

List Collection

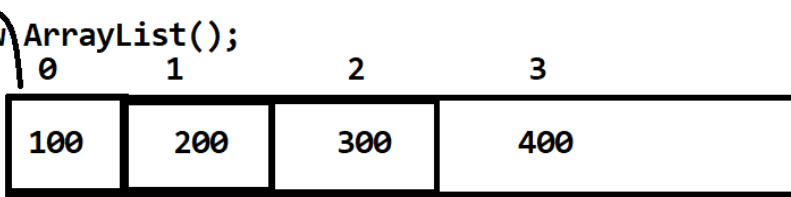
List Collection allows the **duplicated** data and can store infinite number of data and manage data by using **index technique** like as array.

Methods of List Collection or Interface

public abstract E get (int) : this method can retrieve data from List collection using its index.

Example

```
ArrayList al = new ArrayList();  
  
al.add(100);  
al.add(200);  
al.add(300);  
al.add(400);  
  
for(int i=0; i<al.size(); i++)  
{  
    Object obj = al.get(i);  
    System.out.println(obj);  
}
```



0	1	2	3
100	200	300	400

Code Description:

In above code when we execute the statement `for(int i=0; i<al.size();i++)` Here `al.size()` method return the size of collection as per our example it is 4 means statement work like as `for(int i=0;i<4;i++)`

When we execute the statement `al.get(i)` means first our `i` is 0 in for loop so statement like as `al.get(0)` means fetch data from 0th index of ArrayList when loop second time then `i` increase by 1 means 1 is less than 4 and statement get executed `al.get(1)` means fetch second position data from ArrayList and so on

public abstract E set(int, E) : this method is used for replace the element in collection on specified index.

```
ArrayList al = new ArrayList();
```

```
al.add(100);  
al.add(200);  
al.add(300);  
al.add(400);
```

0	1	2	3
100	200	300	400

Before Replacement ArrayList

```
al.set(2,600);
```

Here we replace the 300 value on second position of ArrayList

0	1	2	3
100	200	600	400

public abstract void add(int, E): this method can add the element on specified index in Collection and move the remaining element on next index.

```
ArrayList al = new ArrayList();
```

```
al.add(100);  
al.add(200);  
al.add(300);  
al.add(400);
```

0	1	2	3
100	200	300	400

Before inserting value

```
al.add(1,1000);
```

we added 1000 element on 1st index of ArrayList and shifted previous element by index position

0	1	2	3	4
100	1000	200	300	400

public abstract E remove(int): this method can remove the element using some specified index

public abstract int indexOf(java.lang.Object): this method can return the index of specified element in collection Normally we use this method for searching purpose in collection if element found return its index and if element not found return -1

```

ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
al.add(400);

2
int index = al.indexOf(300); //this method return the index of 300
                           i.e 2

```

0	1	2	3
100	200	300	400

Where we can use this method or scenario where we can use this method

1) Search the element from collection: `indexOf()` method return the index of particular element from collection and if element not found return -1 means when we found the -1 we can declare element not present in collection and if found any value other than -1 we can consider element present in collection.

```

ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
al.add(400);

Scanner xyz = new Scanner(System.in);
System.out.println("Enter the search value\n");
int value =xyz.nextInt();

int index = al.indexOf(value);
if(index!=-1)
{
    System.out.println("Element Found");
}
else
{
    System.out.println("Element Not Found");
}

```

0	1	2	3
100	200	300	400

If we think about above code then we have the for input `int value=xyz.nextInt()` and we provide input 200 then next statement get executed i.e `int index = al.indexOf(value);` means we pass 200 value of

indexOf() method e.g int index=al.indexOf(200) and in our example 200 having 1st index

then we have the next statement is if(index!=-1) means the index of 200 is 1 so our if statement look like as if(1!=-1) so this is the true condition so we get out put element found

suppose consider we provide the 600 value to indexOf() method means your code look like as int index = al.indexOf(600) so it is not present in ArrayList collection so it will return -1 in index so our if statement look like as if(-1!=-1) so it is false condition and our else get executed and we found the output element not found.

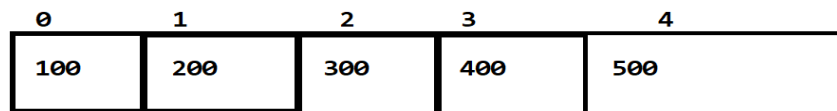
public abstract java.util.ListIterator<E> listIterator(int): this method can travel the List collection **in forward and in backward direction**

Note: example of this method we will discuss later in chapter.

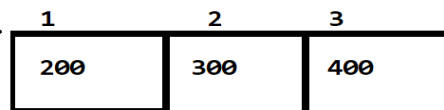
public abstract java.util.List<E> subList(int, int) : it is used for extract the some specified position of List collection between two indexes.

```
ArrayList al = new ArrayList();
```

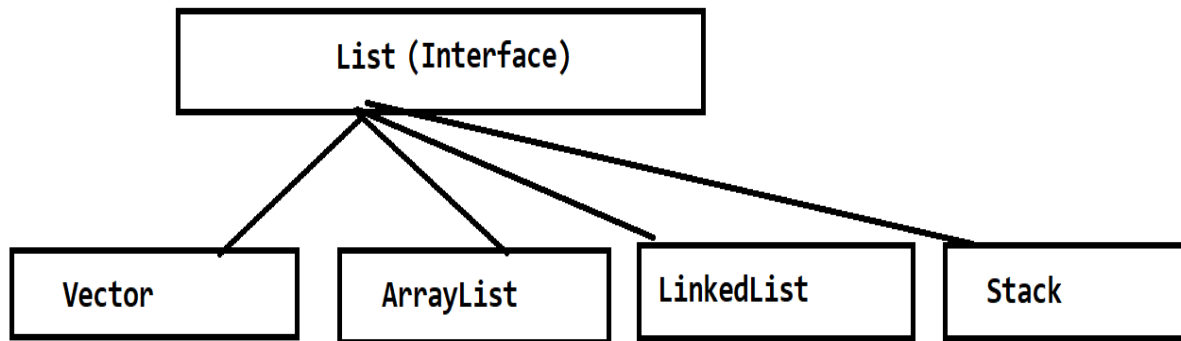
```
al.add(100);  
al.add(200);  
al.add(300);  
al.add(400);  
al.add(500);
```



```
List list = al.subList(1,3);
```

 we extract the List collection between 1 and 3 index.

There are major four implementor classes of List Collection



Now we will discuss about the Vector class

Q.what is the Vector?

- 1) Vector is legacy collection and implement of List collection
- 2) Vector is popular as dynamic array
- 3) Vector is synchronized and Thread safe collection

Q.what is the legacy Collection ?

Legacy collection means those classes is not part of collection in previous version but later they added as part of collection called as legacy collection. Means before jdk 1.2 Vector is not part of Collection but from jdk1.2 version of java Vector added as part collection so people say it is a legacy collection. If we want to work with Vector or create the object of Vector we have the three types of constructor means Vector having overloaded constructor.

Constructors of Vector class

Vector(): this constructor create the object of Vector class with default capacity provided by java collection framework. Default Capacity of Vector is 10 element.

If we want to see the capacity of Vector we have the int capacity() method of Vector class.

Following Example display the capacity of Vector

```
package org.techhub;
```

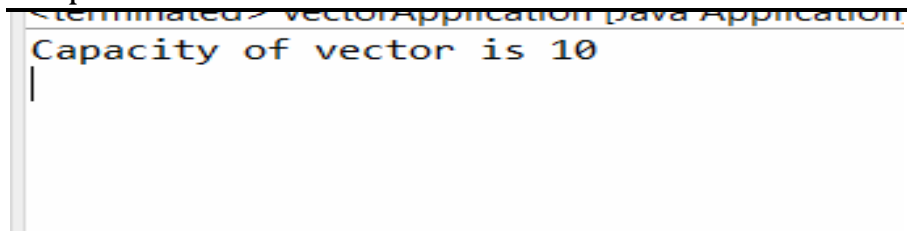
```

import java.util.*;
public class VectorApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v = new Vector();
        System.out.println("Capacity of vector is "+v.capacity());
    }
}

```

Output



```

<terminated> VectorApplication [Java Application] C:\Users\Admi
Capacity of vector is 10
|

```

If we want to set the default capacity of Vector as per your choice we have the second constructor

Vector(int initialCapacity): this constructor is used for create the vector with initial capacity provided by java collection framework.

```

1 package org.techhub;
2
3 import java.util.*;
4 public class VectorApplication {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8
9         Vector v = new Vector(5);
10        System.out.println("Capacity of vector is "+v.capacity());
11    }
12
13 }
14

```



```

<terminated> VectorApplication [Java Application] C:\Users\Admi
Capacity of vector is 5

```

Note : if we use the Vector then Vector occupy double memory than its current capacity if capacity of Vector is cross.

```

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

```

```

public class VectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector(5);
        System.out.println("Capacity of vector is "+v.capacity());
        v.add(100);
        v.add(200);
        v.add(300);
        v.add(400);
        v.add(500);
        v.add(600);

        System.out.println("After Capacity Increment "+v.capacity());
    }
}

```

Output

```

Capacity of vector is 5
After Capacity Increment 10

```

If we want to decide the incremented capacity according to your choice then we have the third constructor Vector given below

Vector(int initialCapacity,int incrementedCapacity)

int initialCapacity : this parameter is used for set up the initial capacity of Vector.

int incrementalCapacity : this parameter is used for set the incremental capacity of Vector after its current capacity cross.

```

import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector(5,2);
        System.out.println("Capacity of vector is "+v.capacity());
        v.add(100);
        v.add(200);
        v.add(300);
        v.add(400);
        v.add(500);
        v.add(600);

        System.out.println("After Capacity Increment "+v.capacity());
    }
}

```

Output

```

Capacity of vector is 5
After Capacity Increment 7

```

In above example we set the first parameter as 5 means the initial capacity of Vector is 5 and second parameter as 2 means incremental capacity of Vector is 2

Means when we add the sixth element in Vector then next capacity is 7 and when we add the 8th element in Vector then next capacity is 9 and so on.

How Vector Constructor work internally

Vector constructor work by using constructor chaining concept internally.

Wh

```
class Vector
{
    Object elementData[];
    int capacityIncrement;
    Vector()
    {
        this(10);
    }
    Vector(int initialCapacity)
    {
        this(initialCapacity,0);
    }
    Vector(int initialCapacity,int capacityIncrement)
    {
        if(initialCapacity<0)
            return throw new IllegalArgumentException("Invalid capacity "+initialCapacity);
        this.elementData=new Object[initialCapacity];
        this.capacityIncrement=capacityIncrement;
    }
}
```

Basic operations with Vector

1. add element 2) fetch element 3) search element 4) remove element 5) count the element in Vector 6) update element 7) insert element on specified position

WAP using Vector class to perform following operations

case 1: Add the new element in collection

case 2: show all data from Vector

case 3: search element from vector

case 4: remove element from vector

case 5: count the number of element present in Vector

case 6: update or replace element on specified index in Vector

case 7 : insert the element on specified position in Vector

Example

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

```

public class VectorApplication {
    public static void main(String[] args) {
        int choice;
        Vector v = new Vector();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the value in collection");
                    int val = xyz.nextInt();
                    v.add(val);
                    break;
                case 2:
                    for (int i = 0; i < v.size(); i++) {
                        System.out.println(v.get(i));
                    }
                    break;
                case 3:
                    System.out.println("Enter the Element for searching in Collection");
                    val = xyz.nextInt();
                    boolean b = v.contains(val);
                    if (b) {
                        System.out.println("Element Found " + val);
                    } else {
                        System.out.println("Element not found ");
                    }
                    break;
                case 4:
                    System.out.println("Enter the element for remove");
                    int value = xyz.nextInt();
                    int index = v.indexOf(value);
                    if (index != -1) {
                        v.remove(index);
                    } else {
                        System.out.println("Element Not Present in collection remove");
                    }
            }
        } while (choice != 5);
    }
}

```

```

        break;
    case 5:
        System.out.println("Number of element in Vector " + v.size());
        break;
    case 6:
        System.out.println("enter the index for replacement");
        index = v.size();
        if (index <= (v.size() - 1)) {
            System.out.println("Enter the value for replacement");
            val = xyz.nextInt();
            v.set(index, val);
        } else {
            System.out.println("Index not present");
        }
        break;
    case 7:
        System.out.println("enter the index for replacement");
        index = v.size();
        if (index <= (v.size() - 1)) {
            System.out.println("Enter the value for replacement");
            val = xyz.nextInt();
            v.insertElementAt(val, index);
        } else {
            System.out.println("Index not present");
        }
        break;
    case 8:
        System.exit(0);
        break;
    default:
        System.out.println("Wrong chocie");
    }
} while (true);
}
}

```

Cursors in Collection or Iterators in Collection

Cursor or Iterators is used in collection for fetching data from collection.

There are major five types of Cursor in Collection

- i) Enumeration
- ii) Iterator
- iii) ListIterator
- iv) for each in jdk 1.5 version of java
- v) for in jdk 1.8 version of java

Enumeration:

Enumeration is read only cursor means we cannot perform any modification on Collection using Enumeration

It only works with legacy collection classes.

Example: Vector , HashTable, Dictionary etc

If we want to create reference of Collection we have the elements() method of Collection this method return reference of Enumeration interface.

Syntax: Enumeration ref = collref.elements();

Enumeration interface provide the following method to us for fetch data from collection

boolean hasMoreElements(): this method check wheather element 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.

Example

```
import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

        Vector v = new Vector();
        v.add(100);
        v.add(200);
        v.add(300);
        Enumeration enm = v.elements();
```

```

        while (enm.hasMoreElements())
        {
            Object obj = enm.nextElement();
            System.out.println(obj);
        }
    }
}

```

Output

```

100
200
300

```

Iterator

Iterator is used for fetch data from collection it work only with forward direction

Q.what is diff between the Enumeration And Iterator

- a) Enumeration is read only cursor and Iterator is not read only means Iterator can remove the eleemnt from collection at the time data fetching
- b) Iterator can work legacy as well as non legacy collection and Enumeration only works with legacy collection
- 3) Enumeration contain two methods boolean hasMoreElement() and Object nextElement() and Iterator contain three methods boolean hasNext(),Object next() and void remove();

If we want to create the reference of Iterator we have to use the iterator() method of Iterable interface and this method return reference of Iterator interface.

Now We want to create program to fetch data from collection using Iterator

```
package org.techhub;
import java.util.*
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        Iterator i = al.iterator();
        while(i.hasNext())
        {
            Object obj =i.next();
            System.out.println(obj);
        }
    }
}
```

Output

<terminated>

```
100
200
300
400
```

3) ListIterator :

ListIterator is used for fetch data in forward direction as well as in backward direction.

Q. what is the diff between ListIterator and Iterator ?

The Major diff between Iterator and ListIterator is

- 1) Iterator can fetch data in forward direction but ListIterator can fetch data in forward as well as in backward direction
- 2) Iterator can remove the data from collection at the time of data fetching but ListIterator can remove,replace,add data in collection at the time of data fetching
- 3) ListIterator can only works with List Collection and Iterator can work with any collection in java.
- 4) methods of ListIterator
boolean hasNext(),boolean hasPrevious(),Object next(),Object previous(),
void remove(Object),void add(Object),void replace(Object) etc
methods of Iterator boolean hasNext() ,Object next() and void remove()
- 5) ListIterator is child of Iterator interface.

If we want to create the reference of ListIterator we have the method name as listIterator(int index):

Syntax: ListIterator ref = collectionref.listIterator(int index);

Example

```
package org.techhub;
import java.util.*
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        ListIterator li = al.listIterator(al.size());
        while(li.hasPrevious())
        { Object obj =li.previous();
```

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

4) fetch data using for each statement of jdk 1.5 version

```
package org.techhub;
import java.util.*
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        for(Object obj:al)
        {
            System.out.println(obj);
        }
    }
}
```

5) fetch data using jdk 1.8 version of foreach

```
package org.techhub;
import java.util.*
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        al.forEach(val->System.out.println(val));
    }
}
```


Examples

WAP to store the 5 values in Vector and calculate its addition

Note: Collection can store data in the form of Object class means Object is parent of all classes in java so Collection can store any type of data in it but when we fetch data from collection and perform any operation on it then we need to perform type conversion from Object class to its specified type. We cannot perform any operation on Object class directly like as addition, multiplication ,division etc

```
package org.techhub;
import java.util.*
public class DemoDataFetching
{
    public static void main(String x[])
    {
        Vector v = new Vector();
        v.add(100);
        v.add(200);
        v.add(300);
        v.add(400);
        v.add(500);
        Iterator i = v.iterator();
        Integer sum =0;
        while(i.hasNext())
        {
            Object obj = i.next();
            sum = sum +(Integer)obj;
        }
        System.out.printf("Sum is %d\n",sum);
    }
}
```

Output

<terminated> DemoDataFetching [Java Applic

Sum is 1500

In above code have the `Object obj = i.next()` and `sum = sum+(Integer)obj`. We convert `Object` class in to the integer form because `Object` can hold data but we cannot perform any operation on object so perform operation on object we have to convert the `Object` class in to the specified its type. So we convert `Object` class reference to `Integer` value.

WAP to create the collection of Employee Data and show it

Steps

1) create the pojo class name as Employee

Sample code

```
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;
```

```
}  
}
```

2) create the object of Vector class

```
package org.techhub;  
import java.util.*  
public class DemoDataFetching  
{  
    public static void main(String x[])  
    {  
        Vector v = new Vector();  
  
    }  
}
```

3) Create the object of Employee pojo classe and store data in it using setter method

```
import java.util.*;  
public class VectorApplication {  
    public static void main(String[] args) {  
  
        Vector v = new Vector();  
        Employee emp1 = new Employee();  
        emp1.setId(1);  
        emp1.setName("Ram");  
        emp1.setSal(10000);  
  
        Employee emp2 = new Employee();  
        emp2.setId(2);  
        emp2.setName("Shyam");  
        emp2.setSal(20000);  
  
        Employee emp3 = new Employee();  
        emp3.setId(3);  
        emp3.setName("Dinesh");  
        emp3.setSal(10000);  
    }  
}
```

```
}}
```

4) store the all pojo objects in collection

```
package org.techhub;

import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

        Vector v = new Vector();
        Employee emp1 = new Employee();
        emp1.setId(1);
        emp1.setName("Ram");
        emp1.setSal(10000);

        Employee emp2 = new Employee();
        emp2.setId(2);
        emp2.setName("Shyam");
        emp2.setSal(20000);

        Employee emp3 = new Employee();
        emp3.setId(3);
        emp3.setName("Dinesh");
        emp3.setSal(10000);

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

    }
}
```

5) Fetch Data from collection using Iterator

```
package org.techhub;

import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {
```

```

        Vector v = new Vector();
        Employee emp1 = new Employee();
        emp1.setId(1);
        emp1.setName("Ram");
        emp1.setSal(10000);
        Employee emp2 = new Employee();
        emp2.setId(2);
        emp2.setName("Shyam");
        emp2.setSal(20000);

        Employee emp3 = new Employee();
        emp3.setId(3);
        emp3.setName("Dinesh");
        emp3.setSal(10000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
        Iterator i = v.iterator();
        while(i.hasNext())
        {
            Object obj = i.next();
            System.out.println(obj.getId()+"\t"+obj.getName()+"\t"+obj.getSal());
        }
    }
}

```

Note: if we think about above code it will generate the compile time error to us because we store the Employee class object in Vector collection and when fetch data then we get data in Object class but when we want to use any data then we have to convert the data in its original format means as per our example we have to convert Object class in to the Employee class and then we can fetch data using employee object with getter methods.

Your Correct Code is given below

```

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

        Vector v = new Vector();
    }
}

```

```

        Employee emp1 = new Employee();
        emp1.setId(1);
        emp1.setName("Ram");
        emp1.setSal(10000);
        Employee emp2 = new Employee();
        emp2.setId(2);
        emp2.setName("Shyam");
        emp2.setSal(20000);
        Employee emp3 = new Employee();
        emp3.setId(3);
        emp3.setName("Dinesh");
        emp3.setSal(10000);

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

        Iterator i = v.iterator();
        while(i.hasNext())
        { Object obj = i.next();
          Employee emp=(Employee)obj;//we convert Object to Employee
class because Collection contain employee data

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

```

Output

1	Ram	10000
2	Shyam	20000
3	Dinesh	10000

WAP program create the class name as student with id, name and per and store its objects in Collection and display the only students whose percentage is greater than 70

Steps

1) create the pojo class name as Student with three field name id and per

```
package org.techhub;

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

2) Create the new class with main and create the object of Vector class

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

    public static void main(String[] args) {
```

```
// TODO Auto-generated method stub  
Vector v= new Vector();
```

```
}
```

```
}
```

3) Create the object of Student and set data in it

```
package org.techhub;  
import java.util.*;  
public class StudentVectorApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Vector v= new Vector();  
  
        Student s1 = new Student();  
        s1.setId(1);  
        s1.setName("Ram");  
        s1.setPer(60.5f);  
  
        Student s2 = new Student();  
        s2.setId(2);  
        s2.setName("dinesh");  
        s2.setPer(50.5f);  
  
        Student s3 = new Student();  
        s3.setId(3);  
        s3.setName("Rajesh");  
        s3.setPer(80.5f);  
  
    }  
}
```

4) Add the students object in Vector class

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

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v= new Vector();

        Student s1 = new Student();
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(60.5f);

        Student s2 = new Student();
        s2.setId(2);
        s2.setName("dinesh");
        s2.setPer(50.5f);

        Student s3 = new Student();
        s3.setId(3);
        s3.setName("Rajesh");
        s3.setPer(80.5f);
        v.add(s1);
        v.add(s2);
        v.add(s3);

    }

}
```

4) Use The Iterator and fetch data from collection In the form of Student

```
package org.techhub;

import java.util.*;

public class StudentVectorApplication {
```

```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    Vector v = new Vector();

    Student s1 = new Student();
    s1.setId(1);
    s1.setName("Ram");
    s1.setPer(60.5f);

    Student s2 = new Student();
    s2.setId(2);
    s2.setName("dinesh");
    s2.setPer(50.5f);

    Student s3 = new Student();
    s3.setId(3);
    s3.setName("Rajesh");
    s3.setPer(80.5f);
    v.add(s1);
    v.add(s2);
    v.add(s3);
    Iterator i = v.iterator();
    while(i.hasNext())
    {
        Object obj = i.next();
        Student std=(Student)obj;

    }
}

```

5) Fetch per from student class using getPer() method

```

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

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}

```

```

    Vector v = new Vector();

    Student s1 = new Student();
    s1.setId(1);
    s1.setName("Ram");
    s1.setPer(60.5f);

    Student s2 = new Student();
    s2.setId(2);
    s2.setName("dinesh");
    s2.setPer(50.5f);

    Student s3 = new Student();
    s3.setId(3);
    s3.setName("Rajesh");
    s3.setPer(80.5f);
    v.add(s1);
    v.add(s2);
    v.add(s3);
    Iterator i = v.iterator();
    while(i.hasNext())
    {
        Object obj = i.next();
        Student std=(Student)obj;
        if(std.getPer()>70)
        {
            System.out.println(std.getId()+"\t"+std.getName()+"\t"+std.getPer());
        }
    }
}

```

WAP program to create the class name as Employee with field name id and salary and store 5 employee data in collection and search employee using its id.

Steps to implement above assignment

1) Create the class name as Employee with setter and getter method with parameterized constructor and perform constructor overloading

```
package org.techhub;
```

```
public class Employee {
```

```
    private int id;
```

```
    private String name;
```

```
    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;
```

```
    }
```

```
    private int sal;
```

```
}
```

2) Create the class with main method and create object of vector class in it.

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

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);

    }
}
```

3) Add the employee objects in Vector class

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

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

    }
}
```

5) Fetch data from collection using Iterator

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

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
        for(Object obj:v) {
            Employee emp=(Employee)obj;    }
    }
}
```

6) Enter the id for search record in collection and fetch id from employee object and compare and setup the flag

```
package org.techhub;

import java.util.*;

public class EmployeeApplication {

    public static void main(String x[]) {
        Vector v = new Vector();
        Employee emp1 = new Employee("Ram", 1, 1000);
        Employee emp2 = new Employee("Shyam", 2, 2000);
        Employee emp3 = new Employee("Ghanshyam", 3, 30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
        Scanner xyz = new Scanner(System.in);

        System.out.println("Enter the search id for employee");
        int sid = xyz.nextInt();
```

```

        boolean flag = false;
        for (Object obj : v) {
            Employee emp = (Employee) obj;
            if (emp.getId() == sid) {
                flag = true;
                break;
            }
        }
        if (flag) {
            System.out.println("Employee found");
        } else {
            System.out.println("Employee Not Found");
        }
    }
}

```

**WAP Program to create the two classes name as Player and Team
Player class looks like as**

```

class Player
{ private int id;
  private String name;
  private int runs;
  public Player(String name,int id,int runs)
  {
      this.name=name;
      this.id=id;
      this.runs=runs;
  }
  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 setRuns(int runs)
{ this.runs=runs;
}
public int getRuns()
{ return runs;
}
}
```

Your Team class look like as

```
class Team
{
    void addPlayers (List playList)
    {
        // display here playerlist
    }
}
```

In above example Team class is dependent on player list.

Steps to Implement the Above Example

1) Create the player class

```
class Player
{ private int id;
  private String name;
  private int runs;
  public Player(String name,int id,int runs)
  {
      this.name=name;
      this.id=id;
      this.runs=runs;
  }
  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 setRuns(int runs)
{ this.runs=runs;
}
public int getRuns()
{return runs;
}
}

```

**2)Create the Team class and access the reference of List interface
And fetch data from collection and display it**

```

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

    void addPlayers(List playerList)
    {
        for(Object obj:playerList)
        {
            Player p =(Player)obj;

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

}

```

3) Create the main class and create the object player class store all objects in List collection

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

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List list = new Vector();

        Player p1= new Player("Ram",1,10000);
        Player p2 = new Player("Shyam",2,5000);
        Player p3 = new Player("Dinesh",3,30000);
        list.add(p1);
        list.add(p2);
        list.add(p3);

    }
}
```

4) Create the object of Team class and pass reference of List collection in it

```
package org.techhub;
import java.util.*;
public class PlayerApplication {
    public static void main(String[] args) {
        List list = new Vector();
        Player p1= new Player("Ram",1,10000);
        Player p2 = new Player("Shyam",2,5000);
        Player p3 = new Player("Dinesh",3,30000);
        list.add(p1);
        list.add(p2);
        list.add(p3);
        Team t = new Team();
        t.addPlayers (list);
    }
}
```

1) WAP to create the class name as College with method name as addNewAdmission (List adminList) in addNewAdmission() method of Collection class contain the List of student data we have the class name as Student with field id , name , per ,contact and we want to store the all student objects in list collection and pass the list collection reference of addNewAdmission() method which is present in College class.

2) WAP to create the class name as Company with following methods void showEmployeeList(List employeeList): this method can show the all employee list

Employee getEmployeeRecordById (int id): this method can return the specified employee record by using its id

Employee deleteEmployeeRecordById (int id): this method can remove the employee record using its id

Employee getEmployeeById (int id): this method can search the employee record by using its id.

Void updateEmployee(int id): this method update the salary of employee using its id from collection

We have to create the new class name as Employee with three fields id,name and salary and store the all employee objects in List Collection and pass the list collection reference of The company class and perform the above mentioned operation with List collection in Company class

ArrayList Collection

ArrayList is same like as Vector and it is also implementor class of List Collection

But there are some major differences between ArrayList and Vector is

1) Vector occupy double memory when its capacity cross and ArrayList occupy half memory than its current capacity when capacity is cross.

Logic of Vector capacity increment

$$\text{newcapacity} = \text{size} > \text{oldcapacity} ? (\text{oldcapacity} + \text{oldcapacity}) : \text{oldcapacity};$$

Logic of ArrayList capacity increment ?

newcapacity = size > oldcapacity ? Oldcapacity + oldcapacity >> 1 : oldcapacity;

- 2) Vector is legacy collection and ArrayList is non legacy collection
- 3) Vector methods are synchronized means Vector is thread safe collection and ArrayList methods are asynchronized means ArrayList is not thread safe

Similarity between Vector and ArrayList

- 1) Vector default capacity is 10 and ArrayList default capacity is 10
- 2) Vector is part of List implementation and ArrayList also part of list implementation

Sample code with ArrayList Collection

```
ArrayListApplication.java  Markers
<terminated>
1 package org.techhub;
2 import java.util.*;
3 public class ArrayListApplication {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method
7         ArrayList al = new ArrayList();
8         al.add(100);
9         al.add(200);
10        al.add(300);
11        al.add(400);
12        Iterator i = al.iterator();
13        while(i.hasNext())
14        {
15            Object obj = i.next();
16            System.out.println(obj);
17        }
18    }
19
20 }
21
```

WAP to create the class name as Product with production information like as Product name, quantity, price with setter and getter methods and Create the new class name as Bill with calBill(List list) method and calBill() method contain the reference of List interface or collection and we can store the all Product class object in List Collection and pass list reference to callBill and calculate the bill of products

Output should like as

Product Name	Quantity	Rate	Total
Parle-G	5	10	50
Cadbury	10	10	100
Grand Total is			: 150

Source Code

```
package org.techhub.billing;
```

```
public class Product {
```

```
    private String name;  
    private int price;  
    private int quantity;
```

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

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

```
    public int getPrice() {  
        return price;  
    }
```

```

    public void setPrice(int price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}

```

Billing

```

package org.techhub.billing;
import java.util.*;
public class Billing {
    int grandTotal=0;
    void calBill(List list)
    {
        System.out.println("Product Name\tQuantity\tRate\tTotal");

        System.out.println("_____");
        for(Object obj:list)
        {
            Product p =(Product)obj;
            int total = p.getQuantity()*p.getPrice();
            grandTotal = grandTotal + total;

            System.out.println(p.getName()+"\t\t\t"+p.getQuantity()+"\t"+p.getPrice()+"\t"+total);
        }
        System.out.println("_____");
        System.out.println("\t\t\t\tTotal Bill :"+grandTotal);
    }
}

```

BillingApplication

```
package org.techhub.billing;

import java.util.*;

public class BillingApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List al = new ArrayList();
        System.out.println("Enter the Product count");
        Scanner xyz = new Scanner(System.in);
        int size = xyz.nextInt();
        Product p[] = new Product[size];
        for(int i=0; i<p.length;i++)
        {
            p[i]=new Product();
            System.out.println("Enter product name price and quantity");
            xyz = new Scanner(System.in);
            String name=xyz.nextLine();
            int price=xyz.nextInt();
            int quantity=xyz.nextInt();
            p[i].setName(name);
            p[i].setPrice(price);
            p[i].setQuantity(quantity);
            al.add(p[i]);
        }

        Billing b = new Billing();
        b.calBill(al);
    }
}
```

Output

Enter the Product count

2

Enter product name price and quantity

Parle-G

5

10

Enter product name price and quantity

Cadbury

10

10

Product Name	Quantity	Rate	Total
--------------	----------	------	-------

Parle-G	10	5	50
---------	----	---	----

Cadbury	10	10	100
---------	----	----	-----

Total Bill :150

1) WAP to create the Application for OnlineExam Using Collection

Create the pojo class Question with field

question,id,option1,option2,option3,option4,option5,answer

Create the class OnlineExamHelper which contain the following methods

void addNewQuestion(Question question): this method is used for add the new question in ArrayList collection

void removeQuestion(int questionId): this method is used for remove the question from collection

void viewAllQuestions(): this method shows the all question from collection

boolean searchQuestion(String questionname): this method is used for search question using question name

void attemptQuestion(int questionid,String answer): this method can search question using question from collection and check its anwser if answer is correct then count the marks otherwise not

void showResult(): this method show the result means in this method we have to display number of question , count correct question and count incorrect question and display the result in percentage

Steps to implement Above Assignment

- a) create the project in eclipse
- b) create the package name as org.techhub.question
- c) create the pojo Question under the org.techhub.question package.

```
package org.techhub.question;
```

```
public class Question {  
    private int id;  
    private String name;  
    private String option1;  
  
    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 String getOption1() {  
        return option1;  
    }  
}
```

```

    public void setOption1(String option1) {
        this.option1 = option1;
    }
    public String getOption2() {
        return option2;
    }
    public void setOption2(String option2) {
        this.option2 = option2;
    }
    public String getOption3() {
        return option3;
    }
    public void setOption3(String option3) {
        this.option3 = option3;
    }
    public String getOption4() {
        return option4;
    }
    public void setOption4(String option4) {
        this.option4 = option4;
    }
    public String getAnswer() {
        return answer;
    }
    public void setAnswer(String answer) {
        this.answer = answer;
    }
    private String option2;
    private String option3;
    private String option4;
    private String answer;
}

```

d) create the one more package name as org.techhub.helper

e) create the OnlineExamHelper class under the org.techhub.helper package

and create one method name as addNewQuestion(Question question) in OnlineExamHelper class.

```
package org.techhub.helper;  
import org.techhub.question.*;  
import java.util.*;  
public class OnlineExamHelper {  
  
    void addNewQuestion(Question question)  
    {  
  
    }  
}
```

e) create the object of ArrayList Collection in OnlineExamHelper class and add the question objects in ArrayList object under the addNewQuestion method.

```
package org.techhub.helper;  
import org.techhub.question.*;  
import java.util.*;  
public class OnlineExamHelper {  
  
    List list = new ArrayList();  
    void addNewQuestion(Question question)  
    {  
        list.add(question);  
    }  
}
```

f) create the new package name as org.techhub.onlineexam

g) create the class OnlineExamClientApp under the org.techhub.onlineexam package and it contain the main method.

```
package org.techhub.onlineexam;  
  
public class OnlineExamClientApp {  
  
    public static void main(String[] args) {
```

```
// TODO Auto-generated method stub
```

```
}
```

```
}
```

g) create the object of OnlineExamHelper class and call its method addNewQuestion() but in addNewQuestion() method contain reference of Question so we have to create the object of Question class and store data in it like as question ,id,options and answer and pass to addNewQuestion method

```
package org.techhub.onlineexam;
```

```
import java.util.*;
```

```
import org.techhub.helper.OnlineExamHelper;
```

```
import org.techhub.question.Question;
```

```
public class OnlineExamClientApp {
```

```
    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
```

```
        OnlineExamHelper helper = new OnlineExamHelper();
```

```
        do {
```

```
            Scanner xyz = new Scanner(System.in);
```

```
            System.out.println("Enter your choice");
```

```
            int choice = xyz.nextInt();
```

```
            switch (choice) {
```

```
                case 1:
```

```
                    xyz.nextLine();
```

```
                    System.out.println("Enter the question id");
```

```
                    int qid=xyz.nextInt();
```

```
                    xyz.nextLine();
```

```
                    System.out.println("Enter the question");
```

```
                    String question=xyz.nextLine();
```

```
                    System.out.println("Enter the option1 value");
```

```

        String option1 =xyz.nextLine();
        System.out.println("Enter the option2 value");
        String option2 = xyz.nextLine();
        System.out.println("Enter the option3 value");
        String option3 =xyz.nextLine();
        System.out.println("Enter the option4 value");
        String option4=xyz.nextLine();
        System.out.println("Enter the answer");
        String ans=xyz.nextLine();
        Question q = new Question();
        q.setId(qid);
        q.setName(question);
        q.setOption1(option1);
        q.setOption2(option2);
        q.setOption1(option3);
        q.setOption4(option4);
        q.setAnswer(ans);
        helper.addNewQuestion(q);
        break;
    default:
        System.out.println("wrong choice");
    }
    }while(true);
}
}

```

h) create the new function name as viewAllQuestions() under the OnlineExamHelper class and fetch data from collection created in OnlineExamHelper class

```

package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    List list = new ArrayList();
    public void addNewQuestion(Question question)

```

```

    {
        list.add(question);
    }
    public void viewAllQuestions()
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;

            System.out.println(q.getId()+"\t"+q.getName()+"\t"+q.getOption1()+"\t"+q.getOption2()+"\t"+q.getOption3()+"\t"+q.getOption4()+"\t"+q.getAnswer());

        }
    }
}

```

i) call the ViewAllStudents() function from main on choice number 3

```

package org.techhub.onlineexam;
import java.util.*;
import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;

public class OnlineExamClientApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

            switch (choice) {
                case 1:
                    xyz.nextLine();
                    System.out.println("Enter the question id");
                    int qid=xyz.nextInt();
                    xyz.nextLine();

```

```

        System.out.println("Enter the question");
        String question=xyz.nextLine();
        System.out.println("Enter the option1 value");
        String option1 =xyz.nextLine();
        System.out.println("Enter the option2 value");
        String option2 = xyz.nextLine();
        System.out.println("Enter the option3 value");
        String option3 =xyz.nextLine();
        System.out.println("Enter the option4 value");
        String option4=xyz.nextLine();
        System.out.println("Enter the answer");
        String ans=xyz.nextLine();
        Question q = new Question();
        q.setId(qid);
        q.setName(question);
        q.setOption1(option1);
        q.setOption2(option2);
        q.setOption1(option3);
        q.setOption4(option4);
        q.setAnswer(ans);
        helper.addNewQuestion(q);
        break;
    case 3:
        helper.viewAllQuestions();
        break;
    default:
        System.out.println("wrong choice");
    }
    }while(true);
}
}

```

j) create new method under the class OnlineExamHelper name as void removeQuestion(int questionId) and remove the question using question id
Note: first we have to travel the whole collection and extract the single single object of question from collection and extract the question id from Question object and compare the your input question id with object question object id

and find the index of Question object if question id match then remove the question object by using index of that object.

Example

```
package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    List list = new ArrayList();
    public void addNewQuestion(Question question)
    {
        list.add(question);
    }
    public void viewAllQuestions()
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;

            System.out.println(q.getId()+"\t"+q.getName()+"\t"+q.getOption1()+"\t"+q.getOption2()+"\t"+q.getOption3()+"\t"+q.getOption4()+"\t"+q.getAnswer());
        }
    }
    public void removeQuestion(int questionId)
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;
            int qid=q.getId();
            if(qid==questionId)
            {
                int index=list.indexOf(q);
                if(index!=-1)
                { list.remove(index);
                }
            }
        }
    }
}
```



```
}  
}
```

k) in main method class input the question id under the choice 2 and call the removeQuestion() method of OnlineExamHelper class object and pass the question id in removeQuestion() method

```
package org.techhub.onlineexam;
```

```
import java.util.*;
```

```
import org.techhub.helper.OnlineExamHelper;
```

```
import org.techhub.question.Question;
```

```
public class OnlineExamClientApp {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        OnlineExamHelper helper = new OnlineExamHelper();
```

```
        do {  
            Scanner xyz = new Scanner(System.in);  
            System.out.println("Enter your choice");  
            int choice = xyz.nextInt();
```

```
            switch (choice) {  
                case 1:  
                    xyz.nextLine();  
                    System.out.println("Enter the question id");  
                    int qid=xyz.nextInt();  
                    xyz.nextLine();  
                    System.out.println("Enter the question");  
                    String question=xyz.nextLine();  
                    System.out.println("Enter the option1 value");  
                    String option1 =xyz.nextLine();  
                    System.out.println("Enter the option2 value");  
                    String option2 = xyz.nextLine();  
                    System.out.println("Enter the option3 value");  
                    String option3 =xyz.nextLine();
```

```

        System.out.println("Enter the option4 value");
        String option4=xyz.nextLine();
        System.out.println("Enter the answer");
        String ans=xyz.nextLine();
        Question q = new Question();
        q.setId(qid);
        q.setName(question);
        q.setOption1(option1);
        q.setOption2(option2);
        q.setOption1(option3);
        q.setOption4(option4);
        q.setAnswer(ans);
        helper.addNewQuestion(q);
        break;
    case 2:
        System.out.println("Enter the question id which want to remove");
        int questionId=xyz.nextInt();
        helper.removeQuestion(questionId);
        break;
    case 3:
        helper.viewAllQuestions();
        break;
    default:
        System.out.println("wrong choice");
    }
}while(true);
}

```

}

l) create the searchQuestion(String questionName) under the OnlineExamHelper class and iterate the List of question and fetch single object of Question class from collection and extract the question name using getName() method of Question class and compare the question with user input value if found return true otherwise return false

Example

```
package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {

    List list = new ArrayList();

    public void addNewQuestion(Question question) {
        list.add(question);
    }

    public void viewAllQuestions() {
        for (Object obj : list) {
            Question q = (Question) obj;
            System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t"
+ q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
        }
    }

    public void removeQuestion(int questionId) {
        for (Object obj : list) {
            Question q = (Question) obj;
            int qid = q.getId();
            if (qid == questionId) {
                int index = list.indexOf(q);
                if (index != -1) {
                    list.remove(index);
                }
            }
        }
    }

    public boolean searchQuestion(String questionName) {
        Iterator i = list.iterator();
```

```

        boolean b = false;
        while (i.hasNext()) {
            Object obj = i.next();
            Question q = (Question) obj;
            String question = q.getName();
            if (question.equals(questionName)) {
                b = true;
                break;
            }
        }
        return b;
    }
}

```

m) call the searchQuestion() on case number 4 I main method and pass question from user input to searchQuestion() and catch the result return by searchQuestion() method and compare using if and if return true by searchQuestion() then display message quesiton found otherwise display message question not found.

```

package org.techhub.onlineexam;

```

```

import java.util.*;
import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;
public class OnlineExamClientApp {

```

```

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

            switch (choice) {
                case 1:
                    xyz.nextLine();

```

```

        System.out.println("Enter the question id");
        int qid=xyz.nextInt();
        xyz.nextLine();
        System.out.println("Enter the question");
        String question=xyz.nextLine();
        System.out.println("Enter the option1 value");
        String option1 =xyz.nextLine();
        System.out.println("Enter the option2 value");
        String option2 = xyz.nextLine();
        System.out.println("Enter the option3 value");
        String option3 =xyz.nextLine();
        System.out.println("Enter the option4 value");
        String option4=xyz.nextLine();
        System.out.println("Enter the answer");
        String ans=xyz.nextLine();
        Question q = new Question();
        q.setId(qid);
        q.setName(question);
        q.setOption1(option1);
        q.setOption2(option2);
        q.setOption1(option3);
        q.setOption4(option4);
        q.setAnswer(ans);
        helper.addNewQuestion(q);
        break;
    case 2:
        System.out.println("Enter the question id which want to remove");
        int questionId=xyz.nextInt();
        helper.removeQuestion(questionId);
        break;
    case 3:
        helper.viewAllQuestions();
        break;
    case 4:
        System.out.println("Enter the question for searching purpose");
        xyz.nextLine();
        question=xyz.nextLine();
        boolean b=helper.searchQuestion(question);
        if(b)

```

```

        {
            System.out.println("Question Found");
        }
        else
        {
            System.out.println("Question not found");
        }
        break;
    default:
        System.out.println("wrong choice");
    }
}while(true);
}
}

```

N) attempt question logic : we have to create method name as attemptQuestion(int questionid,String ans) in OnlineExamHelper class and we have the fetch single single Question object from collection class and fetch question id and question answer and compare with given input by user and if found then increment counter and initialize the counter by default with 0 value

Example

```

package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {

    List list = new ArrayList();
    int count=0;
    public void addNewQuestion(Question question) {
        list.add(question);
    }

    public void viewAllQuestions() {
        for (Object obj : list) {
            Question q = (Question) obj;

```

```

        System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t"
        + q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
    }
}

```

```

public void removeQuestion(int questionId) {
    for (Object obj : list) {
        Question q = (Question) obj;
        int qid = q.getId();
        if (qid == questionId) {
            int index = list.indexOf(q);
            if (index != -1) {
                list.remove(index);
            }
        }
    }
}

```

```

public boolean searchQuestion(String questionName) {
    Iterator i = list.iterator();
    boolean b = false;
    while (i.hasNext()) {
        Object obj = i.next();
        Question q = (Question) obj;
        String question = q.getName();
        if (question.equals(questionName)) {
            b = true;
            break;
        }
    }
    return b;
}

public void attemptQuestion(int questionId,String answer)
{
    for(Object obj:list)
    {
        Question q=(Question)obj;

```

```

        int qid=q.getId();
        String ans=q.getAnswer();
        if(qid==questionId && ans.equals(answer))
        {
            ++count;
        }
    }
}

```

O) call the attemptQuestion() function on choice number 5 in main method and we provide the question id and answer as input to attemptQuestion function

Example

```

package org.techhub.onlineexam;

```

```

import java.util.*;

```

```

import org.techhub.helper.OnlineExamHelper;

```

```

import org.techhub.question.Question;

```

```

public class OnlineExamClientApp {

```

```

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

```

```

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

```

```

            switch (choice) {
                case 1:
                    xyz.nextLine();
                    System.out.println("Enter the question id");
                    int qid=xyz.nextInt();

```



```

xyz.nextLine();
System.out.println("Enter the question");
String question=xyz.nextLine();
System.out.println("Enter the option1 value");
String option1 =xyz.nextLine();
System.out.println("Enter the option2 value");
String option2 = xyz.nextLine();
System.out.println("Enter the option3 value");
String option3 =xyz.nextLine();
System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;
case 2:
remove");
System.out.println("Enter the question id which want to

int questionId=xyz.nextInt();
helper.removeQuestion(questionId);
break;
case 3:
helper.viewAllQuestions();
break;
case 4:
purpose");
System.out.println("Enter the question for searching

xyz.nextLine();
question=xyz.nextLine();
boolean b=helper.searchQuestion(question);
if(b)

```

```

        {
            System.out.println("Question Found");
        }
        else
        {
            System.out.println("Question not found");
        }
        break;
    case 5:
        xyz.nextLine();
        System.out.println("Enter the answer");
        ans=xyz.nextLine();
        System.out.println("Enter the question number");
        qid=xyz.nextInt();
        helper.attemptQuestion(qid, ans);
    default:
        System.out.println("wrong choice");
    }
}while(true);
}

```

}

P) create the showResult() method in OnlineExamHelper class and in this method we have the calculate the size of collection means we have to see the total number of question and we have to perform subtraction operation means we have to subtract correct question count from size of collection means we get incorrect question automatically and we have to calculate its percentage .

Example

```

package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {

    List list = new ArrayList();

```

```

int count=0;
    public void addNewQuestion(Question question) {
        list.add(question);
    }

    public void viewAllQuestions() {
        for (Object obj : list) {
            Question q = (Question) obj;
            System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t"
                + q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
        }
    }

    public void removeQuestion(int questionId) {
        for (Object obj : list) {
            Question q = (Question) obj;
            int qid = q.getId();
            if (qid == questionId) {
                int index = list.indexOf(q);
                if (index != -1) {
                    list.remove(index);
                }
            }
        }
    }

    public boolean searchQuestion(String questionName) {
        Iterator i = list.iterator();
        boolean b = false;
        while (i.hasNext()) {
            Object obj = i.next();
            Question q = (Question) obj;
            String question = q.getName();
            if (question.equals(questionName)) {
                b = true;
                break;
            }
        }
    }

```

```

    }
    return b;
}
public void attemptQuestion(int questionId,String answer)
{
    for(Object obj:list)
    {
        Question q=(Question)obj;
        int qid=q.getId();
        String ans=q.getAnswer();
        if(qid==questionId && ans.equals(answer))
        {
            ++count;
        }
    }
}
public void showResult()
{ float totalQuestion = (float)list.size();
  float incorrectQuestion=totalQuestion-count;
  float per=count/totalQuestion;
  System.out.println("Percentage acheive by student "+(per*100));
}
}

```

Q) create the case number six in main method and call the showresult() function of OnlineExamHelper class on case number 6

Example

```

package org.techhub.onlineexam;

import java.util.*;

import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;

public class OnlineExamClientApp {

    public static void main(String[] args) {

```

```

// TODO Auto-generated method stub
OnlineExamHelper helper = new OnlineExamHelper();

do {
    Scanner xyz = new Scanner(System.in);
    System.out.println("Enter your choice");
    int choice = xyz.nextInt();

    switch (choice) {
        case 1:
            xyz.nextLine();
            System.out.println("Enter the question id");
            int qid=xyz.nextInt();
            xyz.nextLine();
            System.out.println("Enter the question");
            String question=xyz.nextLine();
            System.out.println("Enter the option1 value");
            String option1 =xyz.nextLine();
            System.out.println("Enter the option2 value");
            String option2 = xyz.nextLine();
            System.out.println("Enter the option3 value");
            String option3 =xyz.nextLine();
            System.out.println("Enter the option4 value");
            String option4=xyz.nextLine();
            System.out.println("Enter the answer");
            String ans=xyz.nextLine();
            Question q = new Question();
            q.setId(qid);
            q.setName(question);
            q.setOption1(option1);
            q.setOption2(option2);
            q.setOption3(option3);
            q.setOption4(option4);
            q.setAnswer(ans);
            helper.addNewQuestion(q);
            break;
        case 2:
            System.out.println("Enter the question id which want to remove");
            int questionId=xyz.nextInt();

```

```

        helper.removeQuestion(questionId);
        break;
    case 3:
        helper.viewAllQuestions();
        break;
    case 4:
        System.out.println("Enter the question for searching
purpose");
        xyz.nextLine();
        question=xyz.nextLine();
        boolean b=helper.searchQuestion(question);
        if(b)
        {
            System.out.println("Question Found");
        }
        else
        {
            System.out.println("Question not found");
        }
        break;
    case 5:
        xyz.nextLine();
        System.out.println("Enter the answer");
        ans=xyz.nextLine();
        System.out.println("Enter the question number");
        qid=xyz.nextInt();
        helper.attemptQuestion(qid, ans);
        break;
    case 6:
        helper.showResult();
    default:
        System.out.println("wrong choice");
    }
}while(true);
}
}

```

2) WAP to create the Program For Training And Placement Activity

Create pojo class name as Student with field name , id,contact,email, graduationPer and one more class TrainingAndPlacementFilter with following methods .

void addNewStudent(Student student): this method accept the student information and store in collection or ArrayList

void debarStudent(String email): this accept the email of student and remove the record from collection if email found if email not show the message student is not found.

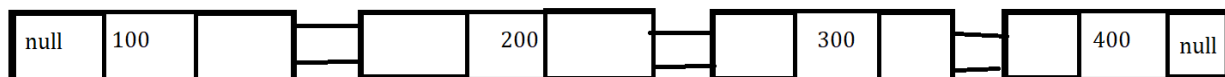
void searchStudent(String email ,String contact): this method search record of student using its email and its contact if found then display the record of student if not found then show the message student is not found.

void arrangeStudent(): this method can show the all students in descending order using their graduationPercentage

LinkedList

LinkedList is by default doubly Linked list in java collection. Double linked list means which having two pointers to ever node previous pointer and next pointers. The Previous pointer points to previous node and next pointer points ot next node. LinkedList is data structure where the elements are not stored in continious location and every element is separate object with data part and address part .the elements are linked using pointers and address. Each element is known as node

Following Diagram shows the double linkedlist



Constructor of LinkedList in java

There are two constructor in LinkedList

1) LinkedList(): this is used for create the default linked list object with no any size.

2) LinkedList(Collection): this constructor is used for create the linked by copying data from another collection .

Performing various operation on LinkedList

1. Adding elements : In order to add() an element to ArrayList we can use the add()method . This method is overloaded to perform multiple operation based on different parameters

void add(Object): this method is used to add an element at the end of the LinkedList

void add(int index,Object): this method is used to add an element at specific index in the LinkedList.

Changing element or replace the element in LinkedList

After adding the element if we wish to change the element ,it can be done using set() method .Since LinkedList is indexed the element which we wish to change is referenced by the index of an element therefore this method takes and index and update the which need to inserted at the index.

void set(int index,Object value): this method can update or change the element on specified index in LinkedList.

Removing elements

In order to remove an element from a LinkedList,we can use the remove() method. This method is also overloaded to perform multiple operations based on different parameter.

void remove(Object): this method is used to simply remove an object from the LinkedList. If there are multiple such objects, then the first occurrence of the object is removed.

void remove(int index): since a LinkedList is indexed, this method takes an integer value which simply removes the element present at the specific index in the LinkedList. After removing the element all the elements are moved to the left to fill the space and the indices of the objects are updated.

Program for LinkedList using Default constructor

Example

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

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LinkedList lst = new LinkedList();
        lst.add(100);
        lst.add(200);
        lst.add(300);
        lst.add(400);
        Iterator i=lst.iterator();
        while(i.hasNext())
        {
            Object obj = i.next();
            System.out.println(obj);
        }

    }

}
```

Program for LinkedList using second constructor

In second constructor we have to copy the content of one collection in to the another collection

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

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        LinkedList lst = new LinkedList(al);
        for(Object obj:lst)
        {
            System.out.println(obj);
        }
    }
}
```

Q.what is the diff between ArrayList and LinkedList

1) ArrayList is dynamic array means it store data in continious allocation and LinkedList is by default doubly LinkedList.

2) ArrayList use the $O(n)$ and LinkedList use the $O(1)$

$O(\text{bigo noation})$ is used for calculate the time complexity of application means the time of complexity of ArrayList is $O(n)$ and the time of complexity of LinkedList is $O(1)$ means if we remove the element from array list or insert the element in ArrayList then it will number of iteration of shifting element means it will take more time for these two operations but if we use the LinkedList then remove and insert the element it will never perform iteration so LinkedList take less time for inserting and removing operation

Means in short we can say LinkedList is faster than ArrayList as for deletion and insertion of element as compare with ArrayList.

Following Example Demonstrate the above concept

Suppose consider we have the ArrayList given below.

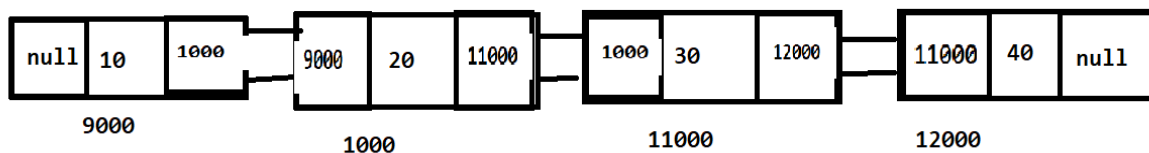
0	1	2	3	4
10	20	30	40	50

ArrayList

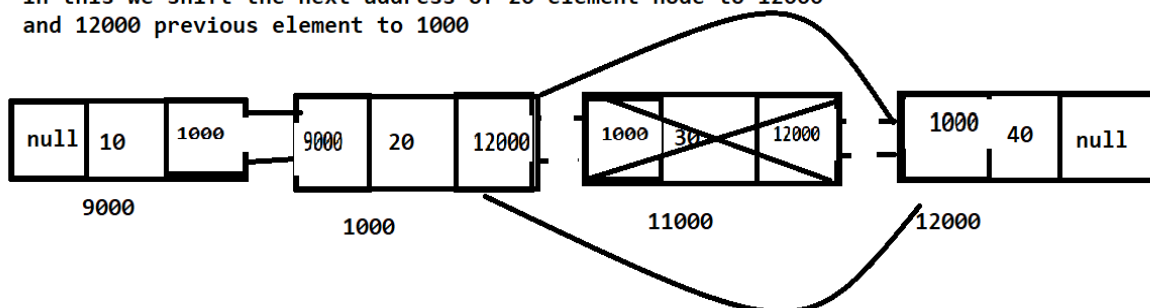
suppose consider we want to delete the element from ArrayList of index 1 then we have to shift the element 2 index on 1 index 3 index on 2 index and fourth index and 3 index and remove the location of last element so it will take time means as per our example we have the perform 3 iteration

suppose consider we have the 1 lakh element in ArrayList and we want to remove the 1 index then there is n number of iteration and it will take more time
so we can say time complexity ArrayList is $O(n)$
so ArrayList is not recommended for deletion and updation purpose.

Suppose consider we have the linkedlist given below.



suppose consider we want to remove the element 30 from LinkedList in this we shift the next address of 20 element node to 12000 and 12000 previous element to 1000



here we not need to perform iteration using linkedlist so the time complexity of LinkedList is $O(1)$

3) ArrayList is faster than linked for data fetching purpose because ArrayList store in memory in continuous manner and LinkedList is slower than ArrayList for data fetching but LinkedList store randomly memory so in LinkedList for data fetching first we have to search the address of the node and then value it will take more

Example

WAP for review management system here we have to perform following task.

Case 1: Add New Teacher

Case 2: Rate To Teacher

Case 3: View Number of Rating to Teacher

Case 4: View Positive Rating

Case 5: view Negative Rating

Case 6: View Average Rating.

Create the pojo class name as Teacher with fields tid,name,rating , count , textreview and create the one more class name as RatingCounter with method name as addTeacher(Teacher) in this method contain the Teacher reference and RatingCounter container the following methods for perform different operation

boolean isAddTeacher(Teacher teacher): this method is used for add the new teacher in Collection

void rateToTeacher(int rating,String teacherName,int teacherId,String textReview): this method is used for rating to particular Teacher

void viewTeacherRating(): this method show the rating of every teacher.

void viewPositiveRating(): this method shows the positive to teacher

void viewNeativeRating(): this method return the negative rating

void showAverateRating(): this method show the average rating to every teacher.

Steps to implementation of above project

- 1) create the java project using eclipse or any other tool.
- 2) create the new package name as org.techhub.teacher and under the package create the new class name as Teacher

Example

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

    public String getName() {
        return name;
    }

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

    public int getId() {
        return id;
    }

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

    public ArrayList getRating() {
        return rating;
    }

    public void setRating(ArrayList rating) {
        this.rating = rating;
    }

    public int getCount() {
        return count;
    }
}
```

```

    public void setCount(int count) {
        this.count = count;
    }

    public ArrayList getTextReview() {
        return textReview;
    }

    public void setTextReview(ArrayList textReview) {
        this.textReview = textReview;
    }

    private ArrayList rating;
    private int count;
    private ArrayList textReview;
}

```

3) create the new package name as org.techhub.rating and create the new class under this package name as RatingCounter

```
package org.techhub.rating;
```

```
public class RatingCounter {

}
```

4) add the new method addTeacher(Teacher teacher) under the RatingCounter class.

```

package org.techhub.rating;
import java.util.*;
import org.techhub.teacher.Teacher;
public class RatingCounter {
    LinkedList teacherList = new LinkedList();
    public boolean isAddTeacher(Teacher teacher)
    {
        boolean b=teacherList.add(teacher);
        return b;
    }
}

```

```
}
```

5) create the new package `org.techhub.clientapp` and under this package we have to create class name as `RatingClientApplication` with main method

```
package org.techhub.clientapp;
```

```
import java.util.*;
```

```
import org.techhub.rating.RatingCounter;
```

```
import org.techhub.teacher.Teacher;
```

```
public class RatingClientApplication {
```

```
    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
```

```
        RatingCounter r = new RatingCounter();
```

```
        do {
```

```
            Scanner xyz = new Scanner(System.in);
```

```
            System.out.println("1:Add New Teacher");
```

```
            System.out.println("2:Rate To Teacher");
```

```
            System.out.println("3:View Teacher Rating");
```

```
            System.out.println("4:View Positive Rating");
```

```
            System.out.println("5:View Negative Rating");
```

```
            System.out.println("6:View Average Rating");
```

```
            System.out.println("Enter Your Choice");
```

```
            int choice = xyz.nextInt();
```

```
            switch (choice) {
```

```
                case 1:
```

```
                    xyz.nextLine();
```

```
                    System.out.println("enter the name of Teacher");
```

```
                    String name=xyz.nextLine();
```

```
                    System.out.println("Enter the id of Teacher");
```

```
                    int id=xyz.nextInt();
```

```
                    Teacher t = new Teacher();
```

```
                    t.setId(id);
```

```
                    t.setName(name);
```

```
                    boolean b = r.isAddTeacher(t);
```

```
                    if(b)//if(true)
```

```
                    {
```

```

        System.out.println("Teacher Added Successfully.....");
    }
    else
    {
        System.out.println("Teacher Not Added .....");
    }
    break;
case 2:
    break;
case 3:
    break;
case 4:
    break;
case 5:
    break;
case 6:
    break;
case 7:
    System.exit(0);
    break;
default:
    System.out.println("wrong choice");
}
} while (true);

}

}

```

6) Rating to Teacher we have the function name as rateToTeacher() with parameters rating and teacher name but we have the LinkedList name as teacherList which contain the n number of teacher so first we have to search teacher name and teacher id for rating if teacher found then we can give rating to teacher otherwise we have to give message teacher not found

Source Code

```

package org.techhub.rating;
import java.util.*;

```



```

import org.techhub.teacher.Teacher;
public class RatingCounter {

    LinkedList teacherList = new LinkedList();

    public boolean isAddTeacher(Teacher teacher)
    {
        boolean b=teacherList.add(teacher);
        return b;
    }

    public void rateToTeacher(int rating,String name,int id,String
textReview)
    { boolean flag=false;
      Teacher t=null;
      for(Object obj:teacherList)
      {
          t=(Teacher)obj;
          String n=t.getName();
          int i=t.getId();
          if(n.equals(name) && i==id)
          {
              flag=true;
              break;
          }
      }
      if(flag)
      {
          ArrayList ratingArrayList=t.getRating();
          if(ratingArrayList==null)
          {
              ratingArrayList = new ArrayList();
          }
          ratingArrayList.add(rating);
          ArrayList textReviewList=t.getTextReview();
          if(textReviewList==null)
          {
              textReviewList=new ArrayList();
          }
      }
    }
}

```

```

        textReviewList.add(textReview);
        t.setRating(ratingArrayList);
        t.setTextReview(textReviewList);
    }
    else
    {
        System.out.println("Teacher Not Found for Rating");
    }
}
public void viewAllRating()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;
        System.out.println(t.getId()+"\t"+t.getName()+"\t"+t.getRating()+"\t"+
t.getTextReview());
    }
}
}

```

Main method class

```

package org.techhub.clientapp;

import java.util.*;
import org.techhub.rating.RatingCounter;
import org.techhub.teacher.Teacher;

public class RatingClientApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        RatingCounter r = new RatingCounter();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("1:Add New Teacher");
            System.out.println("2:Rate To Teacher");
            System.out.println("3:View Teacher Rating");

```

```

System.out.println("4:View Positive Rating");
System.out.println("5:View Negative Rating");
System.out.println("6:View Average Rating");
System.out.println("Enter Your Choice");
int choice = xyz.nextInt();
switch (choice) {
case 1:
    xyz.nextLine();
    System.out.println("enter the name of Teacher");
    String name=xyz.nextLine();
    System.out.println("Enter the id of Teacher");
    int id=xyz.nextInt();
    Teacher t = new Teacher();
    t.setId(id);
    t.setName(name);
    boolean b = r.isAddTeacher(t);
    if(b)//if(true)
    {
        System.out.println("Teacher Added Successfully.....");
    }
    else
    {
        System.out.println("Teacher Not Added .....");
    }
    break;
case 2:
    xyz.nextLine();
    System.out.println("Enter the Teacher name");
    name=xyz.nextLine();
    System.out.println("enter the text review for teacher");
    String textReview =xyz.nextLine();
    System.out.println("Enter the id of teacher");
    id =xyz.nextInt();
    System.out.println("Enter the rating to teacher");
    int rating=xyz.nextInt();
    r.rateToTeacher(rating, name, id, textReview);
    break;
case 3:
    r.viewAllRating();

```

```

        break;
    case 4:
        break;
    case 5:
        break;
    case 6:
        break;
    case 7:
        System.exit(0);
        break;
    default:
        System.out.println("wrong choice");
    }
} while (true);
}

```

7) view positive review : here we have to create the viewReviewRating() function under the RatingCounter class and write the following logics

```

package org.techhub.rating;
import java.util.*;

```

```

import org.techhub.teacher.Teacher;
public class RatingCounter {

```

```

    LinkedList teacherList = new LinkedList();

    public boolean isAddTeacher(Teacher teacher)
    {
        boolean b=teacherList.add(teacher);
        return b;
    }

    public void rateToTeacher(int rating,String name,int id,String
textReview)
    { boolean flag=false;
      Teacher t=null;
      for(Object obj:teacherList)
      {

```

```

        t=(Teacher)obj;
        String n=t.getName();
        int i=t.getId();
        if(n.equals(name) && i==id)
        {
            flag=true;
            break;
        }
    }
    if(flag)
    {
        ArrayList ratingArrayList=t.getRating();
        if(ratingArrayList==null)
        {
            ratingArrayList = new ArrayList();
        }
        ratingArrayList.add(rating);
        ArrayList textReviewList=t.getTextReview();
        if(textReviewList==null)
        {
            textReviewList=new ArrayList();
        }
        textReviewList.add(textReview);
        t.setRating(ratingArrayList);
        t.setTextReview(textReviewList);
    }
    else
    {
        System.out.println("Teacher Not Found for Rating");
    }
}
public void viewAllRating()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;

```

```

        System.out.println(t.getId()+"\t"+t.getName()+"\t"+t.getRating()+"\t"+
t.getTextReview());
    }
}
public void viewPositiveReview()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;
        ArrayList reviewList=t.getTextReview();
        Iterator i=reviewList.iterator();
        System.out.println("View Positive Reviews");

        while(i.hasNext())
        {
            String review=(String)i.next();
            if(review.equals("good") ||review.equals("excellent")
||review.equals("better"))
            {
                System.out.println(review);
            }
        }
    }
}
}

```

Main Method

```

package org.techhub.clientapp;
import java.util.*;
import org.techhub.rating.RatingCounter;
import org.techhub.teacher.Teacher;

public class RatingClientApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}

```

```

RatingCounter r = new RatingCounter();
do {
    Scanner xyz = new Scanner(System.in);
    System.out.println("1:Add New Teacher");
    System.out.println("2:Rate To Teacher");
    System.out.println("3:View Teacher Rating");
    System.out.println("4:View Positive Rating");
    System.out.println("5:View Negative Rating");
    System.out.println("6:View Average Rating");
    System.out.println("Enter Your Choice");
    int choice = xyz.nextInt();
    switch (choice) {
        case 1:
            xyz.nextLine();
            System.out.println("enter the name of Teacher");
            String name=xyz.nextLine();
            System.out.println("Enter the id of Teacher");
            int id=xyz.nextInt();
            Teacher t = new Teacher();
            t.setId(id);
            t.setName(name);
            boolean b = r.isAddTeacher(t);
            if(b)//if(true)
            {
                System.out.println("Teacher Added Successfully.....");
            }
            else
            {
                System.out.println("Teacher Not Added .....");
            }
            break;
        case 2:
            xyz.nextLine();
            System.out.println("Enter the Teacher name");
            name=xyz.nextLine();
            System.out.println("enter the text review for teacher");
            String textReview=xyz.nextLine();
            System.out.println("Enter the id of teacher");
            id =xyz.nextInt();

```

```

        System.out.println("Enter the rating to teacher");
        int rating=xyz.nextInt();
        r.rateToTeacher(rating, name, id, textReview);
        break;
    case 3:
        r.viewAllRating();
        break;
    case 4:
        r.viewPositiveReview();
        break;
    case 5:
        break;
    case 6:
        break;
    case 7:
        System.exit(0);
        break;
    default:
        System.out.println("wrong choice");
    }
} while (true);
}
}

```

Stack

Stack is also implementer class of List Collection. This class work on principal of last in first out format and it is also child class of Vector class

```

List
|
Vector
|
Stack

```

Stack class having only one constructor

Stack(): create the object of stack class using default constructor

Operation with stack

void push (E/Object): this method can push the element in stack the initially top position is -1 when we push the element then top increment by 1

Object pop(): this method can remove the last element from stack collection means this method can remove the top most element from stack collection

Object peek(): this method only show the last or top most element from stack not remove.

boolean isEmpty() : this method check wheather statck is empty or not if empty return true otherwise return false.

WAP to perform basic operation on stack given below

1. Push the element in Stack

2. Pop or Remove element From Stack

3. Display All Element from stack in Last in first out order

Source code given below

```
package org.techhub;
import java.util.*;
public class StackApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Stack s = new Stack();
        int choice;
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the element in stack");
                    int ele = xyz.nextInt();
                    s.push(ele);
                    break;
                case 2:
                    boolean b = s.isEmpty();
                    if (b) {
                        System.out.println("Stack is empty");
                    } else {
```

```

        System.out.println("Removed element is " + s.pop());
    }
    break;
case 3:
    ListIterator li = s.listIterator(s.size());
    while (li.hasPrevious()) {
        Object obj = li.previous();
        System.out.println(obj);
    }
    break;
default:
    System.out.println("wrong choice");
}
} while (true);
}
}

```

Output

```

Enter your choice
1
Enter the element in stack
10
Enter your choice
1
Enter the element in stack
20
Enter your choice
1
Enter the element in stack
30
Enter your choice
1
Enter the element in stack
40
Enter your choice
3
40
30
20
10

```

Mini Project using Stack

WAP to create the passing booking Application

In this application we have the classname as Passenger with field id ,name,contact,source and destination and one more class name as TravelAgency which is depend on Passenger

The Policy the of TravelAgency is first passing allocate last seat which is number 1 then second passenger allocat next second from back side of bus seat number 2 and so on when bus reach destination then last passenger remove first .

Here we have to create the POJO class name as Passenger and we have to create the one class name as TravelAgency with following methods

TravelAgency(): this constructor initialize the seat number by default 0 And when new passengeer added in bus then it will increment by 1 and so on

void addNewPassenger(Passenger passenger) : this method accept the detail of passenger and add in stack

void removePassenger(): this method remove the last passegener added in bus by default .

int getPassgenerCount(): this method return exact of number of passenger in bus.

ListIterator getAllPassengerDetail(): this method return detail of all passenger available in bus.

Steps to Implement above project(refer video from topic Mini Project using Collection Framework)

1) create the new project in eclipse by name SeatBookingApplication

2) create the new package in project name as org.techhub.passenger

And create the pojo class under the org.techhub.passenger package.

Source code given below

```
package org.techhub.passenger;

public class Passenger {

    private int seatNo;
    private String name;
    private String contact;
    private String source;
    private String dest;

    public int getSeatNo() {
        return seatNo;
    }

    public void setSeatNo(int seatNo) {
        this.seatNo = seatNo;
    }

    public String getName() {
        return name;
    }

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

    public String getContact() {
        return contact;
    }

    public void setContact(String contact) {
        this.contact = contact;
    }

    public String getSource() {
        return source;
    }
}
```

```

    public void setSource(String source) {
        this.source = source;
    }

    public String getDest() {
        return dest;
    }

    public void setDest(String dest) {
        this.dest = dest;
    }
}

```

3) Create the package name as org.techhub.travelagency under this package we have to create the class name as TravelAgency

```

package org.techhub.travelagency;

public class TravelAgency {

}

```

4) create the new function/method under the TravelAgency class name as addNewPassenger(Passenger passenger)

This method accept the detail of passegener from main method class and store the passenger detail in Stack.

```

package org.techhub.travelagency;
import java.util.*;
import org.techhub.passenger.Passenger;
public class TravelAgency {
    Stack s = new Stack();
    private static int seatNo;
    public TravelAgency() {
        seatNo = 0;
    }
}

```

```

        public void addNewPassenger(Passenger p) {
            ++seatNo;
            p.setSeatNo(seatNo);
            s.push(p);
        }
    }
}

```

5) create the class name as TravelAgencyApplication with main method and write create the object of TravelAgency and Passenger class and store detail of the passenger in main method by pressing choice one and pass to TravelAgency class by using addNewPassenger() method.

TravelAgency class source code

```

package org.techhub.travelagency;

import java.util.*;

import org.techhub.passenger.Passenger;

public class TravelAgency {

    Stack s = new Stack();
    private static int seatNo;
    public TravelAgency() {
        seatNo = 0;
    }

    public void addNewPassenger(Passenger p) {
        ++seatNo;

        p.setSeatNo(seatNo);
        s.push(p);
    }

    public ListIterator getAllPassengers()
    {
        ListIterator li=s.listIterator(s.size());
        return li;
    }
}

```

```

    public void removePassenger()
    {
        boolean b = s.isEmpty();
        if(b)
        {
            System.out.println("No passenger in bus");
        }
        else
        {
            s.pop();
        }
    }
    public int getPassengerCount() {
        //int count=s.size();
        //return count;
        return s.size();
    }
}

```

Main method class logic

```

package org.techhub.clientapp;

```

```

import java.util.*;

```

```

import org.techhub.passenger.Passenger;

```

```

import org.techhub.travelagency.TravelAgency;

```

```

public class TravelAgencyApplication {

```

```

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        TravelAgency agency = new TravelAgency();
        Scanner xyz = new Scanner(System.in);
        int choice;
        do {
            System.out.println("1:Add New Passenger");
            System.out.println("2:View All Passenger");
            System.out.println("3:Remove Passengers");
            System.out.println("4:Count the Number of Passengers");

```

```

System.out.println("enter your choice");
choice = xyz.nextInt();
switch (choice) {
case 1:
    xyz.nextLine();
    Passenger p = new Passenger();
    System.out.println("enter the name of passenger");
    String name = xyz.nextLine();
    System.out.println("Enter the contact of passenger");
    String contact = xyz.nextLine();
    System.out.println("Enter the source location of
passenger");

    String source = xyz.nextLine();
    System.out.println("Enter the destination location of
passenger");

    String dest = xyz.nextLine();
    p.setName(name);
    p.setContact(contact);
    p.setSource(source);
    p.setDest(dest);
    agency.addNewPassenger(p);
    break;
case 2:
    ListIterator li = agency.getAllPassengers();
    while (li.hasPrevious()) {
        Object obj = li.previous();
        p = (Passenger) obj;
        System.out.println(p.getSeatNo() + "\t" +
p.getName() + "\t" + p.getContact() + "\t" + p.getSource()
+ "\t" + p.getDest());
    }
    break;
case 3:
    agency.removePassenger();
    break;
case 4:
    //int count=agency.getPassengerCount();
    //System.out.println("Now passenger in bus "+count);
    System.out.println("Now passenger in bus "+agency.getPassengerCount());

```



```
        default:
            System.out.println("Wrong choice");
        }
    } while (true);
}
```