**ArrayList - Examples and practice problems**

ArrayList in Java is used to store dynamically sized collection of elements. Contrary to Arrays that are fixed in size, an ArrayList grows its size automatically when new elements are added to it.

ArrayList is part of Java’s collection framework and implements Java’s List interface.

​

Following are few key points to note about ArrayList in Java -

* An ArrayList is a re-sizable array, also called a dynamic array. It grows its size to accommodate new elements and shrinks the size when the elements are removed.
* ArrayList internally uses an array to store the elements. Just like arrays, It allows you to retrieve the elements by their index.
* Java ArrayList allows duplicate and null values.
* Java ArrayList is an ordered collection. It maintains the insertion order of the elements.
* You cannot create an ArrayList of primitive types like int, char etc. You need to use boxed types like Integer, Character, Boolean etc.
* Java ArrayList is not synchronized. If multiple threads try to modify an ArrayList at the same time, then the final outcome will be non-deterministic. You must explicitly synchronize access to an ArrayList if multiple threads are gonna modify it.

**Creating an ArrayList and adding new elements to it**

This example shows:

* How to create an ArrayList using the [ArrayList()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#ArrayList--) constructor.
* Add new elements to an ArrayList using the [add()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#add-E-) method.

import java.util.ArrayList;

import java.util.List;

public class CreateArrayListExample {

public static void main(String[] args) {

// Creating an ArrayList of String

List<String> animals = new ArrayList<>();

// Adding new elements to the ArrayList

animals.add("Lion");

animals.add("Tiger");

animals.add("Cat");

animals.add("Dog");

System.out.println(animals);

// Adding an element at a particular index in an ArrayList

animals.add(2, "Elephant");

System.out.println(animals);

}

}

# Output

[Lion, Tiger, Cat, Dog]

[Lion, Tiger, Elephant, Cat, Dog]

**Creating an ArrayList from another collection**

This example shows:

* How to create an ArrayList from another collection using the [ArrayList(Collection c)](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#ArrayList-java.util.Collection-)constructor.
* How to add all the elements from an existing collection to the new ArrayList using the [addAll()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#addAll-java.util.Collection-)method.

import java.util.ArrayList;

import java.util.List;

public class CreateArrayListFromCollectionExample {

public static void main(String[] args) {

List<Integer> firstFivePrimeNumbers = new ArrayList<>();

firstFivePrimeNumbers.add(2);

firstFivePrimeNumbers.add(3);

firstFivePrimeNumbers.add(5);

firstFivePrimeNumbers.add(7);

firstFivePrimeNumbers.add(11);

// Creating an ArrayList from another collection

List<Integer> firstTenPrimeNumbers = new ArrayList<>(firstFivePrimeNumbers);

List<Integer> nextFivePrimeNumbers = new ArrayList<>();

nextFivePrimeNumbers.add(13);

nextFivePrimeNumbers.add(17);

nextFivePrimeNumbers.add(19);

nextFivePrimeNumbers.add(23);

nextFivePrimeNumbers.add(29);

// Adding an entire collection to an ArrayList

firstTenPrimeNumbers.addAll(nextFivePrimeNumbers);

System.out.println(firstTenPrimeNumbers);

}

}

# Output

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

**Accessing elements from an ArrayList**

This example shows:

* How to check if an ArrayList is empty using the [isEmpty()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#isEmpty--) method.
* How to find the size of an ArrayList using the [size()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#size--) method.
* How to access the element at a particular index in an ArrayList using the [get()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#get-int-) method.
* How to modify the element at a particular index in an ArrayList using the [set()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#set-int-E-) method.

import java.util.ArrayList;

import java.util.List;

public class AccessElementsFromArrayListExample {

public static void main(String[] args) {

List<String> topCompanies = new ArrayList<>();

// Check if an ArrayList is empty

System.out.println("Is the topCompanies list empty? : " + topCompanies.isEmpty());

topCompanies.add("Google");

topCompanies.add("Apple");

topCompanies.add("Microsoft");

topCompanies.add("Amazon");

topCompanies.add("Facebook");

// Find the size of an ArrayList

System.out.println("Here are the top " + topCompanies.size() + " companies in the world");

System.out.println(topCompanies);

// Retrieve the element at a given index

String bestCompany = topCompanies.get(0);

String secondBestCompany = topCompanies.get(1);

String lastCompany = topCompanies.get(topCompanies.size() - 1);

System.out.println("Best Company: " + bestCompany);

System.out.println("Second Best Company: " + secondBestCompany);

System.out.println("Last Company in the list: " + lastCompany);

// Modify the element at a given index

topCompanies.set(4, "Walmart");

System.out.println("Modified top companies list: " + topCompanies);

}

}

# Output

Is the topCompanies list empty? : true

Here are the top 5 companies in the world

[Google, Apple, Microsoft, Amazon, Facebook]

Best Company: Google

Second Best Company: Apple

Last Company in the list: Facebook

Modified top companies list: [Google, Apple, Microsoft, Amazon, Walmart]

**Removing elements from an ArrayList**

This example shows:

1. 1.

How to remove the element at a given index in an ArrayList | [remove(int index)](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#remove-int-)

1. 2.

How to remove an element from an ArrayList | [remove(Object o)](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#remove-java.lang.Object-)

1. 3.

How to remove all the elements from an ArrayList that exist in a given collection | [removeAll()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#removeAll-java.util.Collection-)

1. 4.

How to remove all the elements matching a given predicate | [removeIf()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#removeIf-java.util.function.Predicate-)

1. 5.

How to clear an ArrayList | [clear()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#clear--)

import java.util.ArrayList;

import java.util.List;

import java.util.function.Predicate;

public class RemoveElementsFromArrayListExample {

public static void main(String[] args) {

List<String> programmingLanguages = new ArrayList<>();

programmingLanguages.add("C");

programmingLanguages.add("C++");

programmingLanguages.add("Java");

programmingLanguages.add("Kotlin");

programmingLanguages.add("Python");

programmingLanguages.add("Perl");

programmingLanguages.add("Ruby");

System.out.println("Initial List: " + programmingLanguages);

// Remove the element at index `5`

programmingLanguages.remove(5);

System.out.println("After remove(5): " + programmingLanguages);

// Remove the first occurrence of the given element from the ArrayList

// (The remove() method returns false if the element does not exist in the ArrayList)

boolean isRemoved = programmingLanguages.remove("Kotlin");

System.out.println("After remove(\"Kotlin\"): " + programmingLanguages);

// Remove all the elements that exist in a given collection

List<String> scriptingLanguages = new ArrayList<>();

scriptingLanguages.add("Python");

scriptingLanguages.add("Ruby");

scriptingLanguages.add("Perl");

programmingLanguages.removeAll(scriptingLanguages);

System.out.println("After removeAll(scriptingLanguages): " + programmingLanguages);

// Remove all elements from the ArrayList

programmingLanguages.clear();

System.out.println("After clear(): " + programmingLanguages);

}

}

# Output

Initial List: [C, C++, Java, Kotlin, Python, Perl, Ruby]

After remove(5): [C, C++, Java, Kotlin, Python, Ruby]

After remove("Kotlin"): [C, C++, Java, Python, Ruby]

After removeAll(scriptingLanguages): [C, C++, Java]

After clear(): []

**Iterating over an ArrayList**

The following example shows how to iterate over an ArrayList using

1. 1.

Java 8 [forEach](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#forEach-java.util.function.Consumer-) and lambda expression.

1. 2.

[iterator()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#iterator--).

1. 3.

[iterator()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#iterator--) and Java 8 [forEachRemaining()](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html#forEachRemaining-java.util.function.Consumer-) method.

1. 4.

[listIterator()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#listIterator-int-).

1. 5.

Simple for-each loop.

1. 6.

for loop with index.

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.ListIterator;

public class IterateOverArrayListExample {

public static void main(String[] args) {

List<String> tvShows = new ArrayList<>();

tvShows.add("Breaking Bad");

tvShows.add("Game Of Thrones");

tvShows.add("Friends");

tvShows.add("Prison break");

System.out.println("=== Iterate using Java 8 forEach and lambda ===");

tvShows.forEach(tvShow -> {

System.out.println(tvShow);

});

System.out.println("\n=== Iterate using an iterator() ===");

Iterator<String> tvShowIterator = tvShows.iterator();

while (tvShowIterator.hasNext()) {

String tvShow = tvShowIterator.next();

System.out.println(tvShow);

}

System.out.println("\n=== Iterate using an iterator() and Java 8 forEachRemaining() method ===");

tvShowIterator = tvShows.iterator();

tvShowIterator.forEachRemaining(tvShow -> {

System.out.println(tvShow);

});

System.out.println("\n=== Iterate using a listIterator() to traverse in both directions ===");

// Here, we start from the end of the list and traverse backwards.

ListIterator<String> tvShowListIterator = tvShows.listIterator(tvShows.size());

while (tvShowListIterator.hasPrevious()) {

String tvShow = tvShowListIterator.previous();

System.out.println(tvShow);

}

System.out.println("\n=== Iterate using simple for-each loop ===");

for(String tvShow: tvShows) {

System.out.println(tvShow);

}

System.out.println("\n=== Iterate using for loop with index ===");

for(int i = 0; i < tvShows.size(); i++) {

System.out.println(tvShows.get(i));

}

}

}

# Output

=== Iterate using Java 8 forEach and lambda ===

Breaking Bad

Game Of Thrones

Friends

Prison break

=== Iterate using an iterator() ===

Breaking Bad

Game Of Thrones

Friends

Prison break

=== Iterate using an iterator() and Java 8 forEachRemaining() method ===

Breaking Bad

Game Of Thrones

Friends

Prison break

=== Iterate using a listIterator() to traverse in both directions ===

Prison break

Friends

Game Of Thrones

Breaking Bad

=== Iterate using simple for-each loop ===

Breaking Bad

Game Of Thrones

Friends

Prison break

=== Iterate using for loop with index ===

Breaking Bad

Game Of Thrones

Friends

Prison break

The iterator() and listIterator() methods are useful when you need to modify the ArrayList while traversing.

Consider the following example, where we remove elements from the ArrayList using iterator.remove() method while traversing through it -

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class ArrayListIteratorRemoveExample {

public static void main(String[] args) {

List<Integer> numbers = new ArrayList<>();

numbers.add(13);

numbers.add(18);

numbers.add(25);

numbers.add(40);

Iterator<Integer> numbersIterator = numbers.iterator();

while (numbersIterator.hasNext()) {

Integer num = numbersIterator.next();

if(num % 2 != 0) {

numbersIterator.remove();

}

}

System.out.println(numbers);

}

}

# Output

[18, 40]

**Searching for elements in an ArrayList**

The example below shows how to:

* Check if an ArrayList contains a given element | [contains()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#contains-java.lang.Object-)
* Find the index of the first occurrence of an element in an ArrayList | [indexOf()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#indexOf-java.lang.Object-)
* Find the index of the last occurrence of an element in an ArrayList | [lastIndexOf()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#lastIndexOf-java.lang.Object-)

import java.util.ArrayList;

import java.util.List;

public class SearchElementsInArrayListExample {

public static void main(String[] args) {

List<String> names = new ArrayList<>();

names.add("John");

names.add("Alice");

names.add("Bob");

names.add("Steve");

names.add("John");

names.add("Steve");

names.add("Maria");

// Check if an ArrayList contains a given element

System.out.println("Does names array contain \"Bob\"? : " + names.contains("Bob"));

// Find the index of the first occurrence of an element in an ArrayList

System.out.println("indexOf \"Steve\": " + names.indexOf("Steve"));

System.out.println("indexOf \"Mark\": " + names.indexOf("Mark"));

// Find the index of the last occurrence of an element in an ArrayList

System.out.println("lastIndexOf \"John\" : " + names.lastIndexOf("John"));

System.out.println("lastIndexOf \"Bill\" : " + names.lastIndexOf("Bill"));

}

}

# Output

Does names array contain "Bob"? : true

indexOf "Steve": 3

indexOf "Mark": -1

lastIndexOf "John" : 4

lastIndexOf "Bill" : -1

## ArrayList of user defined objects

Since ArrayList supports generics, you can create an ArrayList of **any** type. It can be of simple types like Integer, String, Double or complex types like an ArrayList of ArrayLists, or an ArrayList of HashMaps or an ArrayList of any user defined objects.

In the following example, you’ll learn how to create an ArrayList of user defined objects.

import java.util.ArrayList;

import java.util.List;

class User {

private String name;

private int age;

public User(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

public class ArrayListUserDefinedObjectExample {

public static void main(String[] args) {

List<User> users = new ArrayList<>();

users.add(new User("Rajeev", 25));

users.add(new User("John", 34));

users.add(new User("Steve", 29));

users.forEach(user -> {

System.out.println("Name : " + user.getName() + ", Age : " + user.getAge());

});

}

}

# Output

Name : Rajeev, Age : 25

Name : John, Age : 34

Name : Steve, Age : 29

**Sorting an ArrayList**

Sorting an ArrayList is a very common task that you will encounter in your programs. In this section, I’ll show you how to -

* Sort an ArrayList using [Collections.sort()](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html#sort-java.util.List-) method.
* Sort an ArrayList using [ArrayList.sort()](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#sort-java.util.Comparator-) method.
* Sort an ArrayList of user defined objects with a custom comparator.

**1. Sort an ArrayList using Collections.sort() method**

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class ArrayListCollectionsSortExample {

public static void main(String[] args) {

List<Integer> numbers = new ArrayList<>();

numbers.add(13);

numbers.add(7);

numbers.add(18);

numbers.add(5);

numbers.add(2);

System.out.println("Before : " + numbers);

// Sorting an ArrayList using Collections.sort() method

Collections.sort(numbers);

System.out.println("After : " + numbers);

}

}

# Output

Before : [13, 7, 18, 5, 2]

After : [2, 5, 7, 13, 18]

2. Sort an ArrayList using ArrayList.sort() method

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

public class ArrayListSortExample {

public static void main(String[] args) {

List<String> names = new ArrayList<>();

names.add("Lisa");

names.add("Jennifer");

names.add("Mark");

names.add("David");

System.out.println("Names : " + names);

// Sort an ArrayList using its sort() method. You must pass a Comparator to the ArrayList.sort() method.

names.sort(new Comparator<String>() {

@Override

public int compare(String name1, String name2) {

return name1.compareTo(name2);

}

});

// The above `sort()` method call can also be written simply using lambda expression

names.sort((name1, name2) -> name1.compareTo(name2));

// Following is an even more concise solution

names.sort(Comparator.naturalOrder());

System.out.println("Sorted Names : " + names);

}

}

# Output

Names : [Lisa, Jennifer, Mark, David]

Sorted Names : [David, Jennifer, Lisa, Mark]

**3. Sort an ArrayList of Objects using custom Comparator**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Person {

private String name;

private Integer age;

public Person(String name, Integer age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

@Override

public String toString() {

return "{" +

"name='" + name + '\'' +

", age=" + age +

'}';

}

}

public class ArrayListObjectSortExample {

public static void main(String[] args) {

List<Person> people = new ArrayList<>();

people.add(new Person("Sachin", 47));

people.add(new Person("Chris", 34));

people.add(new Person("Rajeev", 25));

people.add(new Person("David", 31));

System.out.println("Person List : " + people);

// Sort People by their Age

people.sort((person1, person2) -> {

return person1.getAge() - person2.getAge();

});

// A more concise way of writing the above sorting function

people.sort(Comparator.comparingInt(Person::getAge));

System.out.println("Sorted Person List by Age : " + people);

// You can also sort using Collections.sort() method by passing the custom Comparator

Collections.sort(people, Comparator.comparing(Person::getName));

System.out.println("Sorted Person List by Name : " + people);

}

}

# Output

Person List : [{name='Sachin', age=47}, {name='Chris', age=34}, {name='Rajeev', age=25}, {name='David', age=31}]

Sorted Person List by Age : [{name='Rajeev', age=25}, {name='David', age=31}, {name='Chris', age=34}, {name='Sachin', age=47}]

Sorted Person List by Name : [{name='Chris', age=34}, {name='David', age=31}, {name='Rajeev', age=25}, {name='Sachin', age=47}]

### Example d: Get Sub List of Java ArrayList Example

1

\*

Get Sub List of Java ArrayList Example

This Java Example shows how to get sub list of java ArrayList using subList

method by providing start and end index.

\*/

import java.util.ArrayList;

import java.util.List;

public class GetSubListOfJavaArrayListExample {

public static void main(String[] args) {

//create an ArrayList object

ArrayList arrayList = new ArrayList();

//Add elements to Arraylist

arrayList.add("1");

arrayList.add("2");

arrayList.add("3");

arrayList.add("4");

arrayList.add("5");

/\*

To get a sub list of Java ArrayList use

List subList(int startIndex, int endIndex) method.

This method returns an object of type List containing elements from

startIndex to endIndex - 1.

\*/

List lst = arrayList.subList(1,3);

//display elements of sub list.

System.out.println("Sub list contains : ");

for(int i=0; i< lst.size() ; i++)

System.out.println(lst.get(i));

/\*

Sub List returned by subList method is backed by original Arraylist. So any

changes made to sub list will also be REFLECTED in the original Arraylist.

\*/

//remove one element from sub list

Object obj = lst.remove(0);

System.out.println(obj + " is removed from sub list");

//print original ArrayList

System.out.println("After removing " + obj + " from sub list, original ArrayList contains : ");

for(int i=0; i< arrayList.size() ; i++)

System.out.println(arrayList.get(i));

}

}

/\*

Output would be

Sub list contains :

2

3

2 is removed from sub list

After removing 2 from sub list original ArrayList contains :

1

3

4

5

\*/

package com.bb.ilp.tests;

import java.util.ArrayList;

import java.util.Iterator;

import com.bb.ilp.beans.Author;

import com.bb.ilp.beans.Book;

public class SMSTest {

public static void main(String[] args) {

Book book1 = new Book(1, "Algorithms by ex", "ADA", "Cormen");

Book book2 = new Book(2, "Data Structures made easy", "Data structures", "Hewitt");

Book book3 = new Book(3, "Java Complete reference", "Java", "Herbert shield");

Author author1 = new Author(1, "Cormen", (byte) 26, "Sec 31 ggn", book1);

Author author2 = new Author(2, "Rahul", (byte) 27, "Sec 17 punjab", book2);

Author author3 = new Author(3, "Herbert shield", (byte) 35, "Sec 14 delhi", book3);

//creating Arraylist of Book

ArrayList<Book> listOfBooks = new ArrayList<>();

//add()

//Add elements== Object of book type using .add()

listOfBooks.add(book1);

listOfBooks.add(book2);

listOfBooks.add(book3);

//get()

//Accessing book object using get() starting with index 0

System.out.println(listOfBooks.get(0).getBookId());

System.out.println(listOfBooks.get(1).getBookId());

System.out.println(listOfBooks.get(2).getBookId());

//get() using loop

//enhancing above code by using for loop

for(int i=0;i<listOfBooks.size();i++) {

System.out.println(listOfBooks.get(i).getBookId());

}

//get using advanced loop

//using for-each loop

for(Book book:listOfBooks) {

System.out.println(book.getBookId());

}

//get using iterator

Iterator<Book> iterator=listOfBooks.iterator();

while(iterator.hasNext()) {

System.out.println(iterator.next().getBookId());

}

//removing book with id 2 from the list

//always use iterator for doing so

Iterator<Book> iteratorToRemove=listOfBooks.iterator();

while(iteratorToRemove.hasNext()) {

if(iteratorToRemove.next().getBookId()==2) {

iteratorToRemove.remove();

}

}

//isEmpty()

boolean isEmpty=listOfBooks.isEmpty();

if(isEmpty) {

System.out.println("List is empty");

}

else {

System.out.println("List is not empty");

}

//size()

int size=listOfBooks.size();

System.out.println("Size of listOfBooks is "+size);

//contains() , to check whether the list contains our book

if(listOfBooks.contains(book1)) {

System.out.println("yes it contains");

}

else {

System.out.println("no it doesn't");

}

//clear is used to remove all the elements

listOfBooks.clear();

//now check the whether it is cleared or not by checking size or isEmpty

if(listOfBooks.isEmpty()) {

System.out.println("List is empty");

}

else {

System.out.println("List is not empty");

}

}

}

1

2

3

1

2

3

1

2

3

1

2

3

List is not empty

Size of listOfBooks is 2

yes it contains

List is empty

package com;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

/\*\*

\* Java ArrayList Example Program

\*

\* @author Mohit

\*

\*/

public class Test {

public static void main(String args[]) {

List<String> letters = new ArrayList<String>();

// add example

letters.add("A");

letters.add("C");

letters.add("D");

// let's insert B between A and C

letters.add(1, "B");

System.out.println(letters);

List<String> list = new ArrayList<String>();

list.add("E");

list.add("H");

// appending list elements to letters

letters.addAll(list);

System.out.println(letters);

// clear example to empty the list

list.clear();

list.add("F");

list.add("G");

// inserting list inside letters to get right sequence

letters.addAll(5, list);

System.out.println(letters);

// contains example

System.out.println("Letters list contains E ? " + letters.contains("E"));

System.out.println("Letters list contains Z ? " + letters.contains("Z"));

// ensureCapacity example, it's ArrayList method, so object should be defined

// like below.

ArrayList<String> tempList = new ArrayList<>();

tempList.ensureCapacity(1000);

// get example

String e = letters.get(4);

System.out.println("Letter at 5th place: " + e);

// tempList is empty?

System.out.println("tempList is empty ? " + tempList.isEmpty());

// indexOf example

System.out.println("First index of D = " + letters.indexOf("D"));

System.out.println("Last index of D = " + letters.lastIndexOf("D"));

// remove examples

System.out.println(letters);

String removed = letters.remove(3);

System.out.println("After removing '" + removed + "' letters contains " + letters);

// remove first occurrence of H

boolean isRemoved = letters.remove("H");

System.out.println("H removed? " + isRemoved + ". Letters contains " + letters);

System.out.println("list contains " + list);

// remove all matching elements between letters and list

letters.removeAll(list);

System.out.println(letters);

// retainAll example

list.clear();

list.add("A");

list.add("B");

list.add("C");

letters.retainAll(list);

System.out.println("letters elements after retainAll operation: " + letters);

// size example

System.out.println("letters ArrayList size = " + letters.size());

// set example

letters.set(2, "D");

System.out.println(letters);

// toArray example

String[] strArray = new String[letters.size()];

strArray = letters.toArray(strArray);

System.out.println(Arrays.toString(strArray));

}

}

[A, B, C, D]

[A, B, C, D, E, H]

[A, B, C, D, E, F, G, H]

Letters list contains E ? true

Letters list contains Z ? false

Letter at 5th place: E

tempList is empty ? true

First index of D = 3

Last index of D = 3

[A, B, C, D, E, F, G, H]

After removing 'D' letters contains [A, B, C, E, F, G, H]

H removed? true. Letters contains [A, B, C, E, F, G]

list contains [F, G]

[A, B, C, E]

letters elements after retainAll operation: [A, B, C]

letters ArrayList size = 3

[A, B, D]

[A, B, D]

### Java ArrayList Iterator

Iterator is an interface in Java Collections framework. ArrayList provides fail-fast iterator implementation. When you want to perform some operation on all the list elements, you should use Iterator. If any structural modification is made to the list while iterating, it’s next() operation will throw ConcurrentModificationException. Below is a simple example of ArrayList iterator.

### Example 3: