Java & JEE Training

Day 15 – Collections

MindsMapped Consulting

Agenda

- Recap of Arrays
- Introduction to Collections API
- Lists – ArrayList, Vector, LinkedList

Recap of Arrays

MindsMapped Consulting

Arrays…

- How are arrays defined?

```
int[] myArray = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };

int[] myArray = new int[10];
myArray[0] = 2;
myArray[1] = 5; …
```

- Iterating through array: Use for loop
- Sort and search: Use java.util.Arrays
  - Demo of arrays…

Major shortcoming of arrays?

- Length of arrays is fixed when the array is created. It cannot be changed after that.
- The solution is to use one of the List classes from the Collections API.
- **PROGRAM = DATA STRUCTURE + ALGORITHM;**

Collections API

MindsMapped Consulting

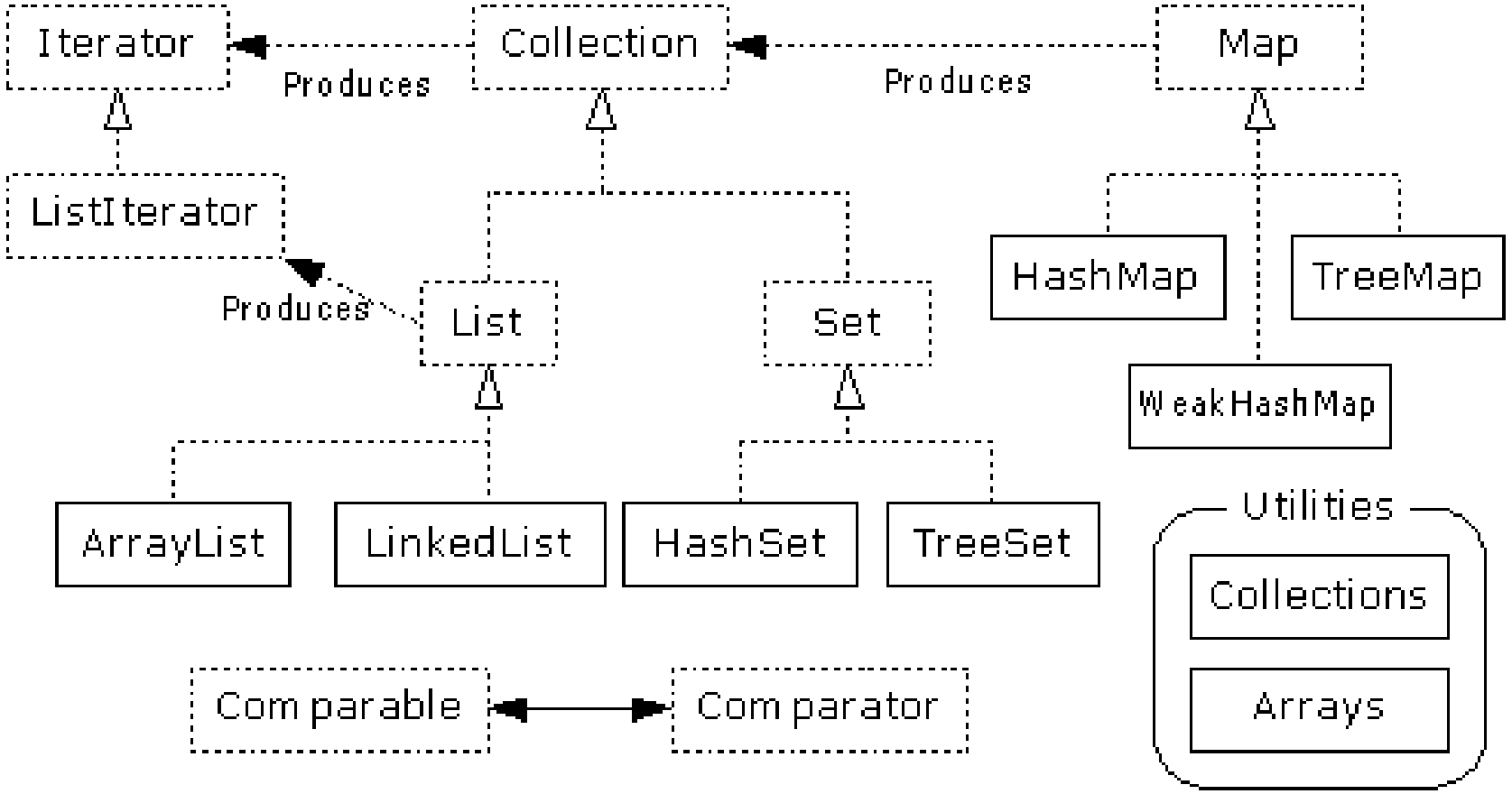
Readings and References

- References
  - "Collections", Java tutorial
  - http://java.sun.com/docs/books/tutorial/collections/index.html

Collections Framework

- Unified architecture for representing and manipulating collections.
- A collections framework contains three things
  - Interfaces
  - Implementations
  - Algorithms

Collections Framework Diagram



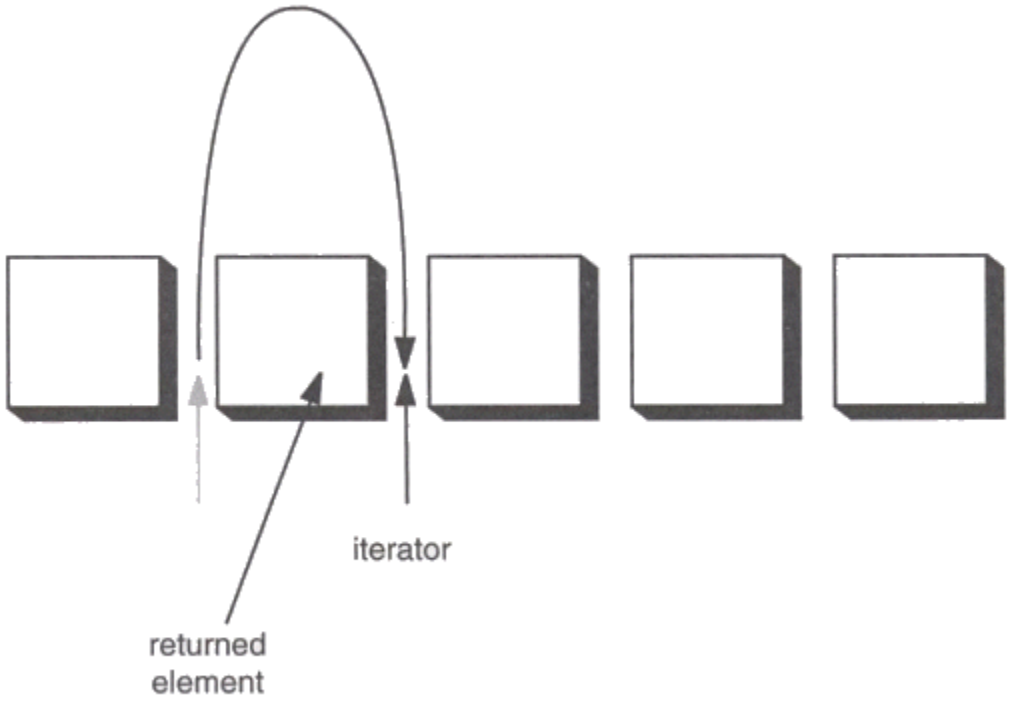
Collection Interface

- Defines fundamental methods
  - `int size();`
  - `boolean isEmpty();`
  - `boolean contains(Object element);`
  - `boolean add(Object element);`
  - `boolean remove(Object element);`
  - `Iterator iterator();`
- These methods are enough to define the basic behavior of a collection
- Provides an Iterator to step through the elements in the Collection

Iterator Interface

- Defines three fundamental methods
  - `Object next()`
  - `boolean hasNext()`
  - `void remove()`
- These three methods provide access to the contents of the collection
- An Iterator knows position within collection
- Each call to next() “reads” an element from the collection
  - `Then you can use it or remove it`

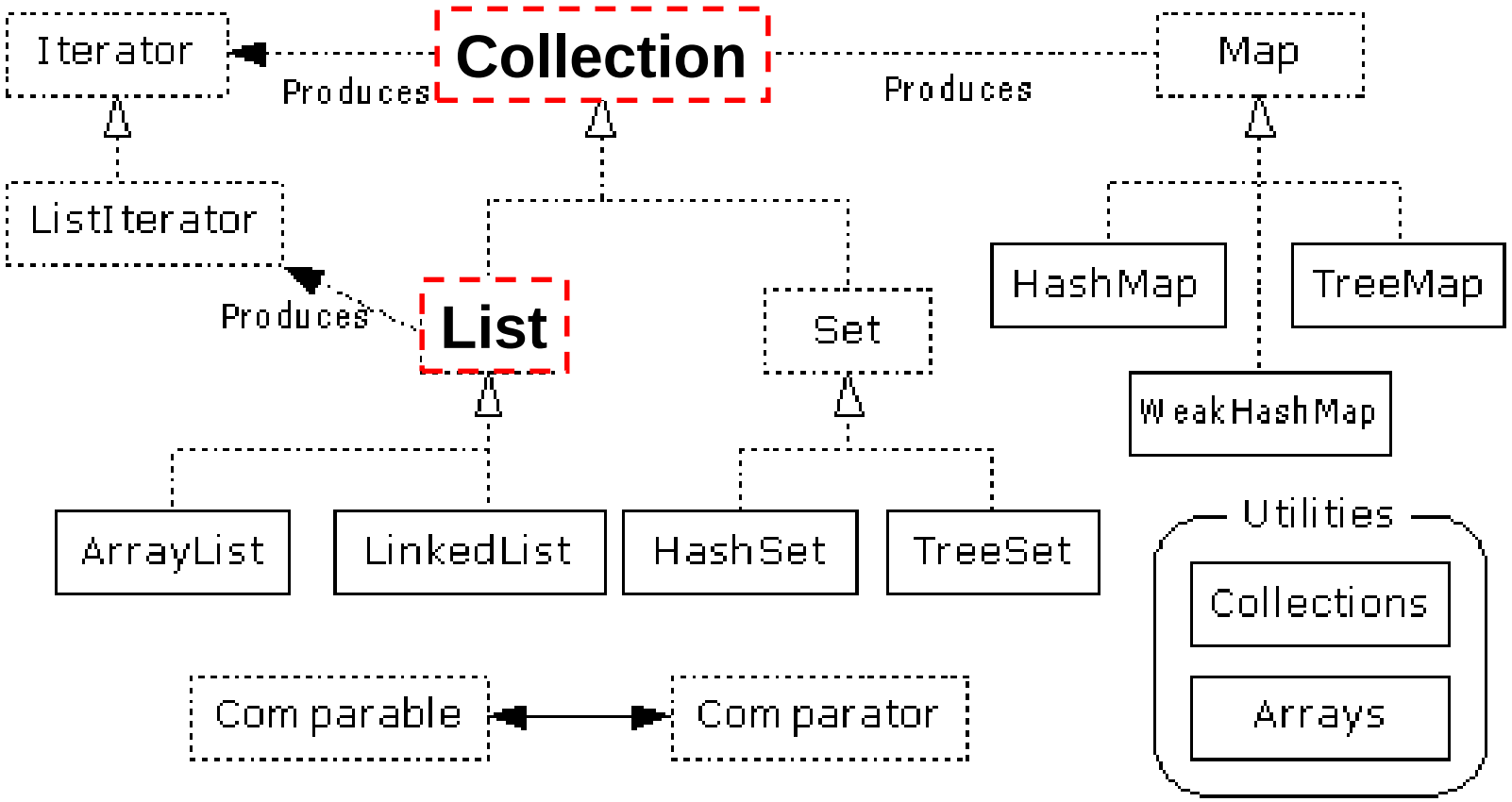
Iterator Position



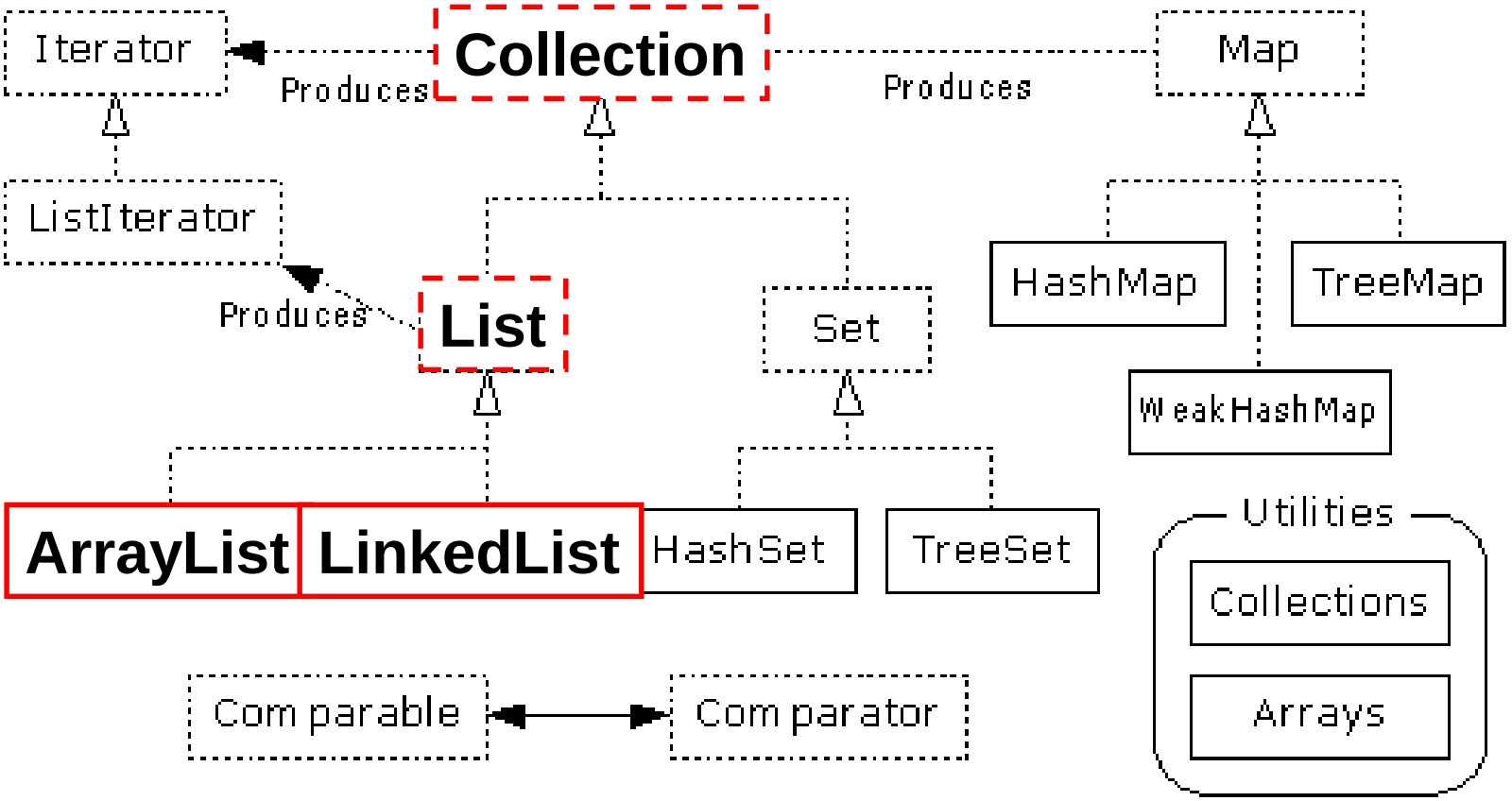
Example - SimpleCollection

```
public class SimpleCollection  {
     public static void main(String[] args) {
             Collection c;
             c = new ArrayList();
             System.out.println(c.getClass().getName());
             for (int i=1; i <= 10; i++) {
                     c.add(i + " * " + i + " = "+i*i);
             }
             Iterator iter = c.iterator();
             while (iter.hasNext())
                     System.out.println(iter.next());
     }
}
```

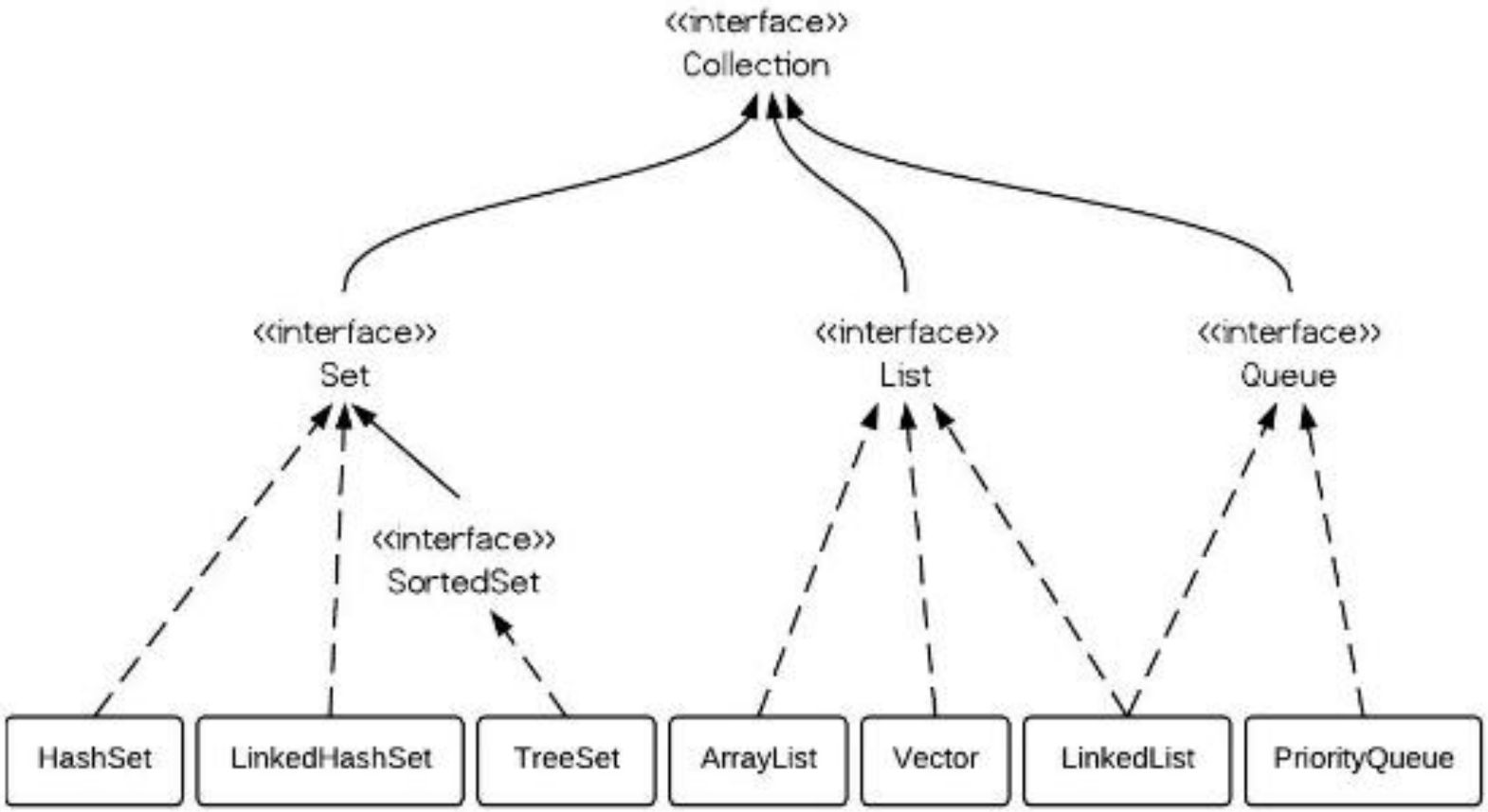
List Interface Context



ArrayList and LinkedList Context



List as part of Collection



List Implementations

- ArrayList
  - low cost random access
  - high cost insert and delete
  - array that resizes if need be
- LinkedList
  - sequential access
  - low cost insert and delete
  - high cost random access
- Vector
  - Similar to ArrayList, but thread-safe

ArrayList overview

- Constant time positional access (it's an array)
- One tuning parameter, the initial capacity

```
public ArrayList(int initialCapacity) {
    super();
    if (initialCapacity < 0)
        throw new IllegalArgumentException(
                "Illegal Capacity: "+initialCapacity);
    this.elementData = new Object[initialCapacity];
}
```

ArrayList methods

- The indexed get and set methods of the List interface are appropriate to use since ArrayLists are backed by an array
  - `Object get(int index)`
  - `Object set(int index, Object element)`
- Indexed add and remove are provided, but can be costly if used frequently
  - `void add(int index, Object element)`
  - `Object remove(int index)`
- May want to resize in one shot if adding many elements
  - `void ensureCapacity(int minCapacity)`

Example: ArrayList

```
ArrayList al = new ArrayList();
al.add(3);
al.add(2);
al.add(1);
al.add(4);
al.add(5);
al.add(6);
al.add(6);

Iterator iter1 = al.iterator();
while(iter1.hasNext()){
System.out.println(iter1.next());
}
```

Example: ArrayList (Using Generics + Iterating using Iterator and for-each loop)

```
import java.util.*;
class TestCollection1{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
  list.add("Ravi");//Adding object in arraylist
  list.add("Vijay");
  list.add("Ravi");
  list.add("Ajay");
  //Traversing list through Iterator
  Iterator itr=list.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
  //Traversing using for-each loop
  for(String obj:list)
    System.out.println(obj);
 }
}
```

User-defined class objects in Java ArrayList

```
class Student{
  int rollno;
  String name;
  int age;
  Student(int rollno,String name,int age){
   this.rollno=rollno;
   this.name=name;
   this.age=age;
  }
}
```

```
import java.util.*;
public class TestCollection3{
 public static void main(String args[]){
  //Creating user-defined class objects
  Student s1=new Student(101,"Sonoo",23);
  Student s2=new Student(102,"Ravi",21);
  Student s3=new Student(103,"Hanumat",25);
  //creating arraylist
  ArrayList<Student> al=new ArrayList<Student>();
  al.add(s1);//adding Student class object
  al.add(s2);
  al.add(s3);
  //Getting Iterator
  Iterator itr=al.iterator();
  //traversing elements of ArrayList object
  while(itr.hasNext()){
    Student st=(Student)itr.next();
    System.out.println(st.rollno+" "+st.name+"
"+st.age);
  }
 }
}
```