class `java.util.ArrayList<E>`

1

Handout B for e01 CS56 M18

The following excerpts from the javadoc for `java.util.ArrayList<E>` may be helpful to you in completing this exam.

Inheritance Hierarchy (complete)

```
java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.ArrayList<E>
```

All Implemented Interfaces:	`Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess`
Direct Known Subclasses:	`AttributeList, RoleList, RoleUnresolvedList`

Constructors (complete)

`ArrayList()`	Constructs an empty list with an initial capacity of ten.
`ArrayList(Collection<? extends E> c)`	Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
`ArrayList(int initialCapacity)`	Constructs an empty list with the specified initial capacity.

Most important methods, with brief description

`boolean`	`add(E e)`	Appends the specified element to the end of this list.
`void`	`add(int index, E element)`	Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices). throws `IndexOutOfBoundsException` if `(index < 0 || index > size())`
`void`	`clear()`	Removes all of the elements from this list.
`E`	`get(int index)`	Returns the element at the specified position in this list.
`int`	`indexOf(Object o)`	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
`boolean`	`isEmpty()`	Returns true if this list contains no elements.
`int`	`lastIndexOf(Object o)`	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
`E`	`remove(int index)`	Removes the element at the specified position in this list.
`boolean`	`remove(Object o)`	Removes the first occurrence of the specified element from this list, if it is present.
`E`	`set(int index, E element)`	Replaces the element at the specified position in this list with the specified element. Returns the element previously at the specified position throws `IndexOutOfBoundsException` if `(index < 0 || index >= size())`
`int`	`size()`	Returns the number of elements in this list.
`void`	`sort(Comparator<? super E> c)`	Sorts this list according to the order induced by the specified `Comparator`.

Additional methods, listed by method signature only.

`boolean addAll(Collection<? extends E> c)`	`boolean addAll(int index, Collection<? extends E> c)`
`Object clone()`	`boolean contains(Object o)`
`void ensureCapacity(int minCapacity)`	`void forEach(Consumer<? super E> action)`
`Iterator<E> iterator()`	`ListIterator<E> listIterator()`
`ListIterator<E> listIterator(int index)`	`boolean removeAll(Collection<?> c)`
`boolean removeIf(Predicate<? super E> filter)`	`protected void removeRange(int fromIndex, int toIndex)`
`void replaceAll(UnaryOperator<E> operator)`	`boolean retainAll(Collection<?> c)`
`Spliterator<E> spliterator()`	`List<E> subList(int fromIndex, int toIndex)`
`Object[] toArray()`	`<T> T[] toArray(T[] a)`
`void trimToSize()`	

Methods inherited from:

`class java.util.AbstractList`	`equals, hashCode`
`class java.util.AbstractCollection`	`containsAll, toString`
`class java.lang.Object`	`finalize, getClass, notify, notifyAll, wait, wait, wait`
`interface java.util.List`	`containsAll, equals, hashCode`
`interface java.util.Collection`	`parallelStream, stream`

2

Handout B for e01 CS56 M18

Handout B, Page 2

```
1   public class Student implements Comparable<Student> {
2
3       private String name;
4       private int perm;
5       private String major;
6
7       public Student (String name, int perm, String major) {
8           this.name = name;
9           this.perm = perm;
10          this.major = major;
11      }
12
13      public String getName() { return name; }
14      public int getPerm() { return perm; }
15      public String getMajor() { return major; }
16
17      @Override
18      public String toString() {
19
20          ____________________________________
21      }
22
23      @Override
24      public boolean equals(Object o) {
25          if (this == o) return true;
26          if (o == null || getClass() != o.getClass()) return false;
27          Student s = (Student) o;
28
29          ____________________________________________
30      }
31
32      @Override
33      public int hashCode() {
34        ____________________
35      }
36
37      /**
38         Natural order is lexicographic order by name.  Break ties by by
39         perm; e.g. if two students are named Chris Lee, put them in
40         order by their perm.
41       */
42
43      @Override
44      public int compareTo(Student s) {
45
46          ________________________
47      }
48  }
```

End of Handout