapshot mechanism :** when iterator is created it work on a snapshot (copy) of the underlying array at the time of iterator creation

2. Structural modification: operations like add(),remove() or set() create a new copy of the entire underlying array with the modification applied.

3. Iterator behaviour : Iterator operator on snapshot so they are unaffected by concurrent modification to the list during iteration.

5. Foreach loop : for each is an enhancement for loop introduced by java in JDK 1.5 version and it specially designed for fetch data from array or collection in forward directions only.

Important points related with for each

---

1. Fetch data only by using forward direction
2. Not need to manually
3. Not need to check condition direct fetch from 0 to n-1 index
4. By default increment by 1 internally
5. Cannot travel data using backward direction
6. Not need to use index for fetch data directly we get data from array or collection

Syntax: for(data type variable:array/collection)

```
        {
        }
```

Example with source code

```
package org.techhub;

import java.util.Vector;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.*;
```

```
public class VectorSum {
	public static void main(String x[]) {
		ArrayList v=new  ArrayList();
		 v.add(10);
		 v.add(20);
		 v.add(30);
		 v.add(40);
		 for(Object obj:v) {
			System.out.println(obj);
		 }
	}
}
```

6. For each method from JDK 1.8 version of java

```
	package org.techhub;
	import java.util.Vector;
	import java.util.concurrent.CopyOnWriteArrayList;
	import java.util.*;
	public class VectorSum {
		public static void main(String x[]) {
			ArrayList v=new  ArrayList();
			 v.add(10);
			 v.add(20);
			 v.add(30);
			 v.add(40);
			 v.forEach((val)->System.out.println(val));
		}
	}
```

ArrayList

_____

ArrayList is a internally dynamic array of Object class and it is implementer class of List collection and it is asynchronous collection Means not thread safe means multiple thread object can use ArrayList simultaneously

Constructor of ArrayList

_____

ArrayList(): this constructor help us to create ArrayList with default size i.e 10

ArrayList(int initialCapacity): this constructor can create ArrayList object with initial capacity provided by user
ArrayList(Collection): This constructor can be used to copy data from another collection and store it in the ArrayList Collection.

Example using ArrayList

```java
package org.techhub;
import java.util.*;
public class ArrayListApplication {
    public static void main(String[] args) {
        ArrayList al = new ArrayList();
        al.add(10);
        al.add(20);
        al.add(30);
        al.add(40);
        for(Object obj:al) {
            System.out.println(obj);
        }
    }
}
```

Output

```
10
20
30
40
```

Q. What is the difference between ArrayList and Vector?

1. Vector is legacy collection and ArrayList is not legacy collection
2. Vector is thread safe or synchronized and ArrayList is not thread safe or non synchronized collection
   Means performance wise ArrayList is faster than Vector but not thread safe
3. Vector allocates double memory than its current capacity when capacity crosses and ArrayList occupies half memory than its current capacity when capacity crosses.

Logic of ArrayList capacity increment

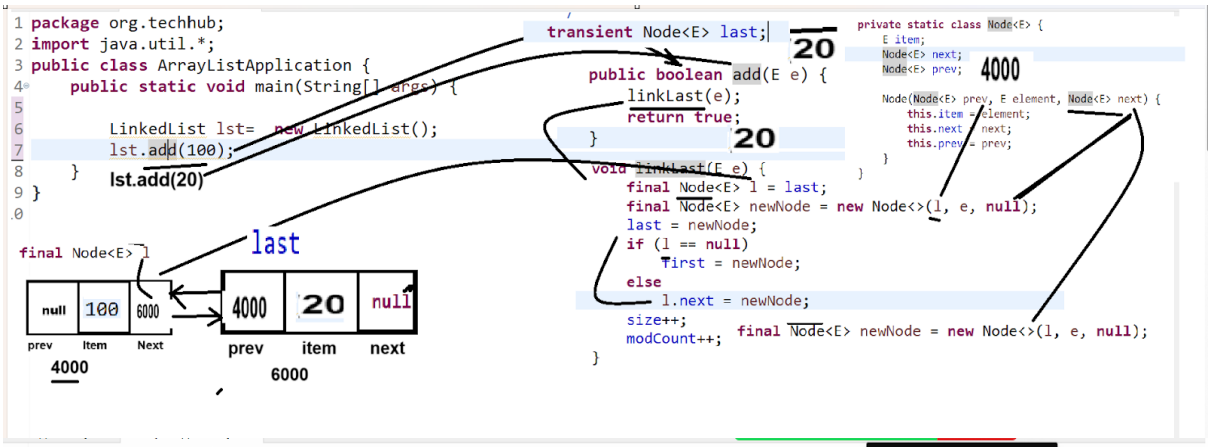
```
int newCapacity = size > currentCapacity ? currentCapacity+currentCapacity>>1 : currentCapacity;
                                          10+ 10>>1
                                          10+ 5
```

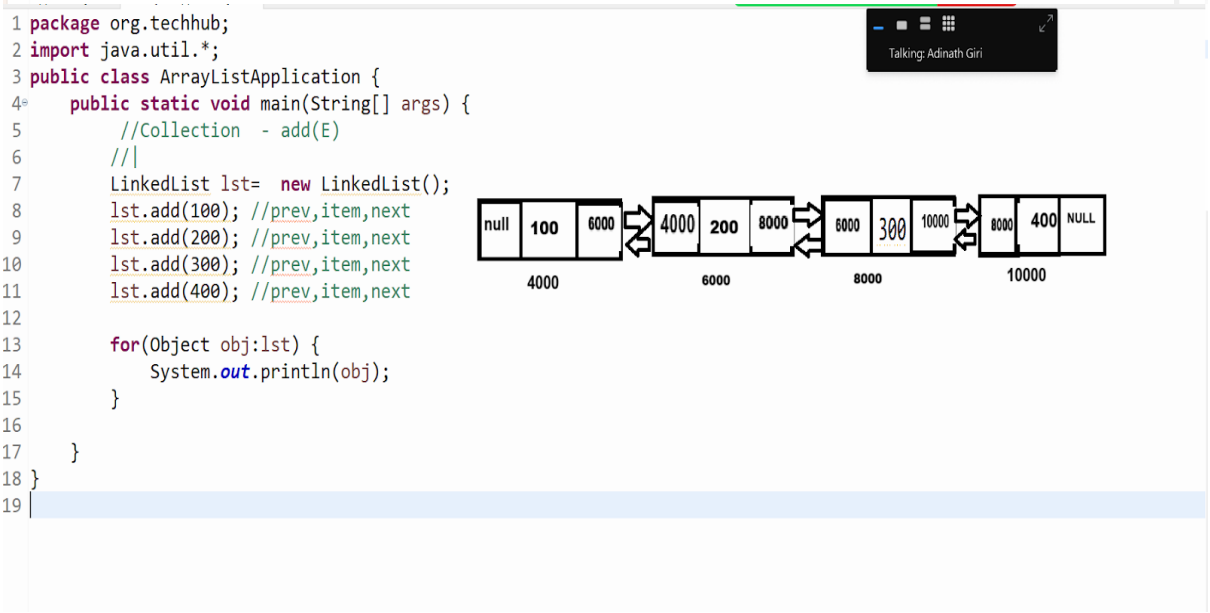
4. Threshold of Vector is 1.0 or 100% and Threshold of ArrayList is 0.5

LinkedList

If we think about LinkedList it is internally implementation of doubly LinkedList and doubly linked list is combination of node with three field Prev,next and item and prev is reference of Node which is help us to hold address of previous node and next is a reference of Node which

help us to store address of Next node and item is Object data type variable which represent internally by E generic notation which help us to store any kind of data or object in LinkedList





Q. What is the difference between ArrayList and LinkedList?

---

1. **Data Structure implementation differences**

   ArrayList: ArrayList is internally dynamic array means ArrayList implemented by using array but resizable nature
   Means when we add element in ArrayList it is add on specified location using index
   LinkedList: LinkedList implemented by using doubly linked list internally means when we add data in linked list it is added as node with three field prev,item and next

2. **Performance differences :**

ArrayList and LinkedList has some permanence difference with operation like as fetching data or insert or deletion data as well as memory usage

Accessing or Fetching data

_____

ArrayList : ArrayList provide constant time complexity (1) for data fetching by using indexing

LinkedList: Access Time is Linear O(n) because it need to travel list from beginning to end means we need to search address of next node for fetch data means LinkedList fetch data of current node and move to next node by address so we required to travel linked from beginning to ending

Insertion and deletion Operation

_____

ArrayList: Insertion and deletion operation ArrayList required indexing shifting so need travel ArrayList for shift indexing for insertion and deletion so it may make some time so insertion and deletion operation slower by ArrayList than LinkedList means ArrayList O(n) time complexity for insertion and deletion

LinkedList : insertion and deletion operation perform faster by LinkedList because LinkedList only shift or manipulate address of node prev and next pointer without traveling so it is faster than ArrayList means LinkedList use O(1) for insertion and deletion

Memory usage:

_____

ArrayList: ArrayList required less memory than LinkedList because ArrayList contain only data not address of another node they connected each other by using sequential address technique so easy for access via index

LinkedList: LinkedList required more memory than ArrayList because LinkedList contain three elements prev,next and node

3. **Uses cases:**

ArrayList: when we have list and we want to frequently search or fetching operation on collection then ArrayList is recommended but when have list and we want to perform continuous insertion or deletion then ArrayList is not recommended

LinkedList: When we required collection with frequently insertion or deletion then LinkedList is recommend and for fetching and searching LinkedList is not recommended

4. **Interface Implementation Differences**
   ArrayList: ArrayList only implement List interface so it act as only list collection

   LinkedList: LinkedList implements List interface as well as can implement DQueue interface means we can say we can use LinkedList as List as well as Queue

Q. What are the similarities between ArrayList and LinkedList?

1. **Interface implementation**: If we think About ArrayList and LinkedList both implements List interface so both classes contain common methods

2. **Ordered collection :** both classes provide data accessing as per the user data sequence means data fetching using ArrayList and LinkedList approach is same
   Means both provide user sequence access
3. **Duplicate elements :** Both are implementer classes of List Collection and List collection allow duplicate elements so ArrayList and LinkedList also allow duplicate elements.

4. **Allow Null Values:** ArrayList and LinkedList both classes can hold null values.
5. **Resizable :** ArrayList is internally array but it is dynamic array so we can resize it as well as LinkedList is dynamic data structure so it is also resizable
6. **Support iterators :** ArrayList and LinkedList support Iterators for data fetching
7. **ThreadSafety :** both classes contain asynchronous method so they are not thread safe

Stack: Stack is last in first out data structure means first insert data remove last and last insert data remove first and stack has a single pointer known as top and when we insert element in stack then top increases by 1 and when remove data from stack then top decreases by 1. Stack is a child of Vector

Methods of Stack collection

E push(E): this method is used for push data or insert data in stack
E pop(): this method can remove data from stack and remove top most data from stack.
E peek(): this method can peek last index data means just we can view last index data but not remove using peek
boolean isEmpty(): this method can check elements present in collection or if present return true otherwise return false.

Note: when we fetch stack using iterator then it look like as Array Not stack so we required to fetch using ListIterator

Example: we want to perform following operation on Stack
Case 1: push
Case 2: pop
Case 3: display

Example with source code

```
package org.techhub;

import java.util.*;

public class ArrayListApplication {
	public static void main(String[] args) {
		Stack s = new Stack();
		do {
			Scanner xyz = new Scanner(System.in);
			System.out.println("1:INSERT");
			System.out.println("2:POP");
			System.out.println("3:DISPLAY");
			System.out.println("Enter your choice");
			int choice = xyz.nextInt();
			switch (choice) {
			case 1:
				System.out.println("Enter value in stack");
				int value = xyz.nextInt();
				s.push(value);
				break;
			case 2:
				boolean b = s.isEmpty();
				if (b) {
				System.out.println("There is no data in stack");
```

```
                        } else {
                    System.out.println("Removed data is  " + s.pop());
                        }
                        break;
                    case 3:
                        ListIterator li =s.listIterator(s.size());
                        while(li.hasPrevious()) {
                            Object obj  = li.previous();
                            System.out.println(obj);
                        }
                        break;
                    case 4:
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Wrong choice");
                    }
                } while (true);// infinite loop

        }
}
```

Example: we want to develop the application maintaining a question bank and every question contains the following data i.e qid , name , op1, op2, op3, op4 answer and we want to perform the following operation to perform the above task.

Case 1: Add New Question in Collection
Case 2: View All Question
Case 3: Search question by using question id
Case 4: remove question using question id

```
package org.techhub;

import java.util.*;

public class QuestionBank {
        public static void main(String[] args) {
                ArrayList al = new ArrayList();
                do {
                        Scanner xyz = new Scanner(System.in);
                        System.out.println("1:Add New Question");
                        System.out.println("2:View All Questions");
```

```
                System.out.println("3:Search Question ");
                System.out.println("4:Delete Question By Id");
                System.out.println("Enter your choice");
                int choice = xyz.nextInt();
                switch (choice) {
                case 1:
        System.out.println("Enter qid name and all options and answer");
                        int qid = xyz.nextInt();
                        xyz.nextLine();
                        String question = xyz.nextLine();
                        String op1 = xyz.nextLine();
                        String op2 = xyz.nextLine();
                        String op3 = xyz.nextLine();
                        String op4 = xyz.nextLine();
                        String ans = xyz.nextLine();
Question q = new Question(qid, question, op1, op2, op3, op4, ans);
                        al.add(q);
                        break;
                case 2:
                        Iterator i = al.iterator();
                        while (i.hasNext()) {
                                Object obj = i.next();
                                Question ques = (Question) obj;
System.out.println(ques.getQid() + "\t" + ques.getName() + "\t" +
ques.getOp1() + "\t"
                                                + ques.getOp2() + "\t" +
ques.getOp3() + "\t" + ques.getOp4() + "\t" + ques.getAnser());
                        }
                        break;
                case 3:
                        System.out.println("Enter question id for search
question");
                        int questionId = xyz.nextInt();
                        i = al.iterator();
                        boolean flag = false;
                        while (i.hasNext()) {
                                Object obj = i.next();
                                Question ques = (Question) obj;
                                if (ques.getQid() == questionId) {
                                        flag = true;
                                        break;
                                }
                        }
```

```
                        if (flag) {
                                System.out.println("Question found in
database or collection");
                        } else {
                    System.out.println("Question not found");
                        }
                        break;
                    case 4:
        System.out.println("Enter question id for search question");
                        questionId = xyz.nextInt();
                        i = al.iterator();
                        flag = false;
                        while (i.hasNext()) {
                                Object obj = i.next();
                                Question ques = (Question) obj;
                                if (ques.getQid() == questionId) {
                                        int index = al.indexOf(ques);
                                        if (index != -1) {
                                                al.remove(index);
                                                flag = true;
                                                break;
                                        }
                                }
                        }
                        if (flag) {
        System.out.println("Question removed from collection");
                        } else {
                                System.out.println("Question not found");
                        }
                        break;
                    default:
                        System.out.println("Wrong choice");
                    }

                } while (true);
        }
}
```

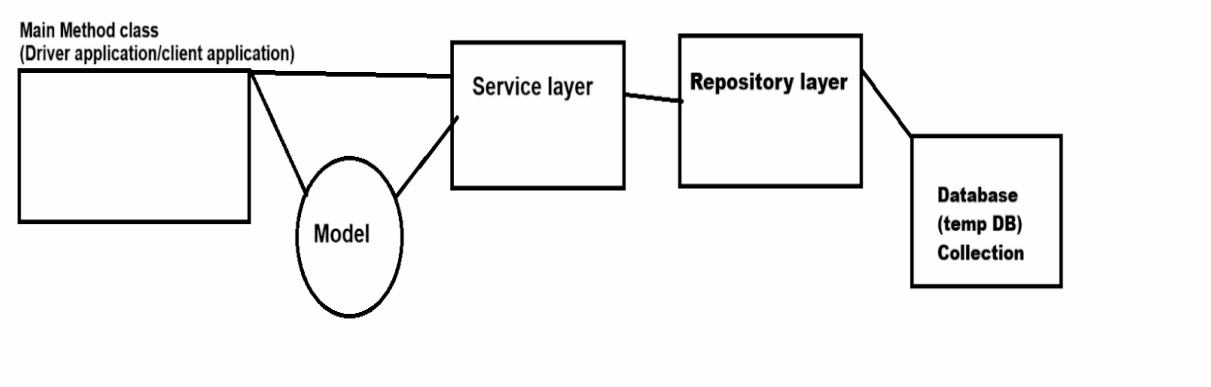
Store Management Application using Collection

1. Add Products in collection
2. View All products in collection

3. Search Product from collection
4. Delete Product from collection
5. Count total number of products from collection
6. Create Customer collection
7. Maintain purchase order of customer
8. View individual custom orders
9. View all customer reports
10. Delete customer and its order
11. Update customer orders

Etc

Coding layer



What is a client application?

Client application or driver class is a class which contain main methods where user can provide input and get results
And we required to object of service layer and model class in client application and using model class we can pass data from client application to service layer and service layer to repository

What is model class?

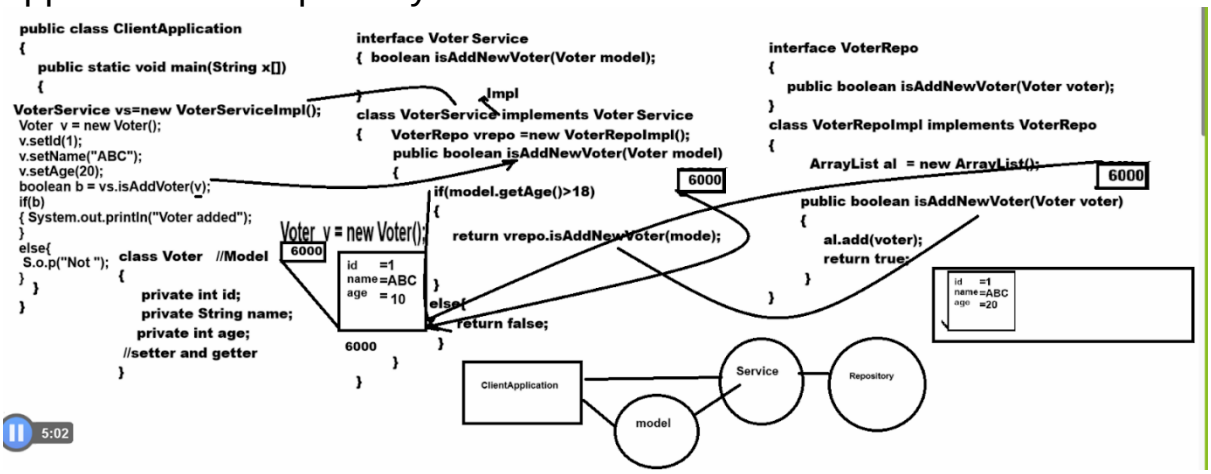
Model class is a class with setter and getter methods i.e POJO class which is used for store data or store different types of data
Means here model class work as container means it help us to store data and pass in different layer in application

What is the service layer?

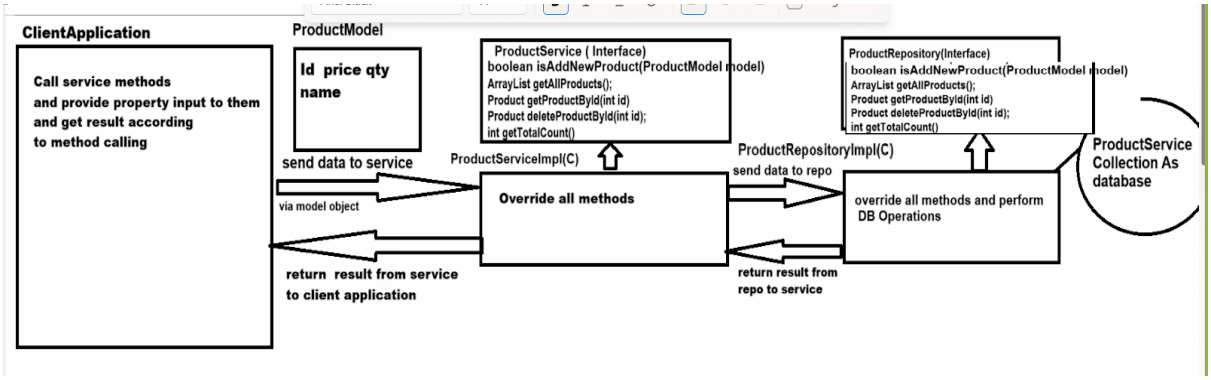
Service layer class which is used for write business logics in application

Note: business logic is not fix it is vary from requirement to requirement

What is the repository layer?
Repository layer means class which contain database logics
Flow: we need to create service layer object in client application , model class object in class or catch model class reference return by service in client application , we need to create object of repository class in service layer means service method call from main method class i.e using client application and repository methods call from service classes



Product Model



public boolean isAddNewProduct(ProductModel): this method can accept product model class and store in database i.e in collection

Logic need to implement we add new products

---

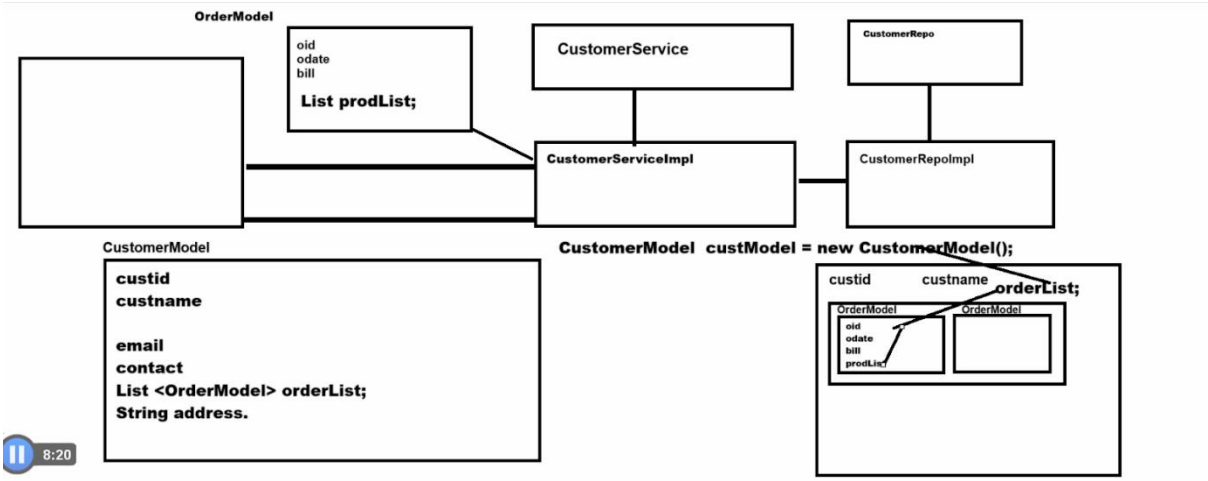
1. Product id should not duplicate
2. Product name should not be blank
3. Quantity must be know means quantity should not zero and price also should not zero

public ArrayList getAllProducts(): this method can return all products from database but if product not found in database then system should generate user define exceptions product not found

public Product getProductById(int id) : this method accept id from keyboard and return product data if id not found then generate user exception name as product not found exception

public Product deleteProductById(int id) : this method can accepted from keyboard and return product data and if id not found then return product not found exception

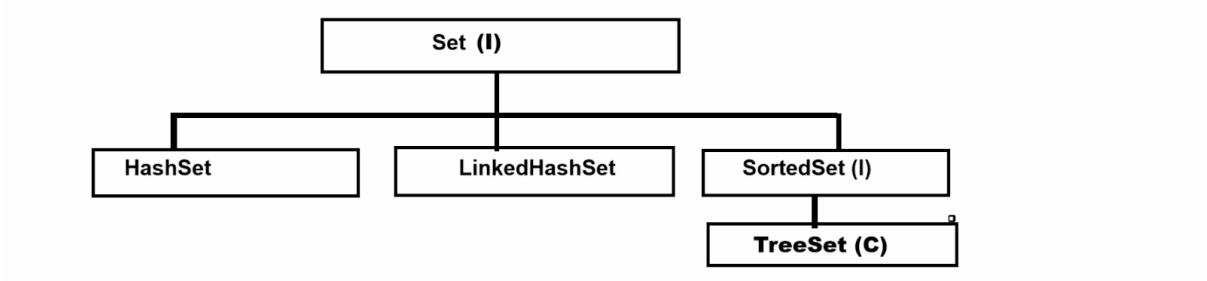
int getTotalCount(): this method can return total count of products



Note: remaining task should be complete by students
We will check 31/03/2024 -audi of assignment

Set Collection

Set Collection does not allow duplicate elements and internally set collection internally works with data structure name as hash table . If we think about set collection we have the following implementer classes.



HashSet : HashSet can store unique element but generate random data means there is no sequence for data generation

```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        HashSet hs = new HashSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
99
20
100
5
22
23
567
10
234
45
```

LinkedHashSet : LinkedHashSet can store unique element but store data sequence according to user sequence

```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        LinkedHashSet hs = new LinkedHashSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
5
20
10
22
45
23
100
99
567
234
```

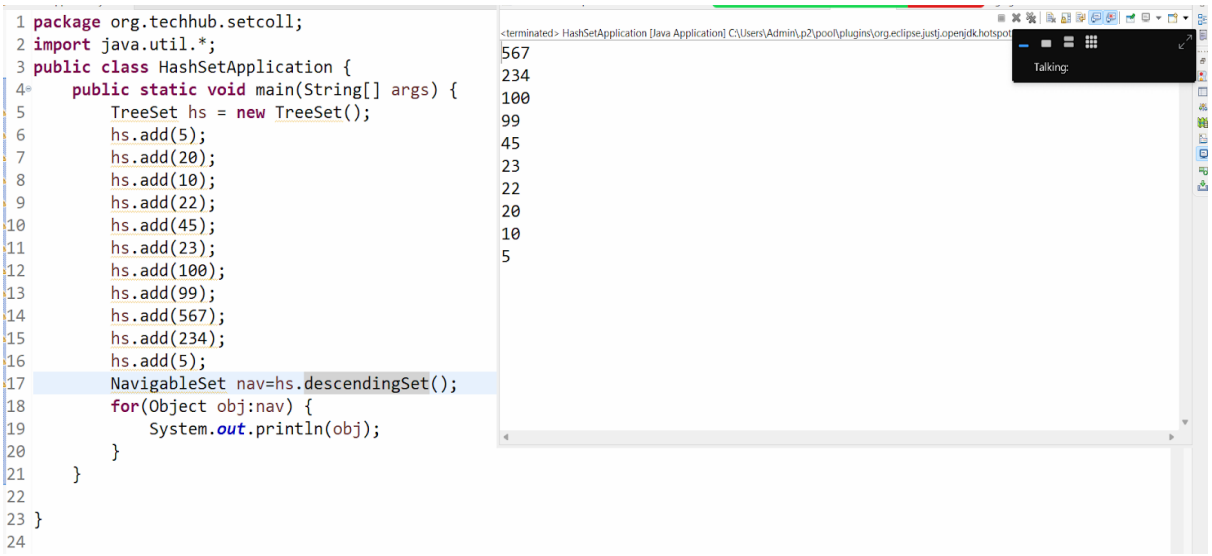
TreeSet : TreeSet Collection can store data in sorted format and arrange data in ascending order

```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        TreeSet hs = new TreeSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
5
10
20
22
23
45
99
100
234
567
```

Note: if we want to organize your data in descending order for TreeSet we have method name as descendingSet() and this method can

generate objects of the NavigableSet interface and NavigableSet is the interface for fetching data in descending order.



```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        TreeSet hs = new TreeSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        NavigableSet nav=hs.descendingSet();
        for(Object obj:nav) {
            System.out.println(obj);
        }
    }

}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
567
234
100
99
45
23
22
20
10
5
```

HashSet class in depth working

HashSet works internally using Hashing technique

1. **Hashing :** In Hashing technique or hashing process
   There are two things involved
   a. **Hash function :** each element inserted into hashset is pass through the hash function which is calculate or computer unique integer called as hashcode and using hashcode decide element should place in hashtable or not

   b. **Hash code :** the hashcode represents the element as an integer value which is then mapped with to an index in the internally array of the hashset i.e bucket

   Q. What is a bucket?

   ---

   a. Bucket is part of hashtable use by the hashset consists of an array of buckets each bucket can have one or more elements

   b. The hashcode of the element is used to determine which bucket the element will go into.

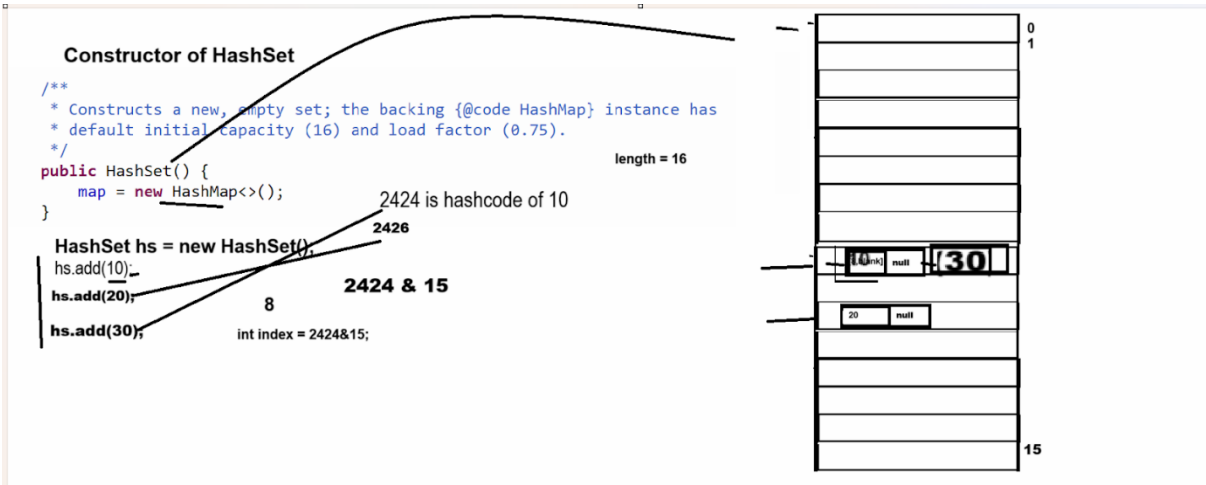
c. There is possibility if two element has same hashcode so their index may be same then there is possibility of data overriding for avoid this problem hashtable use one technique name as collision (where bucket holds linked list of element) or open addressing (where we try to store data in other bucket until bucket empty one found)

Constructor of HashSet

HashSet(): if we use default constructor of HashSet then internally this constructor use HashMap data structure and create one array of bucket with capacity 16 with load factor 0.75

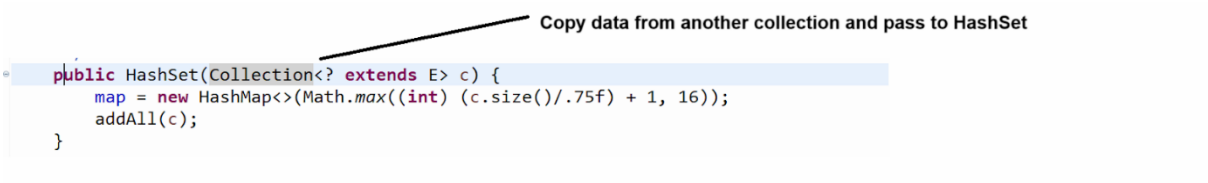
Q. What is the load factor?

Load factor is threshold which decide HashSet can increase capacity or not means here HashSet use default load factor 0.75 means up to 12 element HashSet capacity remain 16 i.e 0.75 but when try to insert 13th element i.e we try cross load factor or threshold set by hashset then hashset allocate memory double than its current capacity i.e 32



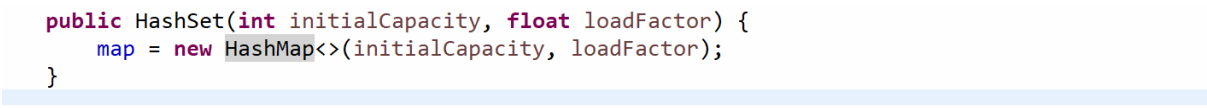
Constructor of HashSet

Syntax: HashSet(Collection): copy data from another collection and store in HashSet collection



Example: WAP to create ArrayList and remove duplicate element from ArrayList

public HashSet(int initialCapacity,float loadFactor): this constructor help us to set the user defined initial capacity and load factor



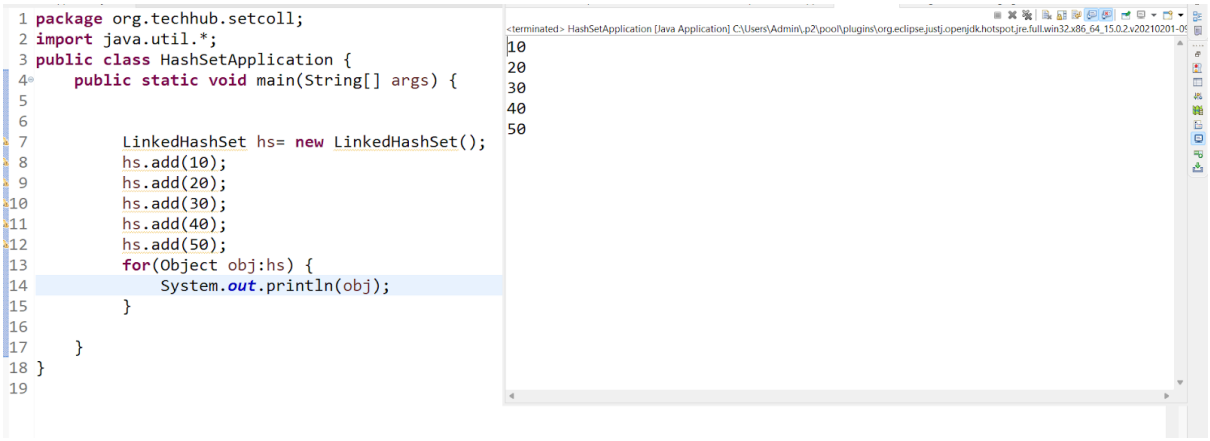
LinkedHashSet: LinkedHashSet is the child of HashSet and internally LinkedHashSet uses LinkedHashMap as data structure.

Constructor of LinkedHashSet

_____

LinkedHashSet(): create internally bucket with default size 16 with load factor 0.75
LinkedHashSet(int initialCapacity,float loadFactor): create LinkedHashSet with initial capacity with default load factor

LinkedHashSet(Collection): this constructor help us copy data from another collection and use as parameter in LinkedHashSet



```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {


        LinkedHashSet hs= new LinkedHashSet();
        hs.add(10);
        hs.add(20);
        hs.add(30);
        hs.add(40);
        hs.add(50);
        for(Object obj:hs) {
            System.out.println(obj);
        }

    }
}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-09
10
20
30
40
50
```

TreeSet Collection

TreeSet() : internally use TreeMap Data Structure
TreeSet(Comparator): this constructor help us to sort the data when user pass user defined object
TreeSet(Collection): this constructor help us to accept data from another collection and sort it
Etc

Collections class

Collections is a utility class of JAVA which is used for perform regular data structure operation on Collection framework
Like as finding max element from collection ,finding element from collection, reverse the collection, sort the list collection, convert asynchronized collection to synchronized collection etc

Example: WAP to create ArrayList and perform operation on in it
Case 1: find the max value from array
Case 2: find the min value from array
Case 3: reverse the arraylist
Case 4: sort the ArrayList
Etc

```
package org.techhub.collectionsapp;
import java.util.*;
public class TestCollectionsApp {
	public static void main(String[] args) {
		ArrayList al = new ArrayList();
		al.add(5);
		al.add(10);
		al.add(100);
```

```
		al.add(20);
		al.add(50);
		al.add(90);
		al.add(40);

		Object maxVal = Collections.max(al);
		System.out.println("Max value from ArrayList is "+maxVal);
		Object minVal = Collections.min(al);
		System.out.println("Min value from ArrayList is "+minVal);
		Collections.sort(al);
		System.out.println(al);
		Collections.reverse(al);
		System.out.println(al);
	}
}
```

Example: WAP to create class name as Employee with field id, name and salary and store five employee objects in ArrayList and sort it

```
package org.techhub.collectionsapp;
import java.util.*;
class Employee{
	private int id;
	private String name;
		private int sal;
		public Employee() {
		}
		public Employee(String name,int id,int sal) {
			this.name=name;
			this.id=id;
			this.sal=sal;
		}
		public int getId() {
			return id;
		}

		public void setId(int id) {
			this.id = id;
		}

		public String getName() {
			return name;
		}
```

```
		public void setName(String name) {
			this.name = name;
		}

		public int getSal() {
			return sal;
		}

		public void setSal(int sal) {
			this.sal = sal;
		}
}
public class TestCollectionsApp {
	public static void main(String[] args) {

		ArrayList al = new ArrayList();

		Employee e1 = new Employee("ABC",3,10000);
		Employee e2 = new Employee("PQR",4,20000);
		Employee e3 = new Employee("STV",1,30000);
		Employee e4 = new Employee("XYZ",2,5000);
		Employee e5 = new Employee("SSSS",5,9000);

		al.add(e1);
		al.add(e2);
		al.add(e3);
		al.add(e4);
		al.add(e5);

		System.out.println("Display before sorting");
		for(Object obj:al) {
			Employee e=(Employee)obj;

	System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
		}
		Collections.sort(al);
		System.out.println("Display after sorting");
		for(Object obj:al) {
			Employee e=(Employee)obj;

	System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
		}

	}
```

}

Output

Note: if we think about above output we have exception ClassCastExcetpion because we have collection with user defined object i.e Employee objects and which contain three types id,name and salary and we store all objects in ArrayList collection and we have Statement Collections.sort(al) here Collections.sort() method get confused which which field employee should sort means using id or name or salary so we get exception means Collections.sort() method by default sort data of collection when collection contain primitive type of data but when Collections contain user defined objects then Collections.sort() method cannot sort data and generate exception at run time so if we want to solve this problem we have two solutions

1. Comparable interface
2. Comparator interface

Q. What is a Comparable interface?

Comparable interface is a member of java.lang package and which is used for perform sorting with user defined objects using List collection or Set collection

Steps to work with Comparable interface

---

1. **Add java.lang package in application**
   Note: we do not need to import the java.lang package because it is the default package of java.
2. **Create POJO class and implements Comparable interface in it**

```
class Employee implements Comparable{
        private int id;
        private String name;
```

```
        private int sal;
        public Employee() {
        }
        public Employee(String name,int id,int sal) {
                this.name=name;
                this.id=id;
                this.sal=sal;
        }
        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getSal() {
                return sal;
        }

        public void setSal(int sal) {
                this.sal = sal;
        }
}
```

3. Override its compareTo() method and perform object comparison

_____

compareTo() method compare current with other elements and perform following things and return some resultant value

1. If current object value is greater than parameter object then return 1

2. If current object value is less than parameter object then return -1
3. If current object is equal with parameter object value return 0

Example with source code

```
package org.techhub.collectionsapp;
import java.util.*;
class Employee implements Comparable{
	private int id;
	private String name;
		private int sal;
		public Employee() {
		}
		public Employee(String name,int id,int sal) {
			this.name=name;
			this.id=id;
			this.sal=sal;
		}
		public int getId() {
			return id;
		}

		public void setId(int id) {
			this.id = id;
		}

		public String getName() {
			return name;
		}

		public void setName(String name) {
			this.name = name;
		}

		public int getSal() {
			return sal;
		}

		public void setSal(int sal) {
			this.sal = sal;
		}
		@Override
		public int compareTo(Object o) {
```

```
                // TODO Auto-generated method stub
                Employee emp=(Employee)o;
                if(this.id>emp.id) {
                        return 1;
                }
                else if(this.id<emp.id) {
                        return -1;
                }
                else {
                        return 0;
                }

        }
}
public class TestCollectionsApp {
        public static void main(String[] args) {
                ArrayList al = new ArrayList();
                Employee e1 = new Employee("ABC",3,10000);
                Employee e2 = new Employee("PQR",4,20000);
                Employee e3 = new Employee("STV",1,30000);
                Employee e4 = new Employee("XYZ",2,5000);
                Employee e5 = new Employee("SSSS",5,9000);

                al.add(e1);
                al.add(e2);
                al.add(e3);
                al.add(e4);
                al.add(e5);

                System.out.println("Display before sorting");
                for(Object obj:al) {
                        Employee e=(Employee)obj;

        System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
                }
                Collections.sort(al);
                System.out.println("Display after sorting");
                for(Object obj:al) {
                        Employee e=(Employee)obj;

        System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
                }
```

```
	}
}
```

Example: WAP to create class name as Player with field id,name and run and store 5 player objects in ArrayList and sort player data by using run.

```
package org.techhub.collectionsapp;
import java.util.*;
class Player implements Comparable{
	private int id;
	private String name;
	public Player() {
	}
	public Player(String name,int id,int run) {
		this.name=name;
		this.id=id;
		this.run=run;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getRun() {
		return run;
	}
	public void setRun(int run) {
		this.run = run;
	}
	private int run;
	@Override
	public int compareTo(Object o) {
		Player p1=(Player)o;
		 return this.run>p1.run?1:this.run<p1.run ?-1:0;
	}
```

```
}
public class PlayerApplication {

	public static void main(String[] args) {
		ArrayList al = new ArrayList();

		Player p1 = new Player("ABC",1,90000);
		Player p2 = new Player("PQR",2,3000);
		Player p3 = new Player("STV",4,12000);
		Player p4 = new Player("XYZ",5,7000);
		Player p5 = new Player("SSS",3,190000);

		al.add(p1);
		al.add(p2);
		al.add(p3);
		al.add(p4);
		al.add(p5);

		System.out.println("Display player record before sorting");
		for(Object obj:al) {
			Player p=(Player)obj;

	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
		}
		Collections.sort(al);

		System.out.println("Display player record After sorting");
		for(Object obj:al) {
			Player p=(Player)obj;

	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
		}
	}

}
```

Comparator interface

Comparator interface is used for perform sorting with a user defined objects but using Comparator we can perform sorting using multiple field

Steps to work with Comparator interface

---

1. **Add java.util package**

2. **Create seperate class for sorting and implements Comparator in it & override compare() method in every implementer class**

Example: suppose consider we want to sort player data by using id,by using name or by using run so we required to declare three class for sorting

a. SortById implements Comparator

```
package org.techhub.collectionsapp;
import java.util.Comparator;
public class SortPlayerById implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		Player p1 = (Player) o1;
		Player p2 = (Player) o2;

		if (p1.getId() > p2.getId()) {
			return 1;
		} else if (p1.getId() < p2.getId()) {
			return -1;
		} else {

			return 0;
		}
	}

}
```

b. SortByName implements Comparator

```
 package org.techhub.collectionsapp;

import java.util.Comparator;

public class SortPlayerByName implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		// TODO Auto-generated method stub
		Player p1=(Player)o1;
		Player p2=(Player)o2;
		return p1.getName().compareTo(p2.getName());
	}
```

```
}
```

c. SortByRun implements Comparator

```
package org.techhub.collectionsapp;

import java.util.Comparator;

public class SortPlayerByRun implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		// TODO Auto-generated method stub
		Player p1 = (Player) o1;
		Player p2 = (Player) o2;
		if (p1.getRun() > p2.getRun()) {
			return 1;
		} else if (p1.getRun() < p2.getRun()) {
			return -1;
		} else {
			return 0;
		}
	}

}
```

Rules of compare() method

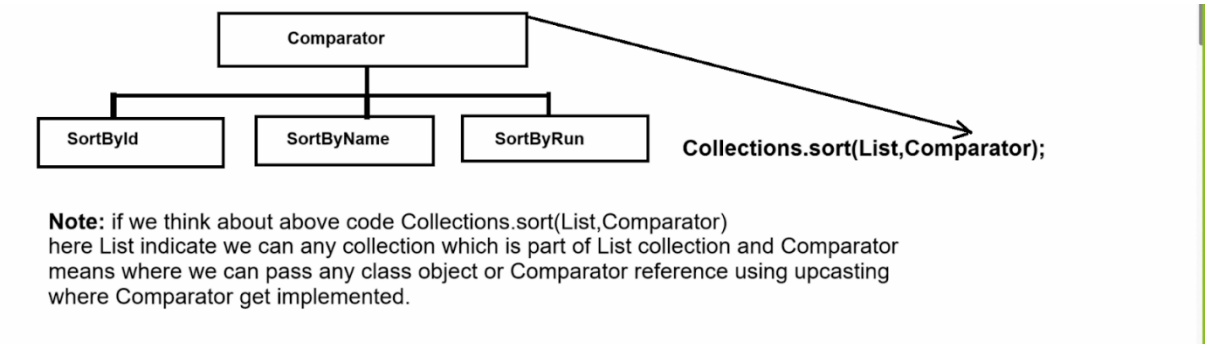
---

a. If first object is greater than second object then return 1
b. If first object is less than second object then return -1
c. If the first object is equal with the second object then return 0.

4. **Use Following method version of Collections.sort()**
Note: when we use the Comparator we have to use Collection.sort() Method using a following version
Note: Collections.sort() it is overloaded method of Collections class

Syntax1: Collections.sort(List): means sort primitive type of list collection or sort user defined object where Comparable get implements

Syntax: Collections.sort(List,Comparator): normally version use when we have Comparator or when we want to perform sorting with multiple field



Main class with Player POJO

```
package org.techhub.collectionsapp;
import java.util.*;
class Player{
	private int id;
	private String name;
	public Player() {
	}
	public Player(String name,int id,int run) {
		this.name=name;
		this.id=id;
		this.run=run;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getRun() {
		return run;
	}
```

```
	public void setRun(int run) {
		this.run = run;
	}
	private int run;

}
public class PlayerApplication {

	public static void main(String[] args) {
		ArrayList ts = new ArrayList();
		Player p1 = new Player("XYZ",1,90000);
		Player p2 = new Player("PQR",2,3000);
		Player p3 = new Player("STV",4,12000);
		Player p4 = new Player("ABC",5,7000);
		Player p5 = new Player("SSS",3,190000);
		ts.add(p1);
		ts.add(p2);
		ts.add(p3);
		ts.add(p4);
		ts.add(p5);


		System.out.println("Display Original Data");
		for(Object obj:ts) {
			Player p=(Player)obj;

	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
		}

		System.out.println("Display player record sort by using id");
		Comparator c=new SortPlayerById();
		Collections.sort(ts,c);
		for(Object obj:ts) {
			Player p=(Player)obj;

	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
		}
		System.out.println("Display player record sort by using
name");
		c=new SortPlayerByName();
		Collections.sort(ts,c);
		for(Object obj:ts) {
			Player p=(Player)obj;
```

```
		System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
			}
			System.out.println("Display player record sort by using run");
			c=new SortPlayerByRun();
			Collections.sort(ts,c);
			for(Object obj:ts) {
				Player p=(Player)obj;

		System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
			}
		}

}
```

How to convert asynchronous collection or map in to synchronous format

---

If we want to convert any asynchronous collection to synchronous collection we have some standard method provided by Collections class to us

List ref= Collections.synchronizedList(List): convert all asynchronous list implementer to synchronized object

Set ref=Collections.synchronizedSet(List)
Collection ref=Collections.synchronizedCollection(Collection)

Map ref=Collections.synchronizedMap(Map);
Etc

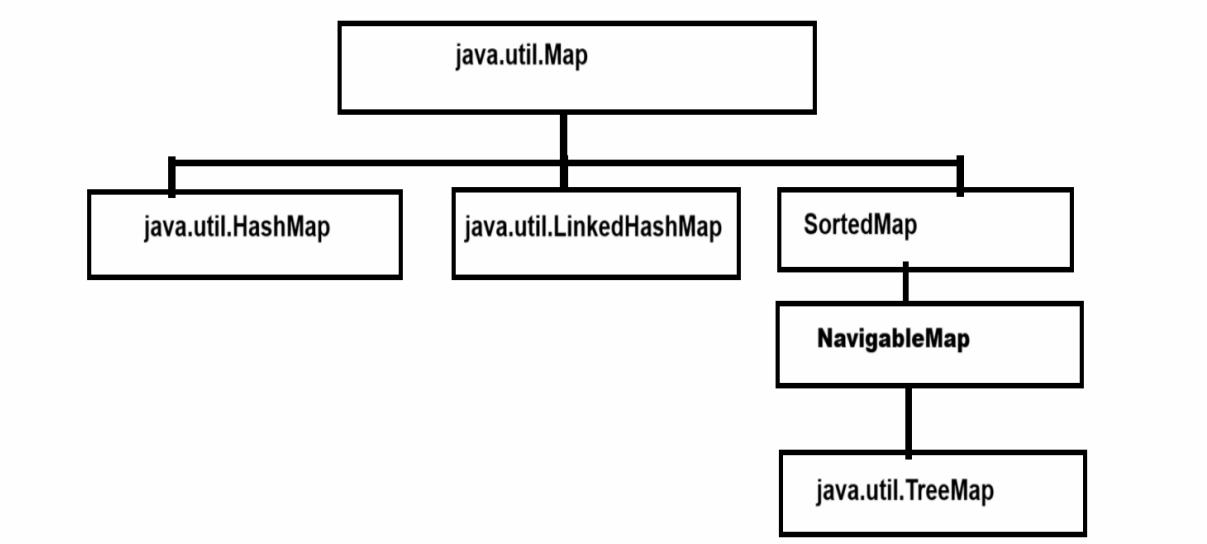
Example : we want to convert ArrayList object as synchronized object

```
package org.techhub.collectionsapp;
import java.util.*;
public class TestArrListApp {
	public static void main(String[] args) {
		List al = new ArrayList();
		al.add(100);
		al.add(200);
		al.add(300);
		List l=Collections.synchronizedList(al);
```

```
            System.out.println(l);
        }

}
```

Map: Map is not part of collection but it is like as Collection and Map can store data in the form of key and value pair and key cannot be duplicated and value may be duplicated
Means we can say map is combination of set and list collection
Set work as key and list work as value
Note: Normally Map recommend when we have duplicated data but we want to maintain uniqueness of that data then map is recommended

Hierarchy of Map



Methods of Map Collection

void put(Object key,Object value) : this method can store data in a map using key and value pairs.

```
HashMap h = new HashMap();

h.put(1,"ABC");
h.put(2,"MNO");
h.put(3,"STV");
h.put(4,"XYZ");
```

key	value
1	ABC
2	MNO
3	STV

Object get(Object key) : this method can fetch from map using key and if key not found return null
Normally we use this method in two cases

a. For fetching data

b. For search data : when the get() method returns a non null value means consider data is present and when get() method returns null then we consider data not present in the map.

```
HashMap h = new HashMap();

h.put(1,"ABC");
h.put(2,"MNO");
h.put(3,"STV");
h.put(4,"XYZ");

Object obj = h.get(1);
    if(obj!=null)
    { System.out.println("Data found");
    }
    else
    {
      System.out.println("Data not found");
    }
```

key	value
1	ABC
2	MNO
3	STV

boolean containsKey(Object key): this method helps us search data from map using key and if key found return true otherwise return false.

```
HashMap  map = new HashMap();
  map.put(1,"ABC");
  map.put(2,"PQR");
  map.put(3,"STV");
  map.put(4,"ABCD");
true
boolean b = map.containsKey(1);

        if(b)
        { System.out.println("Data found");
        }
        else
        { System.out.println("Data not found");
        }
```

1	ABC
2	MNO
3	PQR
4	ABCD

boolean containsValue(Object value) : this method help us to data present in map or not if present return true otherwise return false.

```
HashMap  map = new HashMap();
  map.put(1,"ABC");
  map.put(2,"PQR");
  map.put(3,"STV");
  map.put(4,"ABCD");


boolean   b= map.containsValue("ABCD");

if(b)
{ System.out.println("value found");
}
else
{ System.out.println("Value not found");
}
```

1	ABC
2	MNO
3	PQR
4	ABCD

Object remove(Object key) : this method can remove data from map using its key and return remove value.

Set keySet() : this method can return all keys from map

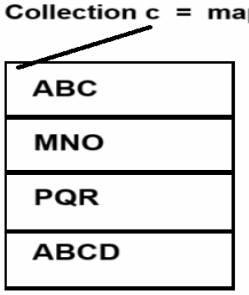
1
2
3
4

```
HashMap  map = new HashMap();
  map.put(1,"ABC");
  map.put(2,"PQR");
  map.put(3,"STV");
  map.put(4,"ABCD");

Set keys  = map.keySet();
```

1	ABC
2	MNO
3	PQR
4	ABCD

Collection values() : this method can return all values from map as collection reference

```
HashMap  map = new HashMap();
  map.put(1,"ABC");
  map.put(2,"PQR");
  map.put(3,"STV");
  map.put(4,"ABCD");

Collection c  =  map.values();
```



1	ABC
2	MNO
3	PQR
4	ABCD

Map.Entry entySet(): this method can return all data from map as Entry object means this method can return data in the form of key and value pair.

```
HashMap  map = new HashMap();
  map.put(1,"ABC");
  map.put(2,"PQR");
  map.put(3,"STV");
  map.put(4,"ABCD");

Set<Map.Entry>  es= map.entrySet();

for(Map.Entry e:es)
{
  System.out.println(e.getKey()+"\t"+e.getValue());
}
```

1	ABC
2	MNO
3	PQR
4	ABCD

int size() : this method can return number of element present in map

boolean isEmpty() : this method can check data present in map or not if present return true otherwise return false.

HashMap

HashMap can generate random data or generate random keys

```java
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        HashMap map = new HashMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}
```

```
<terminated> MapApplication (1) [Java Application] C:\Users\Admin\.p2\pool\plugins\org.ecl
1       ABC
2       MNO
1235    SSSSS
54      XYZ
444     XYZ
44      AAA
99999   SSSSS
```

LinkedHashMap: this map can arrange data in key and value pair but arrange all keys as per user sequence .

```java
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        LinkedHashMap map = new LinkedHashMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}
```

```
<terminated> MapApplication (1) [Java Application] C:\(
1       ABC
2       MNO
444     XYZ
54      XYZ
44      AAA
1235    SSSSS
99999   SSSSS
```

TreeMap: TreeMap can store data in key and value pair but arrange all keys in ascending order.

```java
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        TreeMap map = new TreeMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}
```

```
<terminated> MapApplication (1) [Java
1       ABC
2       MNO
44      AAA
54      XYZ
444     XYZ
1235    SSSSS
99999   SSSSS
```

Example: we want to create a program to perform the following operation on map.

Case 1: store data in map
Case 2: View all data from map
Case 3: search data from map by using key
Case 4: delete data from map using key
Case 5: count total number of elements of map
Case 6: display only keys of map

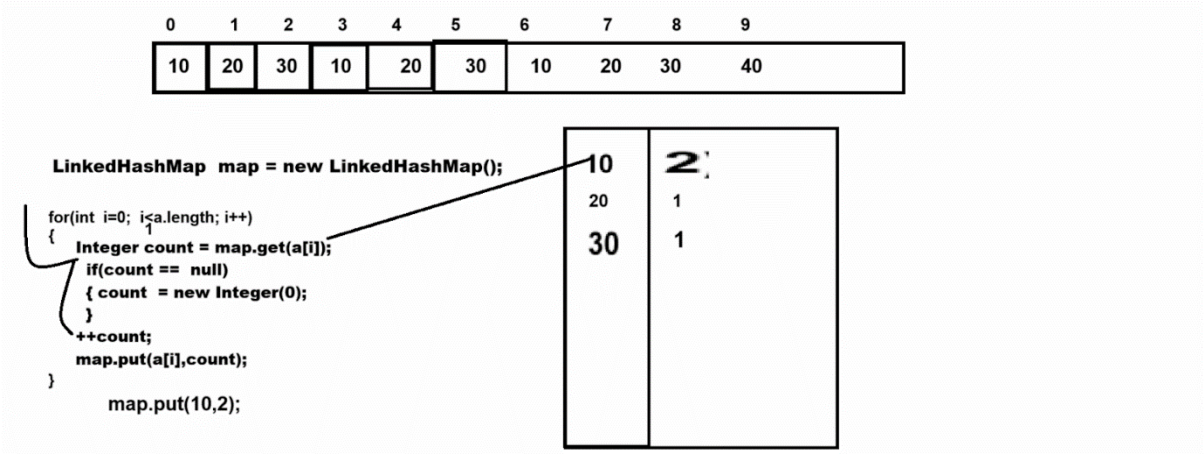
Example:

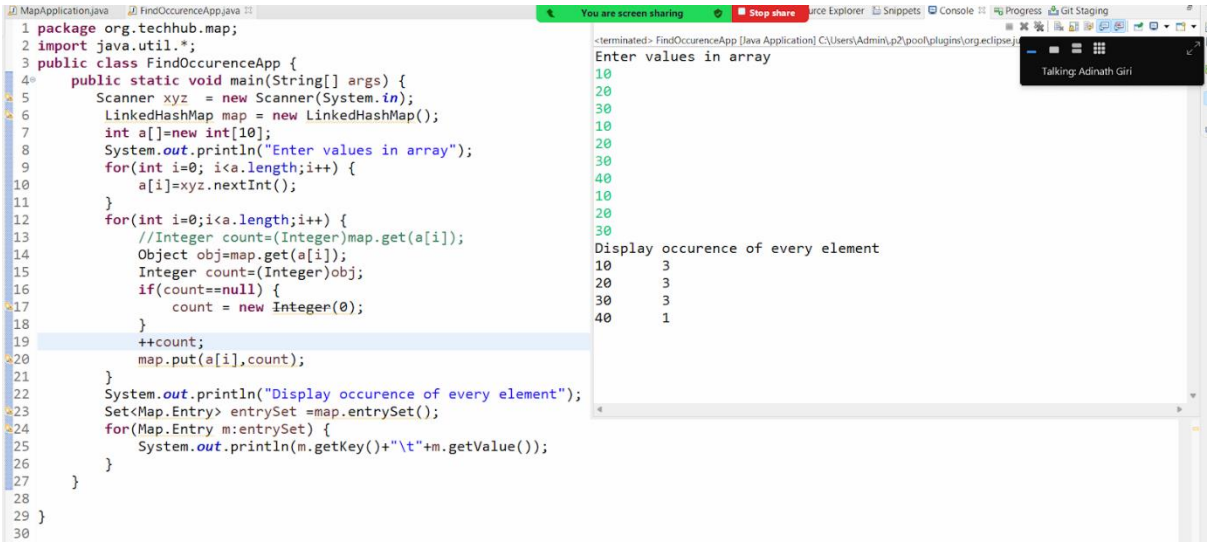
```
package org.techhub.map;
import java.util.*;
public class MapApplication {
	public static void main(String[] args) {
		LinkedHashMap map = new LinkedHashMap();
		do {
			Scanner xyz = new Scanner(System.in);
			System.out.println("1:Add New Data in Map");
			System.out.println("2:View All Data from map");
			System.out.println("3: Search data from map using
key");
			System.out.println("4: Delete data from map using
key");
			System.out.println("5: Count total number of element ");
			System.out.println("6: Display all keys");
			System.out.println("Enter your choice");
			int choice = xyz.nextInt();
			switch (choice) {
```

```
                case 1:
                        xyz.nextLine();
                System.out.println("Enter  name and id of student");
                        String name = xyz.nextLine();
                        int id = xyz.nextInt();
                        map.put(id, name);
                        break;
                case 2:
                        Set<Map.Entry> entrySet = map.entrySet();
                        for (Map.Entry e : entrySet) {
                System.out.println(e.getKey() + "\t" + e.getValue());
                        }
                        break;
                case 3:
                        System.out.println("Enter key for search");
                        int key = xyz.nextInt();
                        boolean b = map.containsKey(key);
                        if (b) {
                                System.out.println("Data found");
                        } else {
                                System.out.println("Data not found");
                        }
                        break;
                case 4:
                        System.out.println("Enter key for search");
                        key = xyz.nextInt();
                        b = map.containsKey(key);
                        if (b) {
                                map.remove(key);
                                System.out.println("Data found");
                        } else {
                                System.out.println("Data not found");
                        }
                        break;
                case 5:
System.out.println("total element present in map " + map.size());
                        break;
                case 6:
                        Set keys = map.keySet();
                        for (Object k : keys) {
                                System.out.println(k);
                        }
                        break;
```

```
                    case 7:
                            System.exit(0);
                            break;
                    default:
                            System.out.println("Wrong choice");
                    }

                } while (true);
        }
}
```

Example: WAP to create array and store 10 values in it and find the occurrence of every element in array by using LinkedHashMap





Example: WAP to find the duplicate elements from array

```java
import java.util.*;

public class FindOccurenceApp {
    public static void main(String[] args) {
        Scanner xyz = new Scanner(System.in);
        LinkedHashMap map = new LinkedHashMap();
        int a[] = new int[10];
        System.out.println("Enter values in array");
        for (int i = 0; i < a.length; i++) {
            a[i] = xyz.nextInt();
        }
        for (int i = 0; i < a.length; i++) {
            // Integer count=(Integer)map.get(a[i]);
            Object obj = map.get(a[i]);
            Integer count = (Integer) obj;
            if (count == null) {
                count = new Integer(0);
            }
            ++count;
            map.put(a[i], count);
        }
        System.out.println("Display occurence of every element");
        Set<Map.Entry> entrySet = map.entrySet();
        for (Map.Entry m : entrySet) {
            if ((Integer) m.getValue() > 1) {
                System.out.println(m.getKey() + "\t" + m.getValue());
            }
        }
    }

}
```

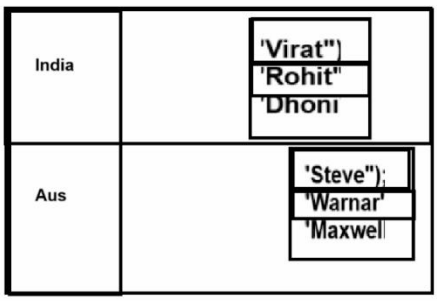
```
Enter values in array
10
20
30
40
50
10
20
30
10
20
Display occurence of every element
10      3
20      3
30      2
```

Example: WAP to find unique values from array

```java
import java.util.*;

public class FindOccurenceApp {
    public static void main(String[] args) {
        Scanner xyz = new Scanner(System.in);
        LinkedHashMap map = new LinkedHashMap();
        int a[] = new int[10];
        System.out.println("Enter values in array");
        for (int i = 0; i < a.length; i++) {
            a[i] = xyz.nextInt();
        }
        for (int i = 0; i < a.length; i++) {
            // Integer count=(Integer)map.get(a[i]);
            Object obj = map.get(a[i]);
            Integer count = (Integer) obj;
            if (count == null) {
                count = new Integer(0);
            }
            ++count;
            map.put(a[i], count);
        }
        System.out.println("Display occurence of every element");
        Set<Map.Entry> entrySet = map.entrySet();
        for (Map.Entry m : entrySet) {
            if ((Integer) m.getValue() ==1) {
                System.out.println(m.getKey() + "\t" + m.getValue());
            }
        }
    }

}
```

```
Enter values in array
10
20
30
10
20
30
10
20
30
50
Display occurence of every element
50      1
```

Example: Implement following scenario using Map




```
LinkedHashMap   map = new LinkedHashMap();

ArrayList al = new ArrayList();
al.add("Virat");
al.add("Rohit");
al.add("Dhoni");

ArrayList al1 = new ArrayList();
al1.add("Steve");
al1.add("Warnar");
al1.add("Maxwell");

map.put("India",al);
map.put("Aus",al1);

Set<Map.Entry>  set = map.entrySet();

for(Map.Entry  e:set)
{   Object obj = e.getKey();
    Object val=e.getValue();
    ArrayList al=(ArrayList)obj;
   for( Object v:al)
    { System.out.println(v);
    }
}
```

Source code above screenshot

```
package org.techhub.map;
import java.util.*;
public class TourApplication {
	public static void main(String[] args)
	{	LinkedHashMap map = new LinkedHashMap();

			ArrayList al = new ArrayList();
			al.add("Virat");
			al.add("dhoni");
			al.add("rohit");

			ArrayList al1 = new ArrayList();
			al1.add("Steve");
			al1.add("Warnar");
			al1.add("Maxwell");

			map.put("India", al);
			map.put("Aus", al1);

			Set<Map.Entry> set=map.entrySet();
			for(Map.Entry m:set) {
				Object key=m.getKey();

		System.out.println("============"+key+"==================");
					Object value=m.getValue();
					ArrayList a=(ArrayList)value;
					for(Object val:a) {
						System.out.println(val);
```

```
                        }
                }
        }
}
```

Example: MAp within collection with user defined objects

```
package org.techhub.map;

import java.util.*;

class Student {
        private int id;

        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        private String name;
        public Student() {

        }
        public Student(int id, String name) {
                this.id = id;
                this.name = name;
        }

}

public class DeptMapApplication {
        public static void main(String[] args) {

                LinkedHashMap dept = new LinkedHashMap();

                        Student s1= new Student(1,"ABC");
                        Student s2= new Student(2,"PQR");
```

```
                ArrayList first=new ArrayList();
                first.add(s1);
                first.add(s2);

                Student s3 = new Student(1,"XYZ");
                Student s4 = new Student(2,"MNO");

                ArrayList second = new ArrayList();
                second.add(s3);
                second.add(s4);


                dept.put("FY",first);
                dept.put("SY", second);

                Set<Map.Entry> entrySet=dept.entrySet();

                for(Map.Entry m:entrySet) {
                        Object key= m.getKey();
                        System.out.println("Class Name :"+key);

        System.out.println("==================================
======");
                        Object val=m.getValue();
                        ArrayList a=(ArrayList)val;
                        System.out.println("ID\tNAME");
                        for(Object o:a) {
                                Student s=(Student)o;

        System.out.println(s.getId()+"\t"+s.getName());
                        }
                }
        }

}
```

Output

```
Class Name :FY
=========================================
ID      NAME
1       ABC
2       PQR
Class Name :SY
=========================================
ID      NAME
1       XYZ
2       MNO
```

Generics

Generics is concept launch by JAVA in JDK 1.5 version and which is used for avoid ClassCastException at run time

Q. What is ClassCastException and when does it occur?

ClassCastException occur when we try to perform referential conversion And it occur when we want to convert object in to specified class but if we different type of class at program runtime then casting not possible so JVM generate ClassCastException to us at run time

Suppose consider we are working with Collection framework and we try to store different types of data in collection then every element added in collection as Object type and when we try to retrieve data from collection then we get data in the form of Object class and we cannot perform direct operation on Object class then we need to cast in its original type then there is possibility of ClassCastException shown in following code

```
1  package org.techhub.map;
2  import java.util.*;
3  public class TestGenApp {
4      public static void main(String[] args) {
5
6          ArrayList al = new ArrayList();
7          al.add(100);
8          al.add(200);
9          al.add(300);
10         al.add(400);
11         al.add(new java.util.Date());
12         al.add("Good");
13         al.add(500);
14         int sum=0;
15         for(Object obj:al) {
16             sum=sum+(Integer)obj;
17         }
18         System.out.println(sum);
19     }
20
21 }
22
```

```
<terminated> TestGenApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (01-A
Exception in thread "main" java.lang.ClassCastException: class java.util.Date cannot be cas
        at org.techhub.map.TestGenApp.main(TestGenApp.java:16)
```

Note: if we think about left hand side code we have ArrayList which contain integer data as well as date and strig objects also and we fetch data from ArrayList using Object format and convert in integer so here first four 100 200 300 400 converted properly integer because they origin type is integer but we have fifth which is originally type of Date and we try to convert in integer so we get Runtime exception name as ClassCastException

Note: if we want to solve above problem we have two solutions

1. **Use instanceof operator :** instanceof operator is used for check reference belongs from particular class or not if reference is belong from particular class return true otherwise return false.

Example with source code

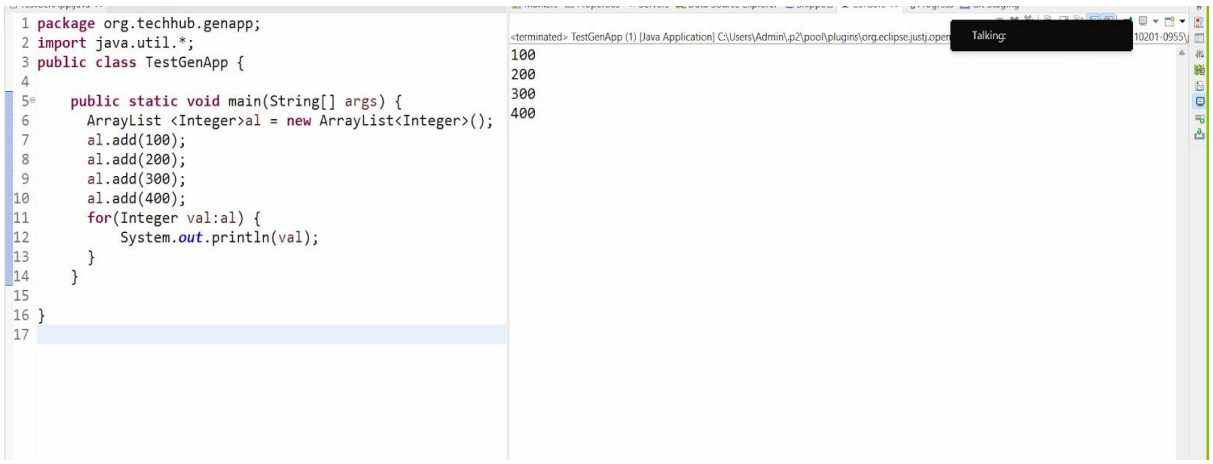
```
package org.techhub.map;
import java.util.*;
public class TestGenApp {
	public static void main(String[] args) {

		ArrayList al = new ArrayList ();
		al.add(100);
		al.add(200);
		al.add(300);
		al.add(400);
		al.add(new java.util.Date());
		al.add("Good");
		al.add(500);
		int sum=0;
		for(Integer  obj:al) {
			if(obj instanceof Integer) {
			sum=sum+(Integer)obj;
			}
		}
		System.out.println(sum);
	}

}
```

2. **Use Generics :** Generics is denoted by < >

Benefits of Generics

---

1. When we use generics with any Java Objects we not required to perform casting
2. We can customize our own classes and interfaces as generic classes and interfaces
3. Generic provide facility to us can work with more values at time by using wildcard generics and bounded generics




```
package org.techhub.genapp;
import java.util.*;
public class TestGenApp {

	public static void main(String[] args) {
	  ArrayList <Integer>al = new ArrayList<Integer>();
	  al.add(100);
	  al.add(200);
	  al.add(300);
	  al.add(400);
	  for(Integer val:al) {
		System.out.println(val);
	  }
	}

}
```

Note: if we think about above code we can say we have Collection with integer type means in Above ArrayList we cannot store data other than integer data type
Means we lost the main feature of collection to store different types of data

How to store different type of data in collection by using Generics

---

If we want to different type of data in collection using Generics we have to create POJO class and store all data in POJO class and use POJO class name as Data type with generics

```
package org.techhub.genapp;
import java.util.*;
```

```
class Employee {
	private int id;
	private String name;
	private int sal;

	public Employee() {

	}

	public Employee(int id, String name, int sal) {
		this.id = id;
		this.name = name;
		this.sal = sal;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getSal() {
		return sal;
	}

	public void setSal(int sal) {
		this.sal = sal;
	}
}

public class TestGenApp {

	public static void main(String[] args) {
		ArrayList<Employee> al = new ArrayList<Employee>();
```

```
		Employee emp1 = new Employee(1, "ABC", 10000);
		Employee emp2 = new Employee(2, "PQR", 20000);
		Employee emp3 = new Employee(3, "STV", 30000);
		al.add(emp1);
		al.add(emp2);
		al.add(emp3);

		Iterator<Employee> i = al.iterator();
		while (i.hasNext()) {
			Employee e = i.next();
	System.out.println(e.getId() + "\t" + e.getName() + "\t" + e.getSal());
		}

	}
}
```

How to create user defined class and user define interface as generics

---

If we want to create user defined class as Generics or interface as Generics class we have some inbuilt notation classes provided by java to us

E - E stands generic elements means we can say when we pass E as parameter in method or with class means method can accept any kind of data but we can restrict that method for work with particular type of data
T - T Stands generic type means when we have some data which may be change its type in implementer classes or some other place at run time then we can use T as Generic notations

K - K stands Generics key means when we want to create then we can use key as generic value
V - V stands for Generic value

Create user define class as Generic class

```java
package org.techhub.genapp;
import java.util.*;
class Test<E>{

    void add(E val) {
        System.out.println(val);
    }
}
public class TestApp {

    public static void main(String[] args) {


        Test <Integer>t=new Test<Integer>();
        t.add(100);
        t.add(200);
        t.add(300);


        Test <String>t1= new Test<String>();
        t1.add("Good");

    }
}
```

Talking:

Note: if we think about left hand side we have class Test<E> this indicate it is generic class means we can use generic notation with Test class object at run time can decide parameter type

void add(E val): this method indicate we can store any kind of data or accept kind of parameter but when we use generic notation with object then method can work according to object parameter

Test <Integer> t = new Test<Integer>() this method indicate t can work with integer type of data means when t call any method from Test class then method parameter is type of integer or method can return type of integer value

Test <String> t1 = new Test<String>() : here t1 indicate can work with string type of data means when we call any method using t1 then method can accept string type of parameter or method can return string type of parameter

User define interface with generics

```java
package org.techhub.genapp;
interface Area<T>{
      void setRadius(T r);
}
class Circle<E> implements Area<Integer>{
    @Override
    public void setRadius(Integer r) {
        System.out.println("Integer r "+r);
    }
}
class Cirm<E> implements Area<Double>{
    @Override
    public void setRadius(Double r) {
        System.out.println("Double r  "+r);
    }
}
public class InterfaceWithGenApp {

    public static void main(String[] args) {
      Circle <Integer> c = new Circle<Integer>();
      c.setRadius(5);
      Cirm <Double> cm = new Cirm<Double>();
      cm.setRadius(10.6);
    }

}
```

```
<terminated> InterfaceWithGenApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.jus
Integer r 5
Double r  10.6
```

Talking: Adinath Giri

Example with source code

```
package org.techhub.genapp;
interface Area<T>{
        void setRadius(T r);
}
class Circle<E> implements Area<Integer>{
        @Override
        public void setRadius(Integer r) {
                System.out.println("Integer r "+r);
        }
}
class Cirm<E> implements Area<Double>{
        @Override
        public void setRadius(Double r) {
                System.out.println("Double r  "+r);
        }
}
```

```
public class InterfaceWithGenApp {

        public static void main(String[] args) {
          Circle <Integer> c = new Circle<Integer>();
          c.setRadius(5);
          Cirm <Double> cm = new Cirm<Double>();
          cm.setRadius(10.6);
        }
}
```

WildCard Generics

Wildcard generics means generics work with any value with any data type and it is denoted by using ?
Normally wildcard generics use by developer when we pass collection as parameter value

There are two types of wild generics

---

1. **Unbounded wildcard generics :** unbounded generics means generics without any restrictions called as unbounded generis

Example:

```
1  package org.techhub.genapp;
2  import java.util.*;
3  class ABC{
4      void accept(List<?> list) {
5          for(Object obj:list) {
6              System.out.println(obj);
7          }
8      }
9  }
10 public class WildCardGenApp {
11     public static void main(String[] args) {
12         ABC a1 = new ABC();
13         System.out.println("List with integer data type");
14         List<Integer> list = Arrays.asList(10,20,30,40,50,60);
15         a1.accept(list);
16         System.out.println("List with float data type");
17         List<Float> list1=Arrays.asList(5.4f,2.3f,5.5f);
18         a1.accept(list1);
19     }
20 }
21
```

```
<terminated> WildCardGenApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20
List with integer data type
10
20
30
40
50
60
List with float data type
5.4
2.3
5.5
```

2. **Bounded wildcard generics :** bounded wildcard generics means we can give some restrictions with generics called as bounded generics

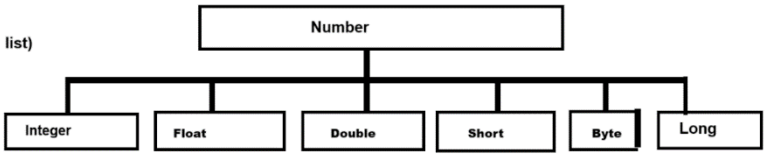
There are two types of bounded generics

---

a. **Upper bound generics:** upper bound generics can be specified by using < ? extends superclass name>

If we specify upper bound generics means we can use any child class name as replacement of ? mark

```
class  TestUpperBound
{
   void acceptNumberList(List <? extends Number>  list)
   {   for(Number  n: list)
       {
          System.out.println(n);
       }
   }
}

public class TestUpperBoundApp
{  public static void main(String x[])
   {
       TestUpperBound  tub =new TestUpperBound();
        List<Integer> list = Arrays.asList(10,20,30,40,50);
        tub.acceptNumberList(list);

        List<Float> list1 = Arrays.asList(5.4f,2.5f,3.5f);
         tub.acceptNumberList(list1);
   }
}
```



b. Lower bound generics : lower bound generics denoted by using <? super chiclassname> means here ? can replace parent class reference

```
package org.techhub.genapp;

import java.util.*;

class A {
	void show() {
		System.out.println("I am A method");
	}
}

class B extends A {
	void display() {
		System.out.println("I am display method");
	}
}

class C {
	void accept(List<? super B> list) {
		for (Object a1 : list) {
			((A) a1).show();
		}
	}
}

public class LowerBoundApp {
	public static void main(String x[]) {
		C c1 = new C();
		A a1 = new A();
		A a2 = new A();
```

```
                List<A> list = new ArrayList<A>();
                list.add(a1);
                list.add(a2);
                c1.accept(list);
            }
        }
```

Object class and its method

_____

Q. What is an Object class?

_____

Object class is parent of all classes in JAVA and it is a member of java.lang package and java.lang is a default package java means no need to import it.

Q. Why is the Object class the parent of every class in JAVA?

_____

Object class contains some inbuilt methods which are required to perform daily or regular operation with objects so developers can override methods provided by Object class and perform operations.

Methods of Object class

_____

boolean equals(Object): this method help us compare two object with each other using its content and if objects are equal return true otherwise return false

int hashCode(): this method help us generate the user defined hashcode and internally equals() and hashcode() method has contract

String toString(): this method can convert user defined object in to string format

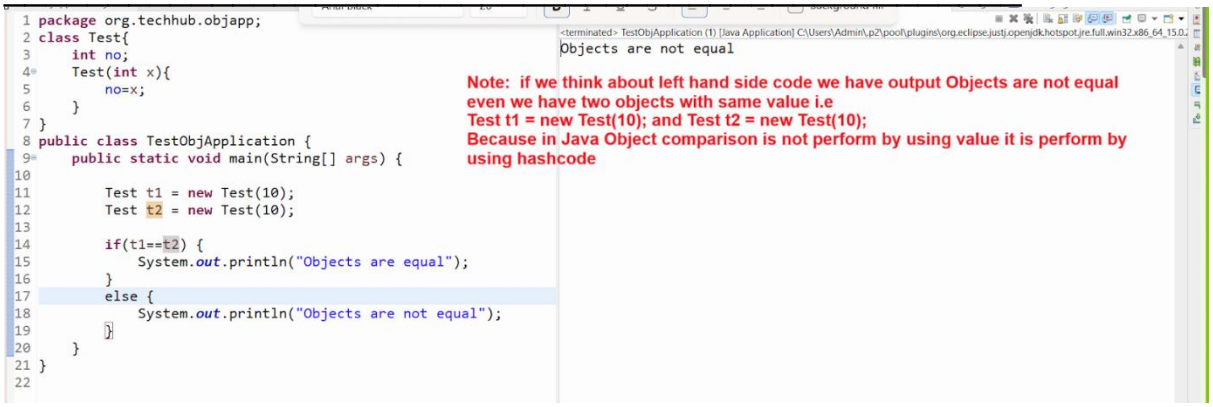
Object clone(): this method can perform object cloning with user defined data types.

Class getClass(): this method return reference of Class from java.lang

void wait(): this method can work with unconditional wait .
void wait(int): this method can work with conditional wait.
void notify():this method calls a waiting thread one by one and single thread at time.

void notifyAll(): this method call a all waited thread at time
void finalize(): this method calls automatically when we call System.gc() method means normally this method helps us to perform garbage collection or resource cleaning purposes.
static{
} **:** static block is also part of Object class

Example

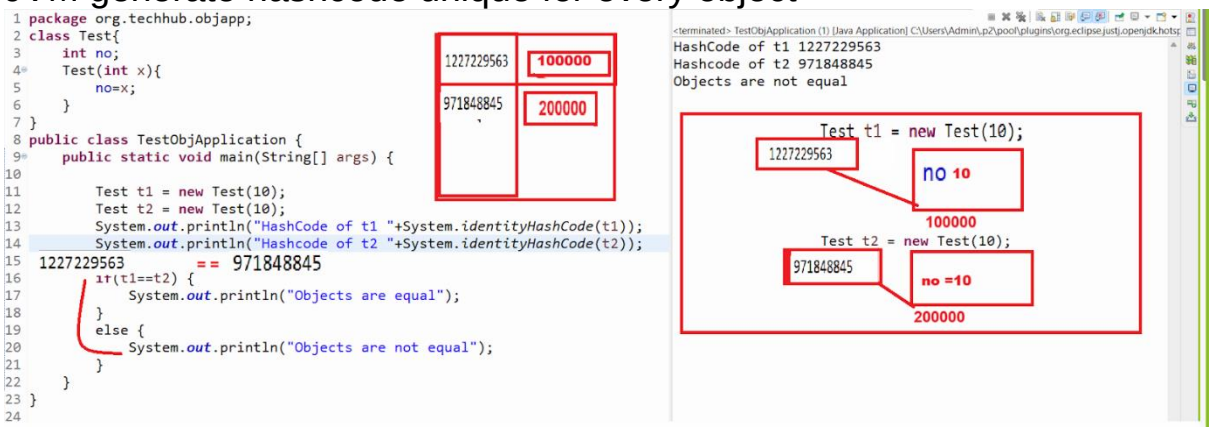


Q. What is hashcode?

Hashcode is unique integer number provide by JVM to every object in memory and it is not actual address it is associated internally with memory address and if we want to check the hashcode generated by JVM then we have method name as System.identityHashCode()

How to hashcode associated with a memory address?

Internally JVM use hashcode as key and memory address as value but not provided access memory address and memory address is unique so JVM generate hashcode unique for every object



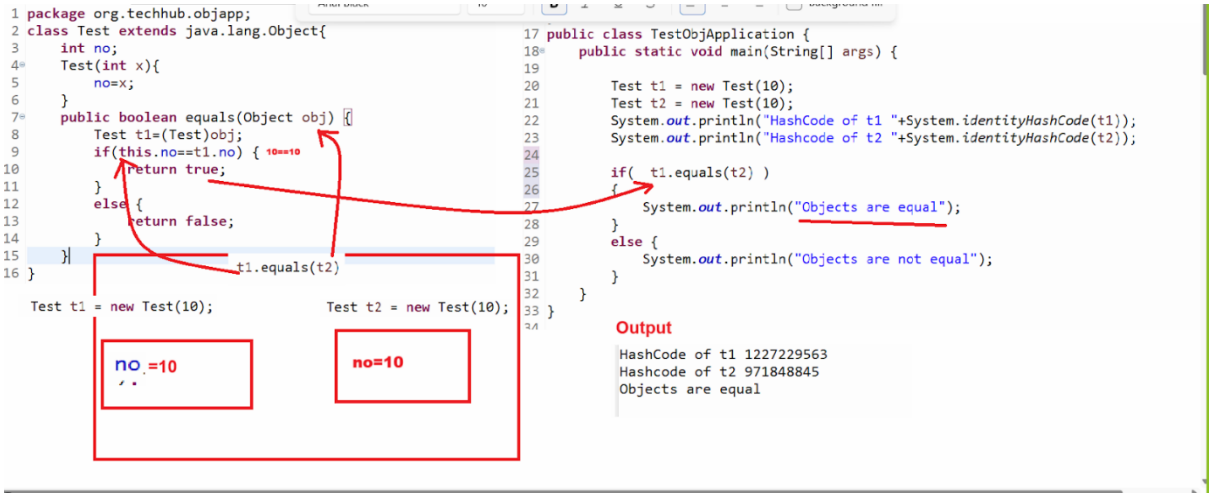
Note: if we think about above code we have reference t1 with hashcode

1227229563 and reference t2 with hashcode 971848845 means when JVM execute statement if(t1 == t2) means if(1227229563 ==971848845) here both hashcode are different so we get answer objects are not equal

Important points: JVM cannot generate same hashcode for two objects

Q. If JVM does not generate the same hashcode for two objects and if JVM considers two objects same when their hashcode code then how do they compare objects with each other in JAVA?

If we want to solve this problem java suggest us override two methods provided by Object class name as equals() and hashcode() and using equal method we can compare object content and if object contents are equal then return true otherwise return false and if two object contain same content then generate same hashcode for both object using hashcode() method provided by Object class and if objects are different then generate different hashcode for two objects using hashcode methods

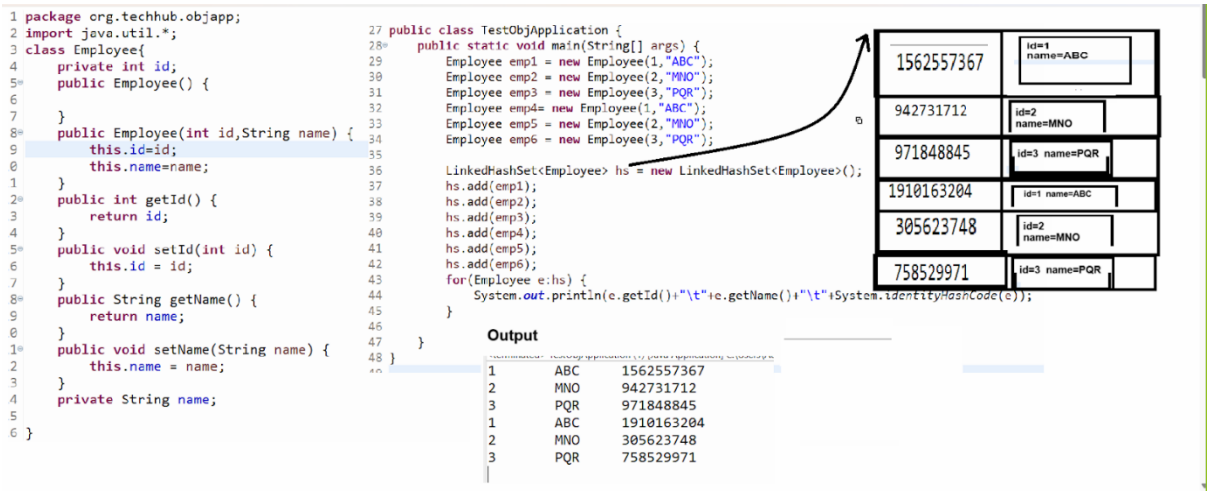


Note: if we think about above output we override only equals() method and we compare two objects and we get Objects are equals without overriding user defined hashCode method and here hashcode of two objects are different generated by JVM

Q. What is the purpose of a user-defined hashcode method or why does java suggest that users override their own hashcode method and generate the same hashcode when two objects are equal?

Note: yes we can compare two object with each other without using user defined hash code

But if we store your user-defined object in Set collection and not override hashcode() method with equals() method then set collection can store duplicate data.



```
package org.techhub.objapp;
import java.util.*;
class Employee{
    private int id;
    public Employee() {

    }
    public Employee(int id,String name) {
        this.id=id;
        this.name=name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    private String name;

}
```

```
public class TestObjApplication {
    public static void main(String[] args) {
        Employee emp1 = new Employee(1,"ABC");
        Employee emp2 = new Employee(2,"MNO");
        Employee emp3 = new Employee(3,"PQR");
        Employee emp4= new Employee(1,"ABC");
        Employee emp5 = new Employee(2,"MNO");
        Employee emp6 = new Employee(3,"PQR");

        LinkedHashSet<Employee> hs = new LinkedHashSet<Employee>();
        hs.add(emp1);
        hs.add(emp2);
        hs.add(emp3);
        hs.add(emp4);
        hs.add(emp5);
        hs.add(emp6);
        for(Employee e:hs) {
            System.out.println(e.getId()+"\t"+e.getName()+"\t"+System.identityHashCode(e));
        }

    }
}
```


Output

```
1       ABC     1562557367
2       MNO     942731712
3       PQR     971848845
1       ABC     1910163204
2       MNO     305623748
3       PQR     758529971
```

Note: if we think about above code we store 6 objects in LinkedHashSet with duplicated data and normally we said set not store duplicate data but in above set contain duplicated data

Q. How to store duplicate data?

When we store data in set the set collection compare the hashcode of data if set found unique hashcode of data then set not allow data but if we store user define objects in set collection and override equals() and hashcode() method in user defined class then set compare the hashcode for that object generated JVM and JVM not generate same hashcode for two different objects even object contain same data so set never find duplicates in this case so set allow duplicate data
So for avoid this problem Java Suggest us override equals() and hashcode() method in class or in user defined class and generate same user hashcode if two objects are equal so set collection use user hashcode for comparison if set found duplicate user hashcode then not store data otherwise store data shown in following code.

Example with source code

```
package org.techhub.objapp;
import java.util.*;
class Employee{
        private int id;
        public Employee() {

        }
```

```
	public Employee(int id,String name) {
		this.id=id;
		this.name=name;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	private String name;

	public boolean equals(Object obj) {
		Employee e=(Employee)obj;
		if(this.id==e.id && this.name.equals(e.name)) {
			return true;
		}
		else {
			return false;
		}
	}
	public int hashCode() {
		return id*10000;
	}
}
public class TestObjApplication {
	public static void main(String[] args) {
		Employee emp1 = new Employee(1,"ABC");
		Employee emp2 = new Employee(2,"MNO");
		Employee emp3 = new Employee(3,"PQR");
		Employee emp4= new Employee(1,"ABC");
		Employee emp5 = new Employee(2,"MNO");
		Employee emp6 = new Employee(3,"PQR");
		LinkedHashSet<Employee> hs = new
LinkedHashSet<Employee>();
		hs.add(emp1);
		hs.add(emp2);
```

```
            hs.add(emp3);
            hs.add(emp4);
            hs.add(emp5);
            hs.add(emp6);
            for(Employee e:hs) {

        System.out.println(e.getId()+"\t"+e.getName()+"\t"+System.identity
HashCode(e));
            }
        }
}
```

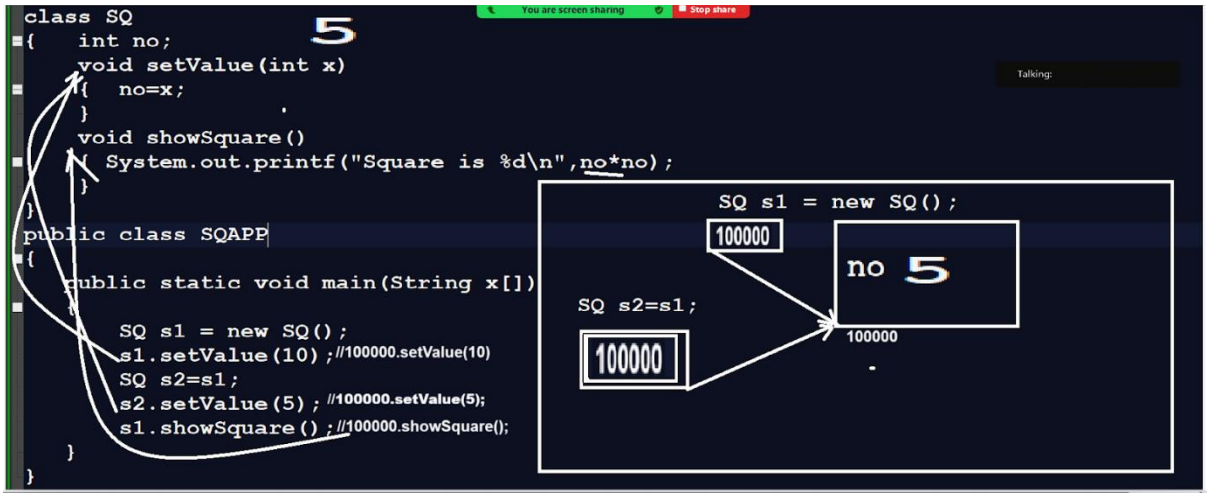
Object cloning concept using JAVA

Q. What is an Object clone?

If we create a duplicate copy of an object called an Object clone.

Q. Why do we need to perform object cloning?

Some time we apply more than one reference on single object and if we want to perform change on object using any one reference the object content may be lost or vanish or update if we want to solve this problem so java suggest we can use object cloning approach so we can persist the object previous content as well as we have one more copy of same object we can use as it is or we can change as per your requirement so original copy of object not hamper by modification



Note: if we think about above code we have main method with code

SQ s1 = new SQ() means we create object of SQ class and consider address of object is 100000 and store this address in s1 reference and we have statement s1.setValue(10) means 100000.setValue(10) means we call function using object address is 100000 and store 10 value in no variable which is present on 100000 address and we have one more statement SQ s2=s1 means we initialize s1 address in s2 reference means s1 and s2 points to same memory location i.e 100000 and we have statement s2.setValue(5) means 100000.setValue(5) so this 5 value get override in no which is present on 100000 location so we lost the previous content of no on address 100000 and no is part of object so lost the object previous content and when we call statement s1.showSquare() means 100000.shwSquare() so we get answer Square is 25 i.e updated value
So conclusion is we lost the object historical data so if we want to solve this problem java suggest us use object cloning concept

How to implement the object cloning practically

If we want to implement object cloning practically we have clone() method provided by Object class

Steps to implement the object cloning

---

1. **Create user define class and implements Cloneable interface**

   ```
   class SQ implements Cloneable
   {   int no;
       void setValue(int x)
           {  no=x;
           }
           void showSquare()
           { System.out.printf("Square is %d\n",no*no);
           }
   }
   ```

   Important point:
   Cloneable is a member of java.lang package and it is marker interface in JAVA

   Q. What is the marker interface?

   Marker interface means a interface which does not contain any method or state means blank interface but JVM provide special run

time environment to them and marker interface provide some special information to class where they implement and JVM behave object of that class according to information provided by Marker interface

Example of marker interfaces

a. Cloneable b. Serializable etc

Example

```
Compiled from "Cloneable.java"
public interface java.lang.Cloneable {
}
```

Or

```
C:\Program Files\Java\jdk1.8.0_291\bin>javap java.io.Serializable
Compiled from "Serializable.java"
public interface java.io.Serializable {
}
```

2. Create a Factory method and call clone() in it and return reference to the new object created by the clone method .

Note: if we think about above code we create new object by using clone() method but we create object by using new then also we get same answer like as clone

Q. What is the difference between object creation using new and using clone()?

1. When we create object by using new keyword then constructor get executed and when we create object using clone then constructor not executed
2. When we use new then logic of constructor get executed if written by developer but by clone logic from constructor not executed written by developer
3. When we use new keyword then instance variable and static initialize according to default values provided by java means new construction of memory but when we create object by clone then existing object content copied in new object means there is no new construction of memory from initialization just constructor memory by using existing content

Example with source code

```
class SQ  implements Cloneable
{   int no;
   static int count;
    SQ(){
                ++count;
                System.out.println("Constructor call number of
time  "+count);
        }
    void setValue(int x)
        {  no=x;
        }
        void showSquare()
        { System.out.printf("Square is %d\n",no*no);
        }
        public SQ getSQClone()throws CloneNotSupportedException
        {     Object obj = this.clone();
                return (SQ)obj;
        }
}
public class SQAPP
{
   public static void main(String x[])throws Exception
   {
      SQ s1 = new SQ();
          s1.setValue(10);
          SQ s2=new SQ();
```

```
      s2.setValue(5);
          s1.showSquare();
          System.out.println("Hashcode of
s1  "+System.identityHashCode(s1));
          System.out.println("Hashcode of
s2  "+System.identityHashCode(s2));
  }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac SQAPP.java

C:\Program Files\Java\jdk1.8.0_291\bin>java SQAPP
Constructor call number of time  1
Constructor call number of time  2
Square is 100
Hashcode of s1  865113938
Hashcode of s2  1442407170

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Note: if we think about above code we execute constructor two times because we create two objects by using new keyword so constructor executed two times

Example object creation using clone method

```
class SQ  implements Cloneable
{   int no;
   static int count;
    SQ(){
                ++count;
        System.out.println("Constructor call number of time  "+count);
        }
    void setValue(int x)
        {  no=x;
        }
        void showSquare()
        { System.out.printf("Square is %d\n",no*no);
        }
        public SQ getSQClone()throws CloneNotSupportedException
        {     Object obj = this.clone();
              return (SQ)obj;
        }
}
public class SQAPP
{
```

```
  public static void main(String x[])throws Exception
  {
     SQ s1 = new SQ();
         s1.setValue(10);
         SQ s2=s1.getSQClone();
     s2.setValue(5);
         s1.showSquare();
         System.out.println("Hashcode of
s1  "+System.identityHashCode(s1));
         System.out.println("Hashcode of
s2  "+System.identityHashCode(s2));
  }
}
```

Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac SQAPP.java

C:\Program Files\Java\jdk1.8.0_291\bin>java SQAPP
Constructor call number of time  1
Square is 100
Hashcode of s1  865113938
Hashcode of s2  1442407170

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Static and instance block

Static block is a member of object class and instance block is user defined block define by user in class

Q. What is the difference between instance block and static block?

1. **Static blocks are executed very first in application even before the main method**

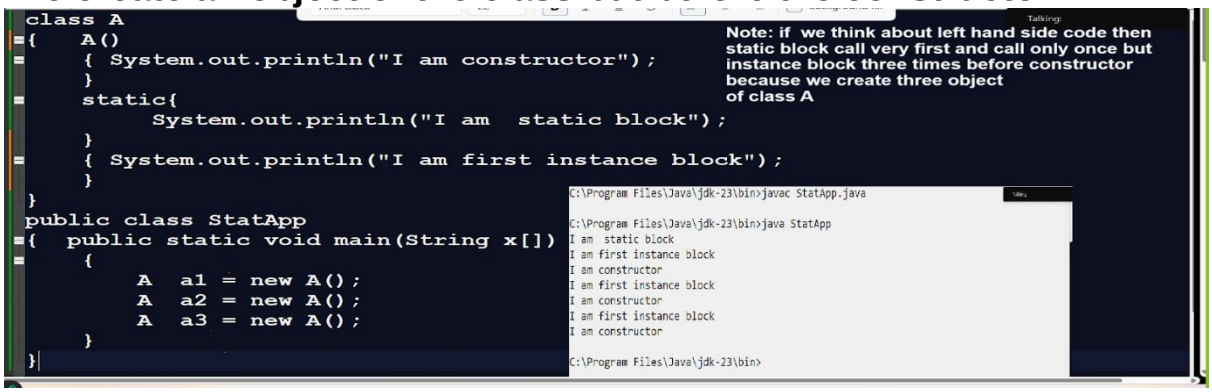
```
public class StatApp
{
    static{
      System.out.println("I am static block");
    }
    public static void main(String x[])
    {
        System.out.println("I am main method");
    }
}
```

Note: if we think about left hand side code static block execute first before main method we can say we can execute code before calling method by using static block.

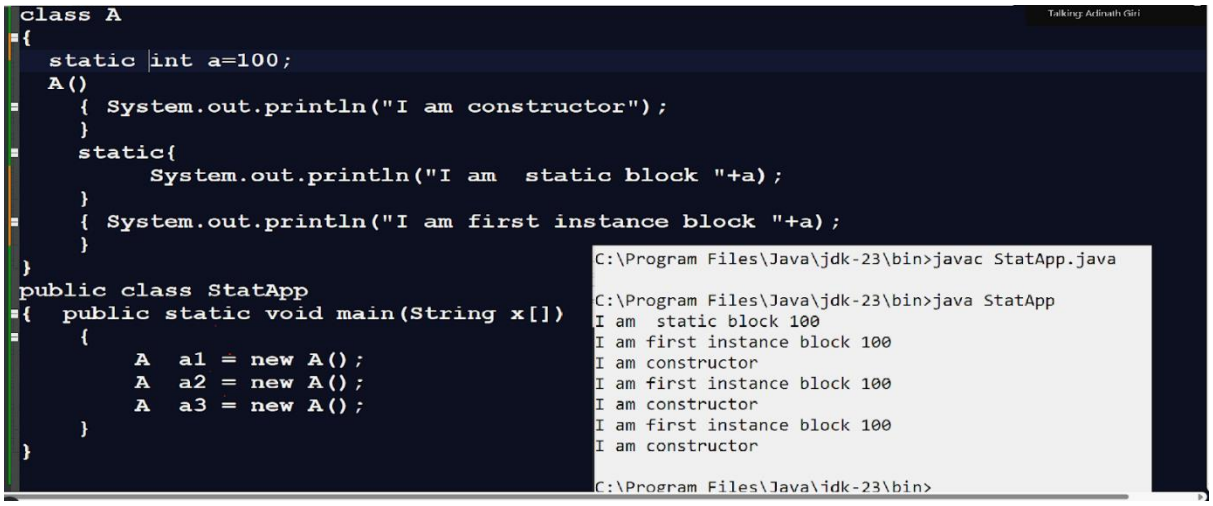
Output

```
C:\Program Files\Java\jdk-23\bin>javac StatApp.java

C:\Program Files\Java\jdk-23\bin>java StatApp
I am static block
I am main method
```

2. Static blocks execute only once or the first time when we create an object of the class but instance block calls every time when we create an object of the class but before the constructor.



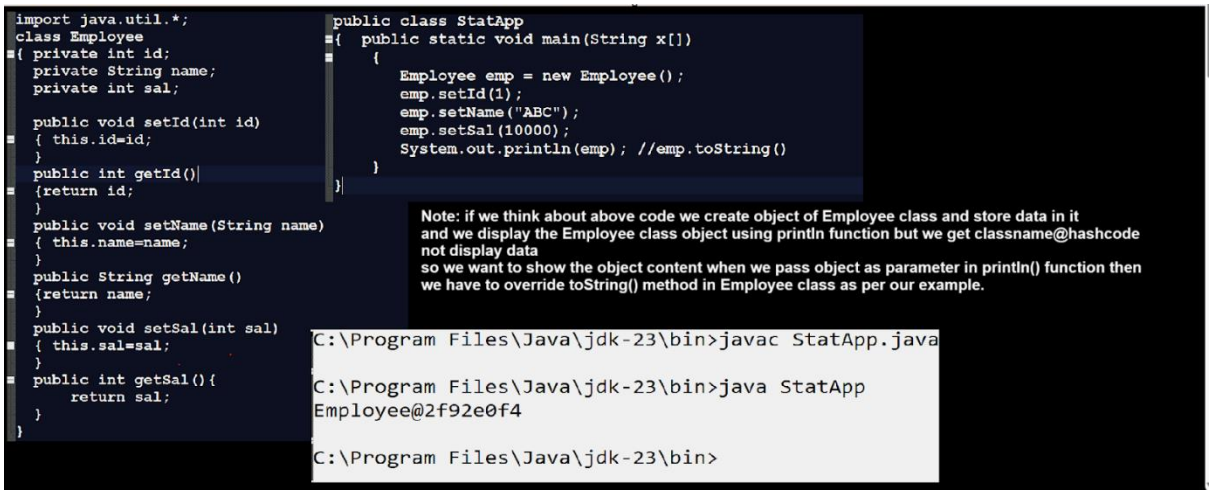
3. Static block can use only static variable but instance block can use static and non static variable or instance variable



String toString(): this method is used for convert java object in to string format

Normally we use this method following purpose

1. When we want to convert an object to string format.
2. When we want display object content when we pass object as parameter in println method
3. When we store object in collection and display the object content when we print collection
   Etc



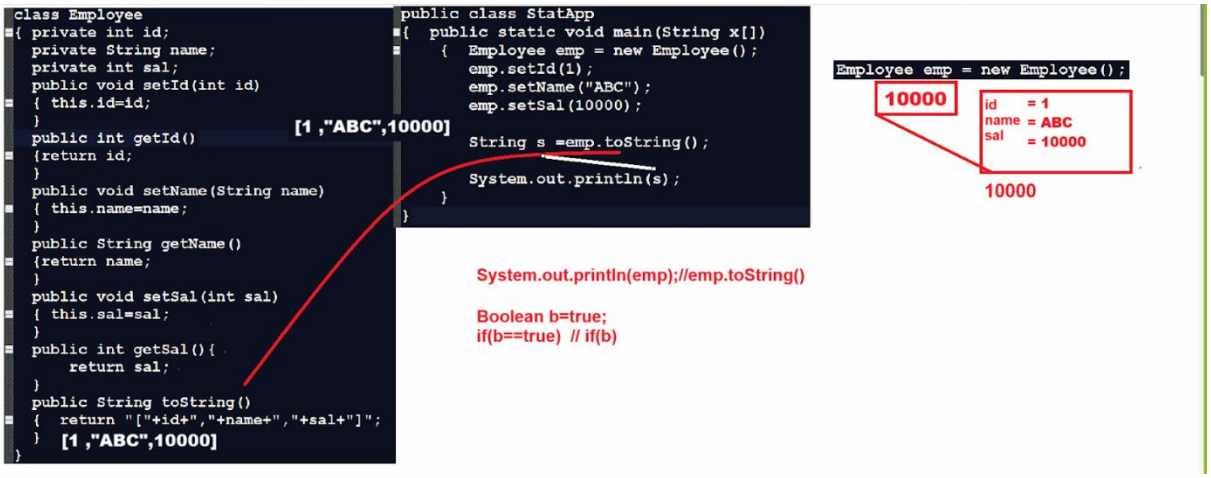
Now we want to override the toString() method in Employee class.

Example: WAP to create class name as Employee and store 3 employee object in ArrayList and display it without iteration

```
import java.util.*;
class Employee
{ private int id;
  private String name;
  private int sal;
  public void setId(int id)
  { this.id=id;
  }
  public int getId()
  {return id;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName()
```

```
  {return name;
  }
  public void setSal(int sal)
  { this.sal=sal;
  }
  public int getSal(){
          return sal;
  }
  public String toString()
  {  return "["+id+","+name+","+sal+"]";
  }
}
public class StatApp
{  public static void main(String x[])
        {  Employee emp = new Employee();
           emp.setId(1);
           emp.setName("ABC");
           emp.setSal(10000);
           Employee emp1 = new Employee();
           emp1.setId(1);
           emp1.setName("ABC");
           emp1.setSal(10000);
           Employee emp2 = new Employee();
           emp2.setId(1);
           emp2.setName("ABC");
           emp2.setSal(10000);
           ArrayList al = new ArrayList();
           al.add(emp);
           al.add(emp1);
           al.add(emp2);

           System.out.println(al);
        }
}
```

Example: WAP to create employee class and convert Employee object in to string



void finalize(): finalize() method is used for resource cleaning purpose means this method call automatically when we perform garbage collection manually using System.gc() method

Means when we delete any object from the heap section and if we want to perform any operation at the time of object of deletion then we can override the finalize() method of Object class and write logic in it.

```
import java.util.*;
class Employee
{ private int id;
  private String name;
  private int sal;
  public void setId(int id)
  { this.id=id;
  }
  public int getId()
  {return id;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName()
  {return name;
  }
  public void setSal(int sal)
  { this.sal=sal;
  }
  public int getSal(){
          return sal;
  }
  public String toString()
  {  return "["+id+","+name+","+sal+"]";
  }
```

```
  public void finalize()
  {  System.out.println("I am finalize method");
  }
}
public class StatApp
{  public static void main(String x[])
        {  Employee emp = new Employee();
           emp.setId(1);
           emp.setName("ABC");
           emp.setSal(10000);
           emp=null;
   System.gc(); //perform garbage collection
//means delete object from memory whose address was present in emp
//currently object not use by any other reference so JVM can delete
//object
//from memory called as garbage collection

        }
}
```

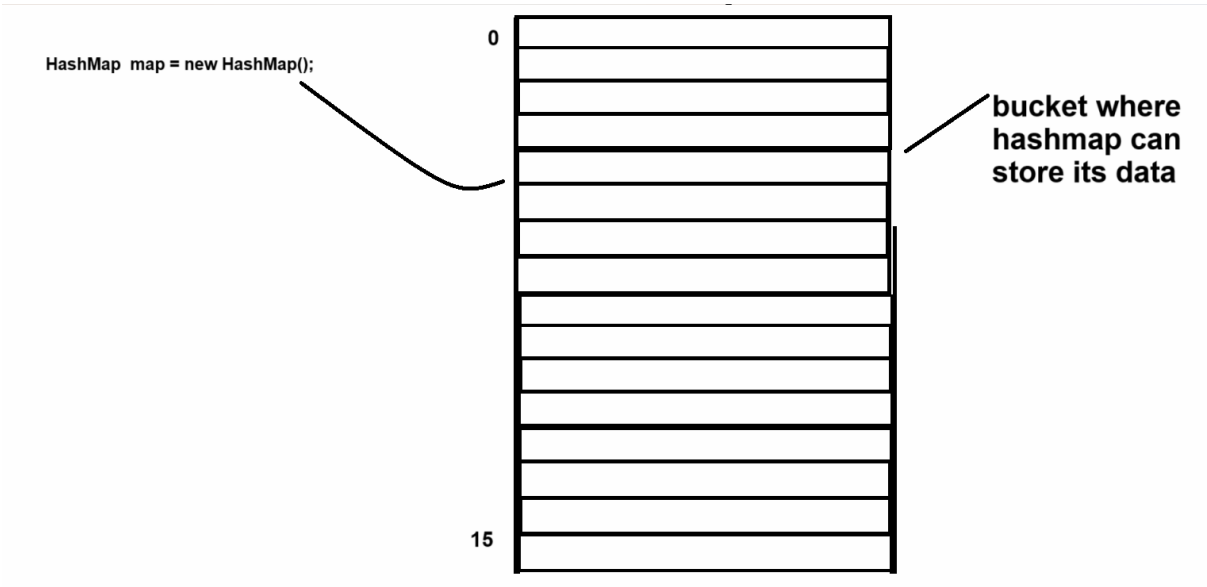
Internal working of HashMap

When we create object of HashMap like as HashMap<String> map=new HashMap<String>() then JVM create 16 bucket of slot object and you can store 16 values in hashmap and load factor of HashMap is 75% of his capacity then it double the existing capacity.
Means when we think about HashMap then internally the capacity 75% means 12 item and when we try to store 13 elements in hashmap then hashmap can increase its capacity from 16 to 32
Note: load factor 75% or 0.75 or ¾

Now we want to discuss about what is bucket and how data store in bucket

---

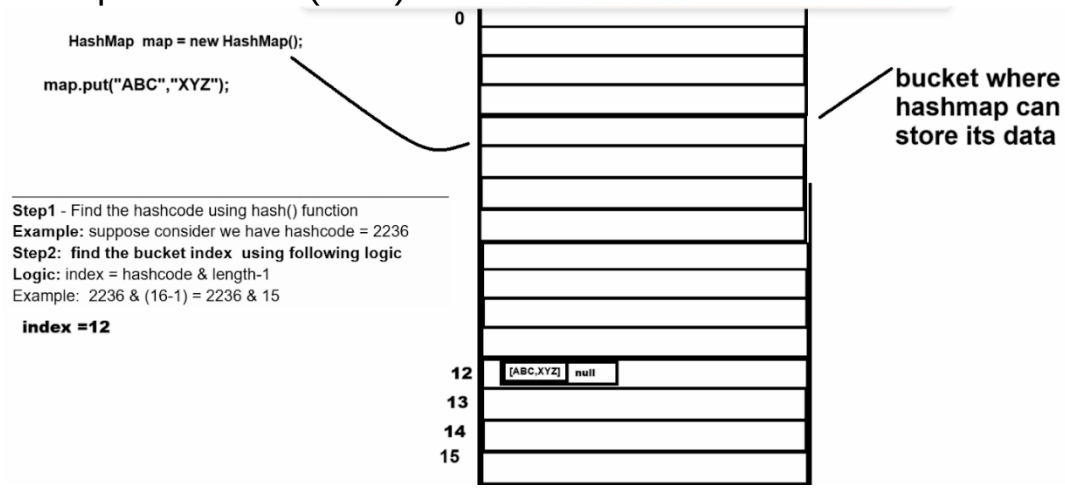
Bucket is internally array and bucket can store data in the form of LinkedList and LinkedList contain data in the form of node and address.



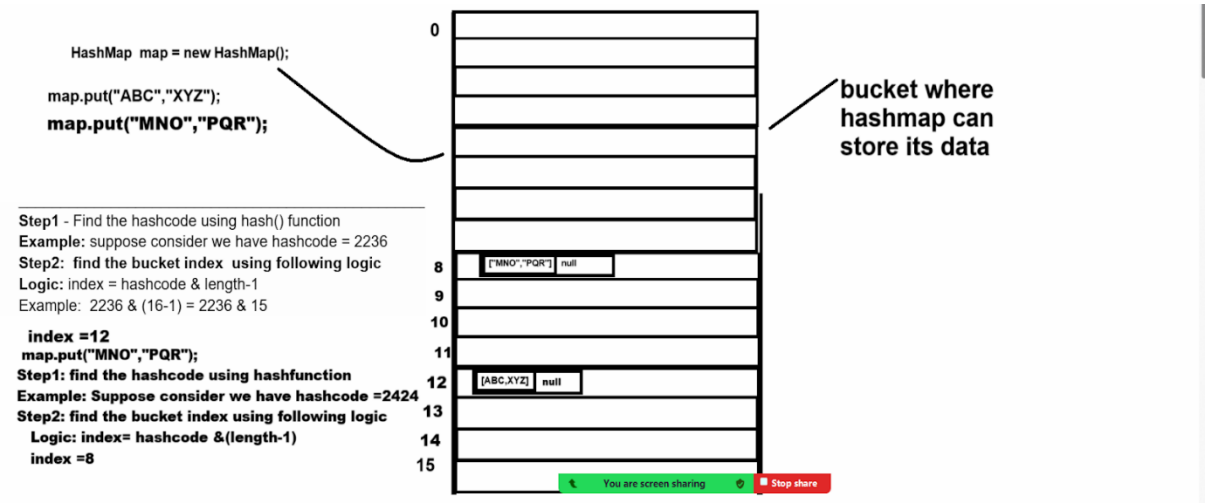
Suppose consider we are going to store data in map by using put() method like as

HashMap<String,String> map = new HashMap<String,String>();
map.put(“FB”,”A”);
Note: Here FB is key of String object
How to store data in HashMap internally

Step1 - Find the hashcode using hash() function
Example: suppose consider we have hashcode = 2236
Step2: find the bucket index using following logic
Logic: index = hashcode & length-1
Example: 2236 & (16-1) = 2236 & 15



Now we want to add one more element



Now we want to discuss about the concept of hash collision

Q. What is hash collision?

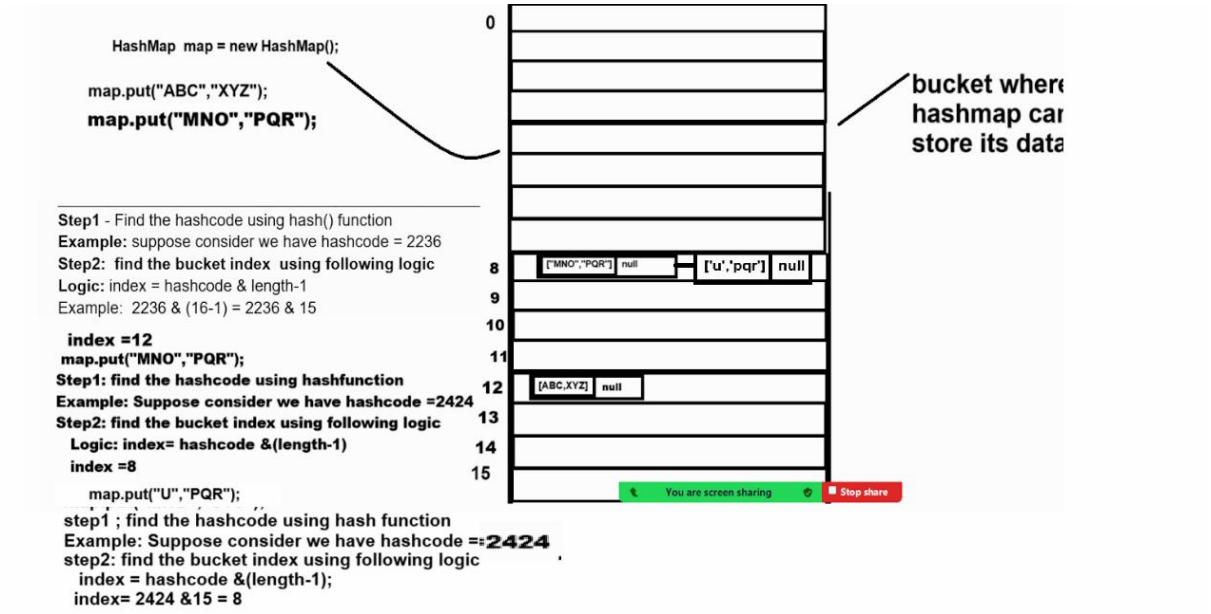
Collision occur when two different keys produce the same hash code then we need to store two keys in same bucket the JVM compare newly generated key with previous key if newly generate key is same as previously key then override newly generated key on previous key and if newly generated key not same with previous generated key then JVM create new node and store in same index and attach new node with previous node

Suppose consider we are going add new data in HashMap

How get() method work internally

---

get() method of hashmap return the value using key very fast map.get(“u”): so the time complexity of get() method is O(1)



Steps

Step1: The JVM find the hashcode of u is 2424
Step2: find the bucked index: 8
Step3: go the bucket index 8
Step4: get the value by using key

Note: HashMap normally store only one node in bucket but if there is hashcode collision then there is possibility to store more than one node in HashMap but if we store more than one in single bucket of the HashMap then there may be possibility to degrade performance
Because we need to travel all node or linked list in that bucket and compare key in every node
So it may degrade the performance of retrieving data so
Java 8 enhance this HashMap
Means Java 8 convert your linked list from bucket in to tree structure after certain threshold and tree structure known as treefy structure and this treefy structure known as red black tree and binary search tree self balancing

Q. What is the red black tree structure?

Red black tree is self balanced tree provide efficient insert/deletion and retrieval operation

Properties of Red black tree

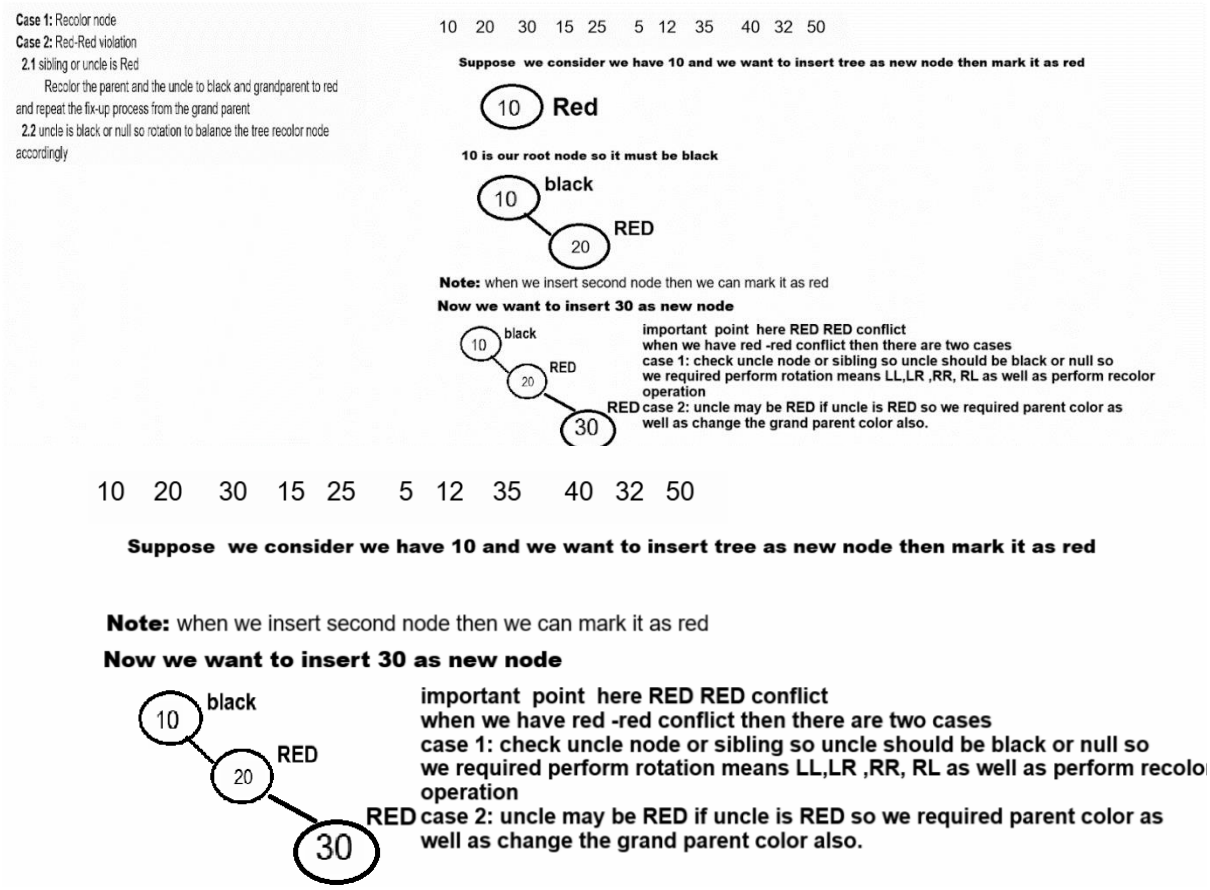
---

1. **Node color :** each node is either red or black
2. **Root Property:** Root is always black

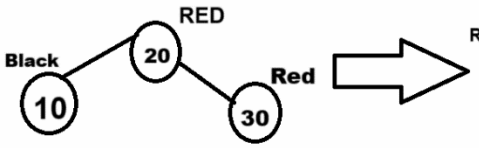
3. **Red Not Property :** red node cannot have red childern means two red nodes can be adjacent
4. **Black depth property :** every path from a node to its descendent leave must have the same number of black nodes.
5. **Leaf node :** all leaf node are black and do not store any data they are placeholders and to maintain tree structure

Example: suppose we have data and we want to perform a red black tree using that data.
10 20 30 15 25 5 12 35 40 32 50

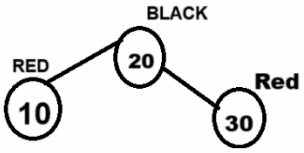
Case 1: Recolor node
Case 2: Red-Red violation
2.1 sibling or uncle is Red
Recolor the parent and the uncle to black and grandparent to red and repeat the fix-up process from the grand parent
2.2 uncle is black or null so rotation to balance the tree recolor node accordingly



Note: here we required to perform rotation from right to left so your tree look like as after rotations



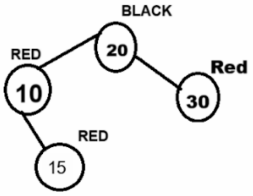
Recolor Tree after rotation



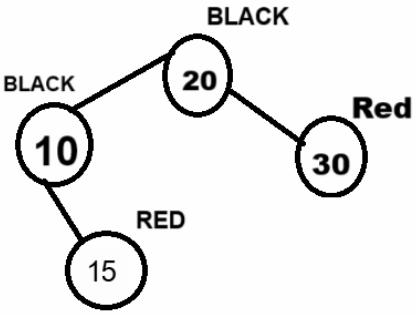
Case 1: Recolor node
Case 2: Rec-Red violation
2.1 sibling or uncle is Red
Recolor the parent and the uncle to black and grandparent to red and repeat the fix-up process from the grand parent
2.2 uncle is black or null so rotation to balance the tree recolor node accordingly

10 20 30 15 25 5 12 35 40 32 50

Suppose we consider we have 10 and we want to insert tree as new node then mark it as red

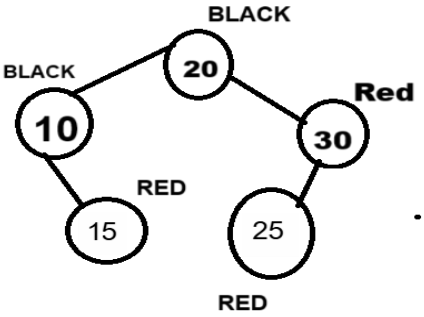


Note: when we insert 15 in Tree then 15 is less than 20 and greater than 10 so 15 should be right hand side of 10 and mark it as red so we have again red and red conflict as per the case 1 we required to find its uncle not so 30 is uncle of 15 and 30 is red not so here we not required rotation just need to perform recolor operation with parent node i.e 10 should be black

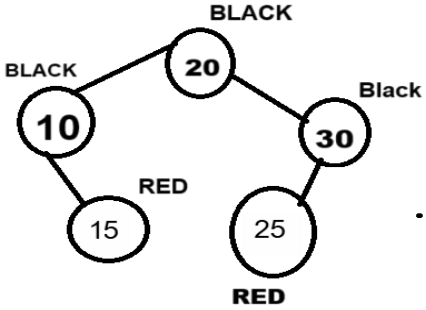


10 20 30 15 25 5 12 35 40 32 50

Suppose we consider we have 10 and we want to insert tree as new node then mark it as red

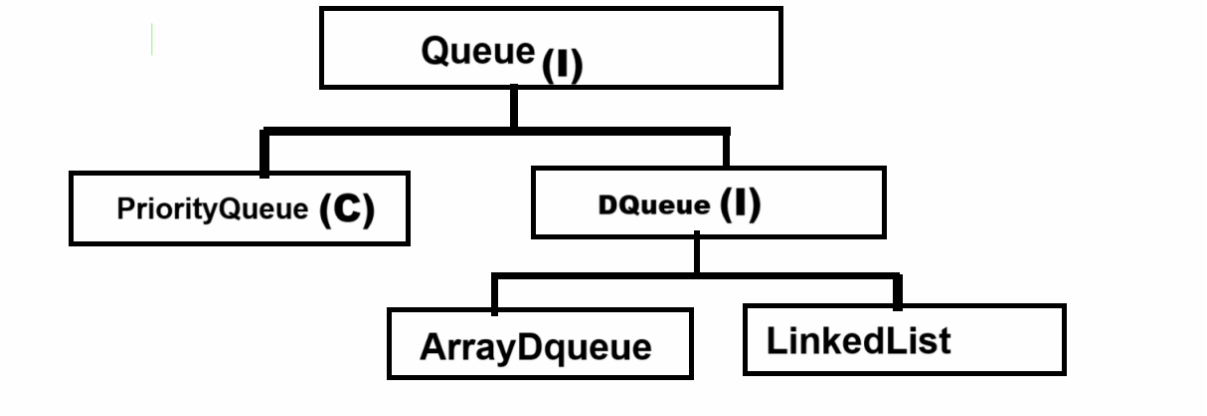


Note: we have again red and red conflict but 30 is not main root it is intermediate node so we required just recolor it



Queue

If we want to work with Queue we have to know the hierarchy of Queue

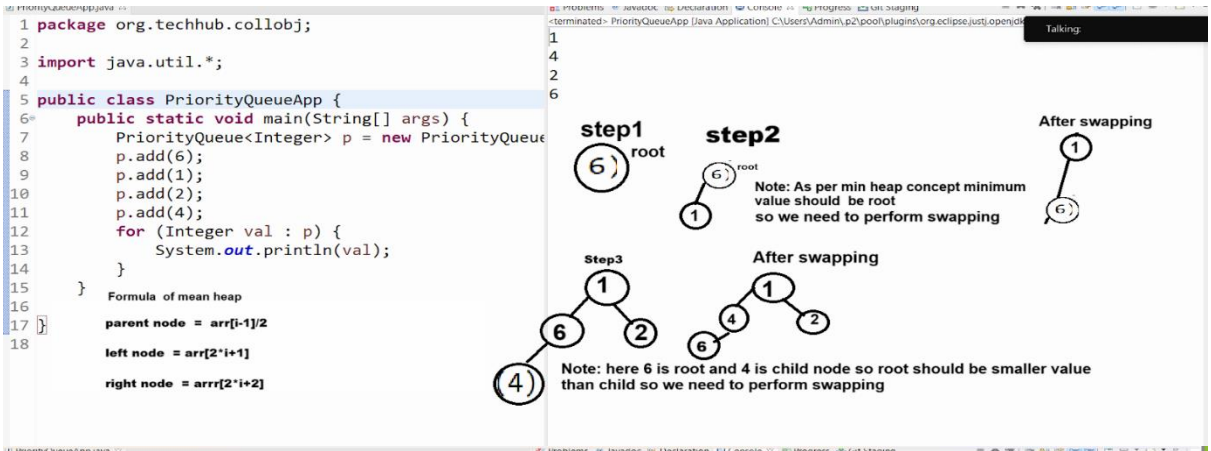


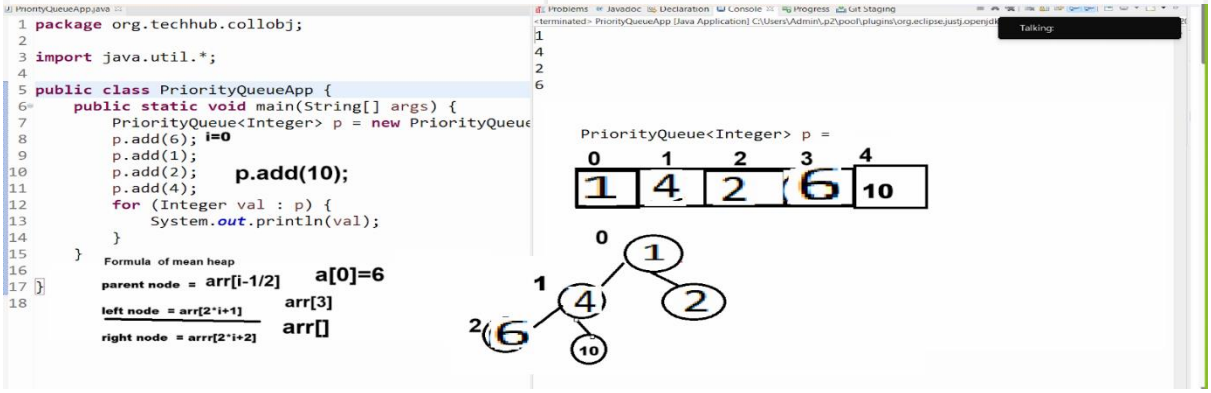
PriorityQueue : PriorityQueue of Java use Min Heap concept or Tree internally

Important points related with priority queue

---

1. **Minimum value should be root node**
2. **Add new node from left hand side**





If we want to organize your priority Queue as Max Heap concept then we have to use Comparator interface and its reverseOrder() method means your maximum node should be root

```java
package org.techhub.collobj;

import java.util.*;

public class PriorityQueueApp {
    public static void main(String[] args) {
        PriorityQueue<Integer> p = new PriorityQueue<Integer>(Comparator.reverseOrder());
        p.add(6);
        p.add(1);
        p.add(2);
        p.add(4);
        p.add(10);
        for (Integer val : p) {
            System.out.println(val);
        }
    }
}

```

```
10
6
2
1
4
```

DQueue : DQueue Stands for Double Ended Queue means we can insert or remove node from front as well as rear

```java
package org.techhub.collobj;

import java.util.*;

public class PriorityQueueApp {
    public static void main(String[] args) {
         Deque<Integer> d=new ArrayDeque<Integer>();
         d.add(10);
         d.add(20);
         d.addFirst(100);
         d.addLast(200);
         int value=d.remove();
         System.out.println("Deleted value is "+value);
         for(Integer val:d) {
             System.out.println(val);
         }
    }
}

```

```
Deleted value is 100
10
20
200
```