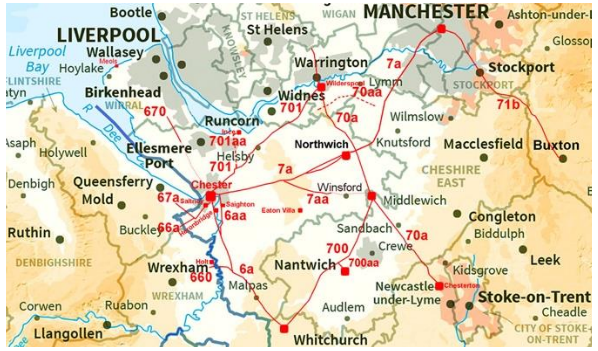
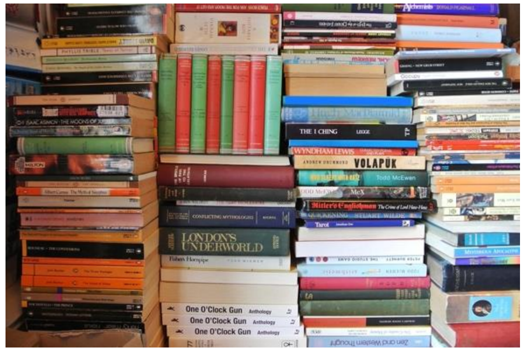
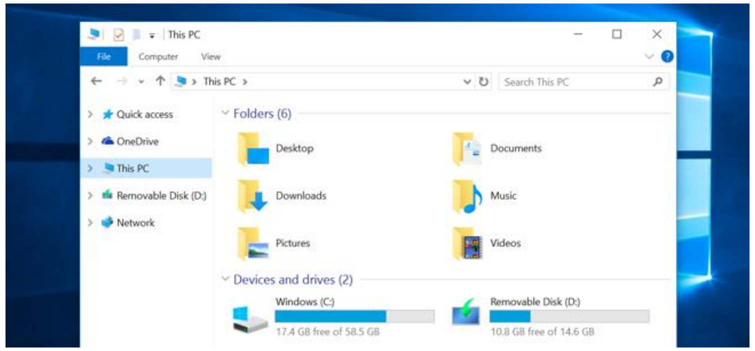
COM410 Programming in Practice

A4.3 Introduction to Java Collections

Collections



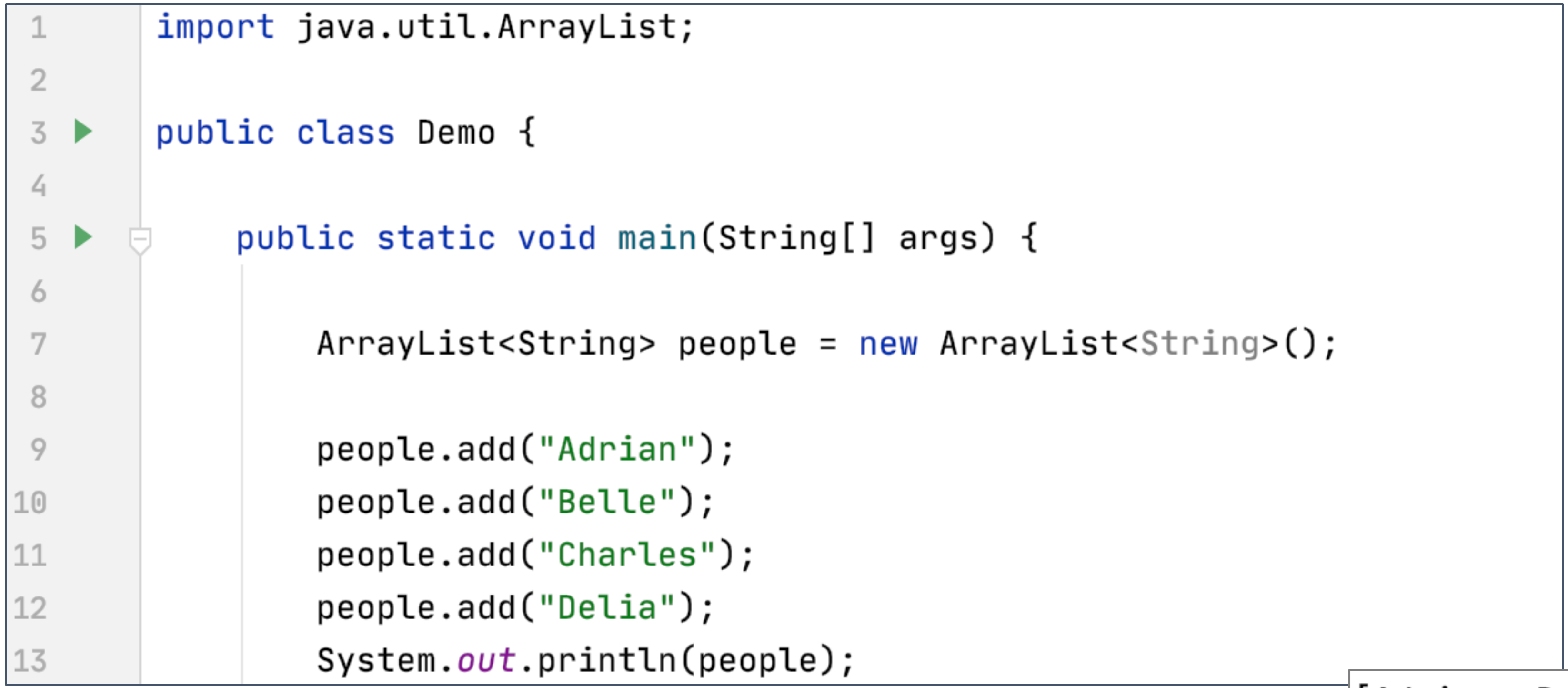
- It is important as a programmer to understand how to define collection ADTs in response to application-specific requirements

Collections in Java

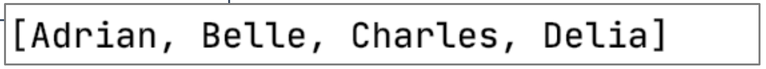
- Java provides a rich range of collection types built-in to the language, through the Java Class Library (and included through the `import` statement)
- In most circumstances, a requirement for a collection ADT that does not match exactly with one of the built-in collection types can be satisfied by using one of the built-in types as a base.
- In particular, there are two collection types available in the `java.util` library that can be seen as extensions to the basic array and linked chain structures that we used in the `Bag` implementations
    - Class `ArrayList`
    - Class `LinkedList`

Class `ArrayList`

- Provides a resizable array, removing many of the limitations of the standard `Array` type.
- `ArrayList` elements are ordered, but elements can be added by the `add()` method



```java
import java.util.ArrayList;

public class Demo {

    public static void main(String[] args) {

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

        people.add("Adrian");
        people.add("Belle");
        people.add("Charles");
        people.add("Delia");
        System.out.println(people);
```

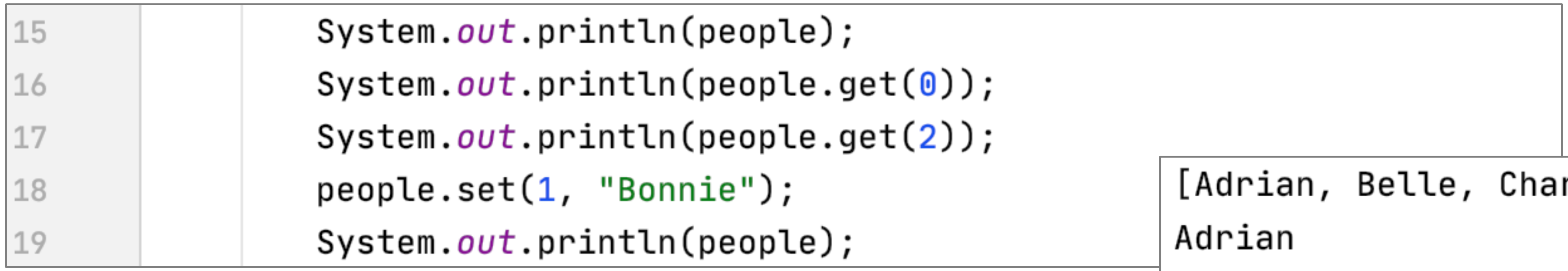


```
[Adrian, Belle, Charles, Delia]
```

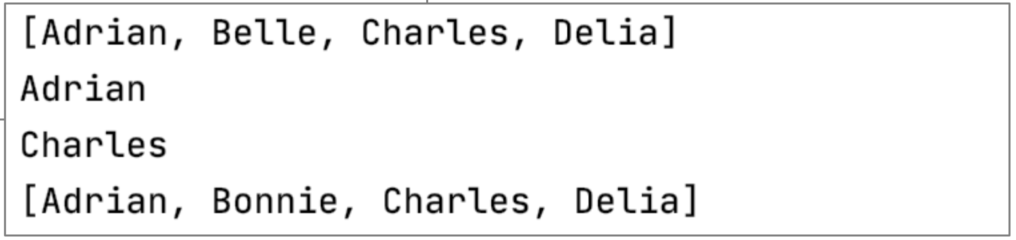
- Note that the type of the `ArrayList` is specified using the Generic notation

Class `ArrayList`

- Elements can be accessed by `get()` and `set()` methods which take an index number as a parameter



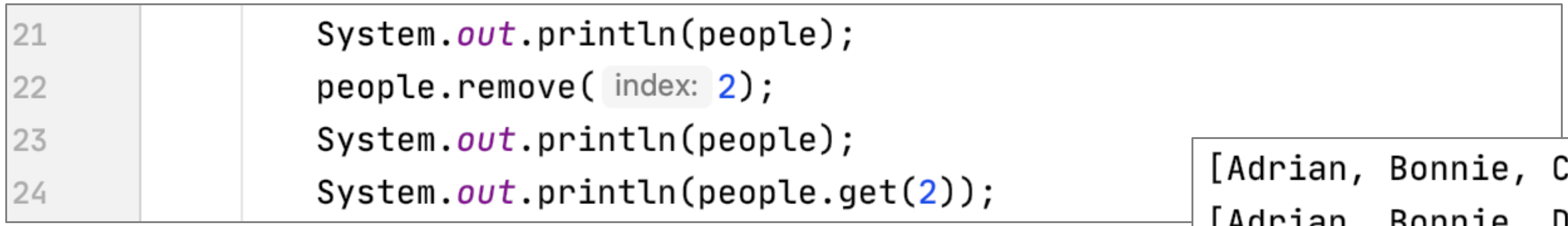
```
        System.out.println(people);
        System.out.println(people.get(0));
        System.out.println(people.get(2));
        people.set(1, "Bonnie");
        System.out.println(people);
```



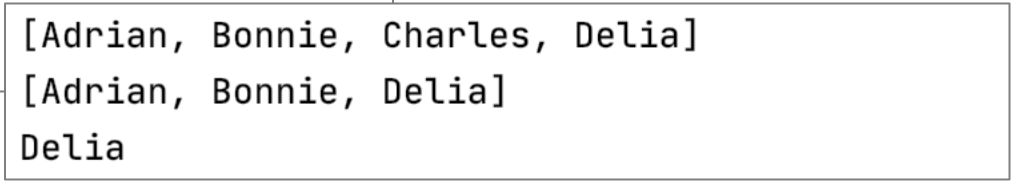
```
[Adrian, Belle, Charles, Delia]
Adrian
Charles
[Adrian, Bonnie, Charles, Delia]
```

Class `ArrayList`

- Elements can be removed by the `remove()` method which takes an index number as a parameter



```
System.out.println(people);
people.remove( index: 2);
System.out.println(people);
System.out.println(people.get(2));
```

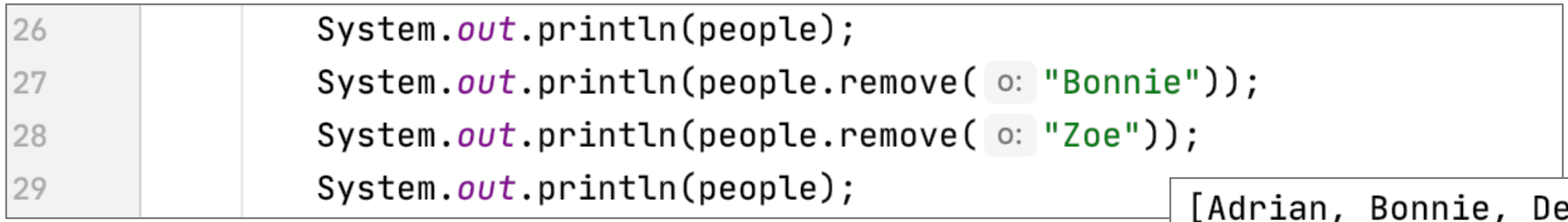


```
[Adrian, Bonnie, Charles, Delia]
[Adrian, Bonnie, Delia]
Delia
```

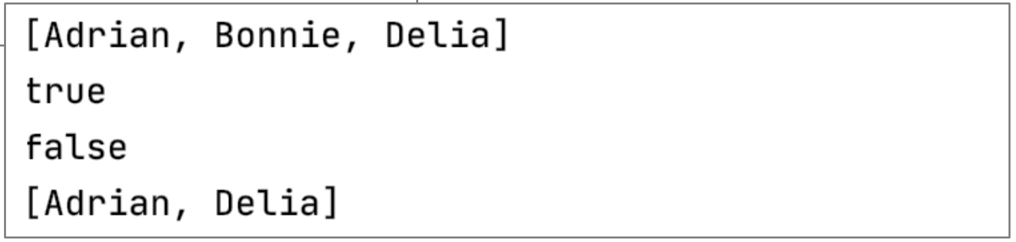
- The `remove()` method automatically closes the gaps in the array

Class `ArrayList`

- Another version of the `remove()` method takes an object as a parameter and returns true or false depending on the success of the operation



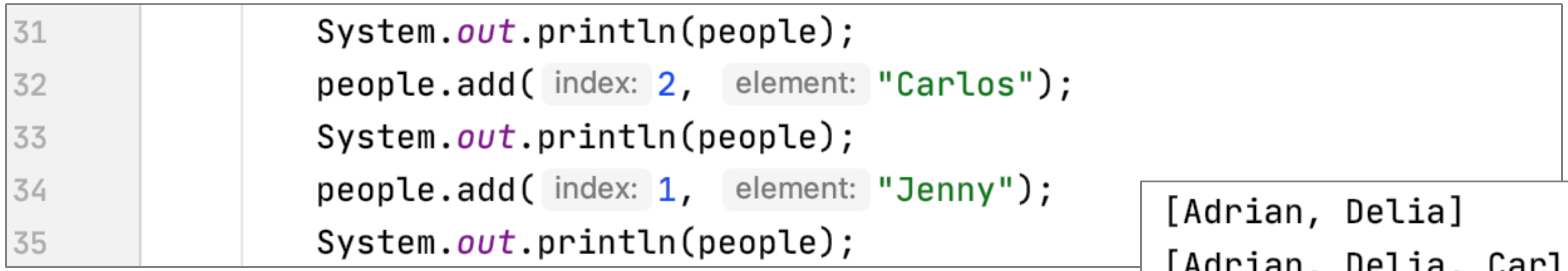
```
        System.out.println(people);
        System.out.println(people.remove( o: "Bonnie"));
        System.out.println(people.remove( o: "Zoe"));
        System.out.println(people);
```



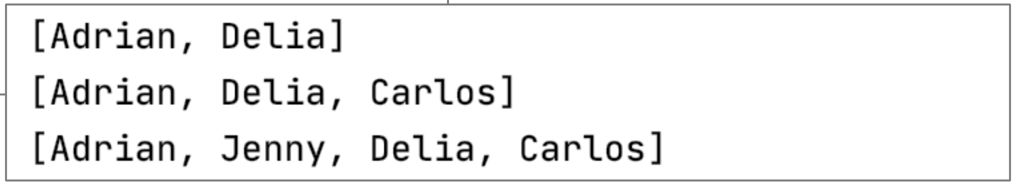
```
[Adrian, Bonnie, Delia]
true
false
[Adrian, Delia]
```

Class `ArrayList`

- Elements can be inserted at a specific position by providing a parameter to the `add()` method specifying the position at which the new element should be inserted



```
        System.out.println(people);
        people.add( index: 2,  element: "Carlos");
        System.out.println(people);
        people.add( index: 1,  element: "Jenny");
        System.out.println(people);
```

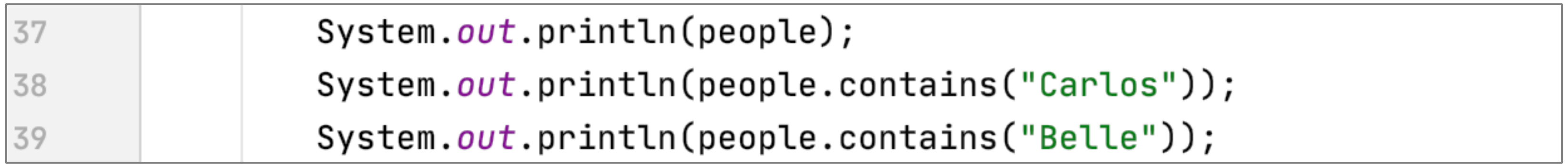


```
[Adrian, Delia]
[Adrian, Delia, Carlos]
[Adrian, Jenny, Delia, Carlos]
```

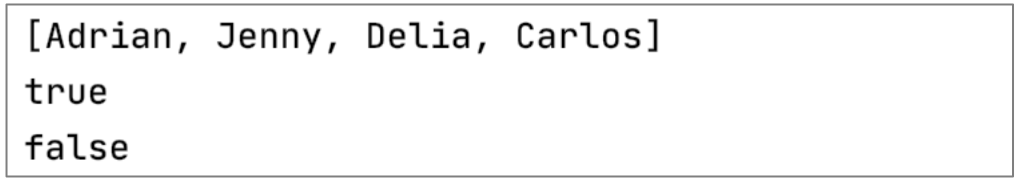
- The `add()` method automatically pushes array elements further forward

Class ArrayList

- Checking for a specific value

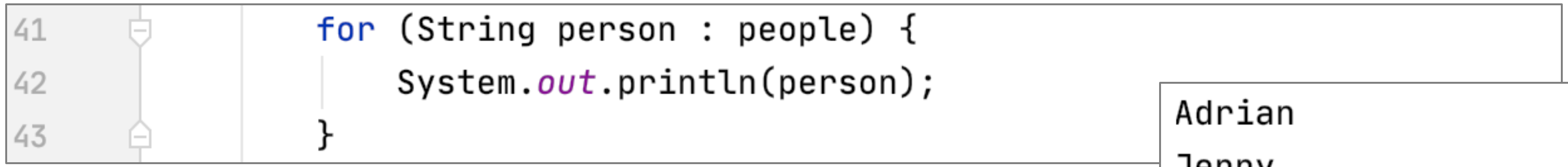


```java
System.out.println(people);
System.out.println(people.contains("Carlos"));
System.out.println(people.contains("Belle"));
```

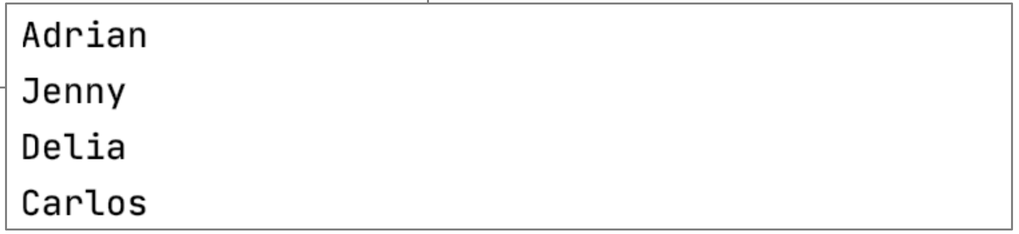


```
[Adrian, Jenny, Delia, Carlos]
true
false
```

- Looping through an ArrayList



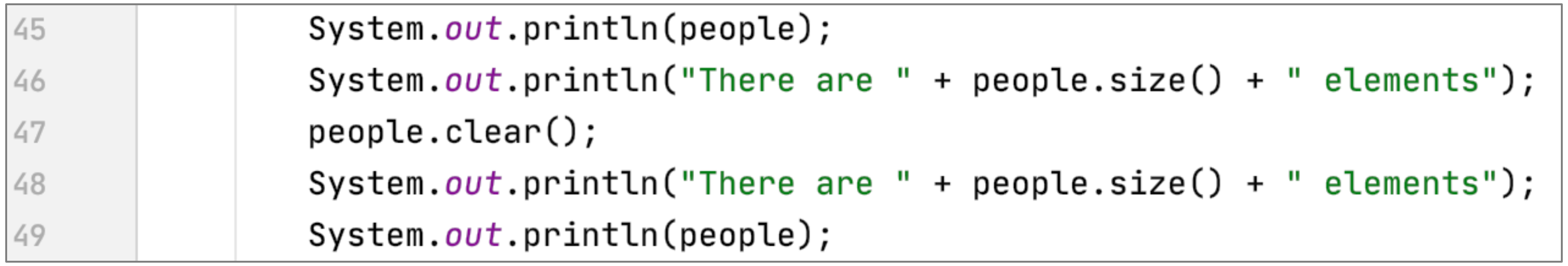
```java
for (String person : people) {
    System.out.println(person);
}
```



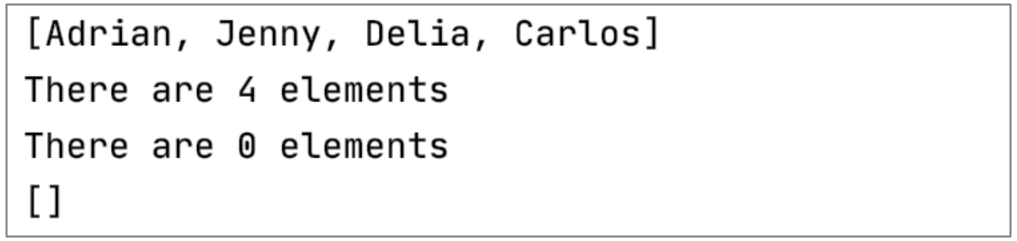
```
Adrian
Jenny
Delia
Carlos
```

Class `ArrayList`

- The `size()` method reports the number of elements
- The `clear()` method removes all elements from the collection



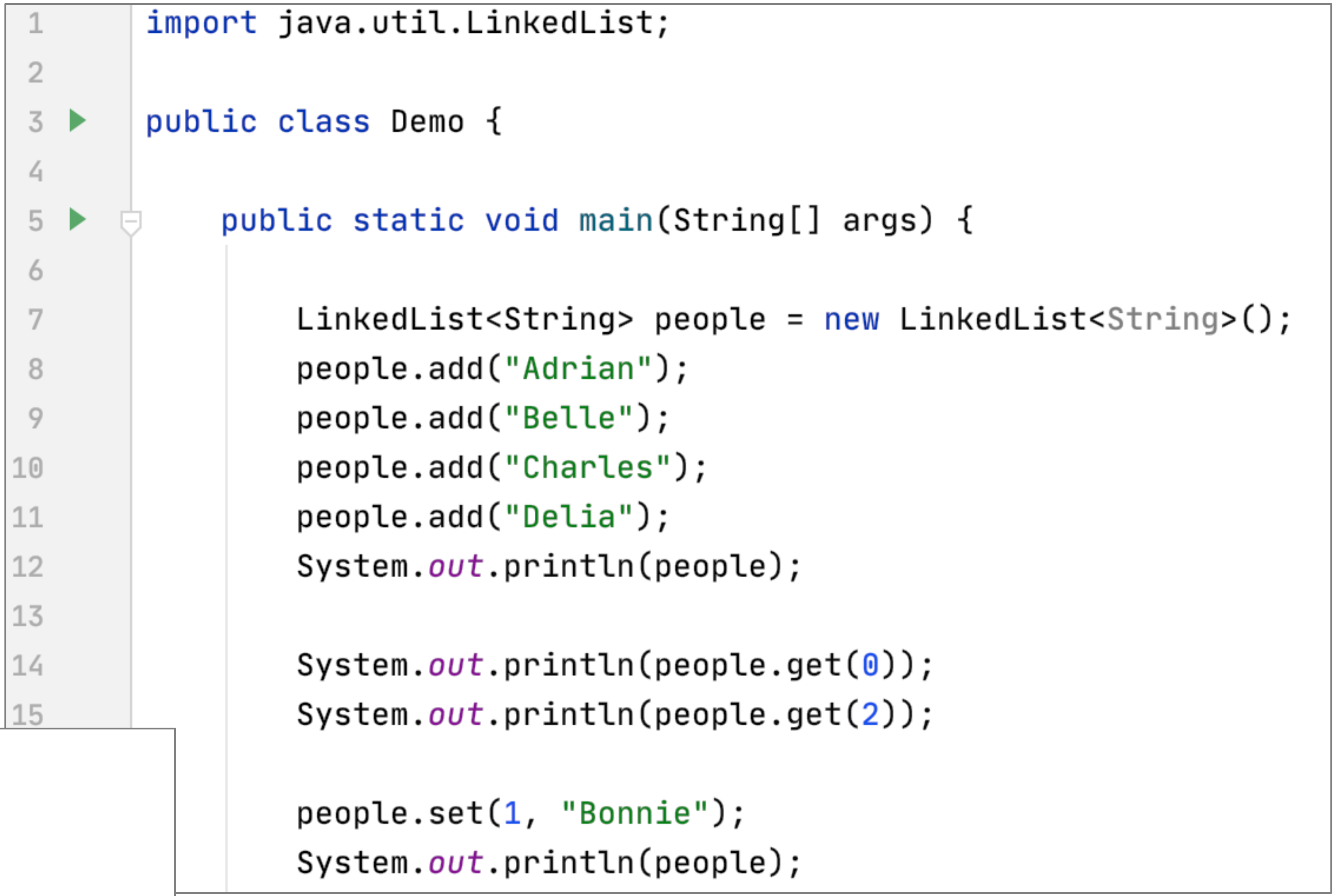
```java
        System.out.println(people);
        System.out.println("There are " + people.size() + " elements");
        people.clear();
        System.out.println("There are " + people.size() + " elements");
        System.out.println(people);
```



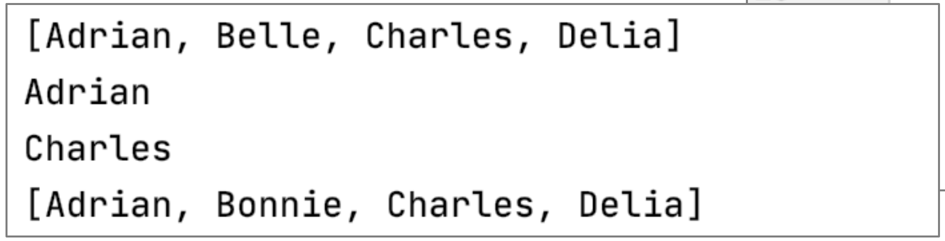
```
[Adrian, Jenny, Delia, Carlos]
There are 4 elements
There are 0 elements
[]
```

Class `LinkedList`

- Implements all of the same methods as `ArrayList`…



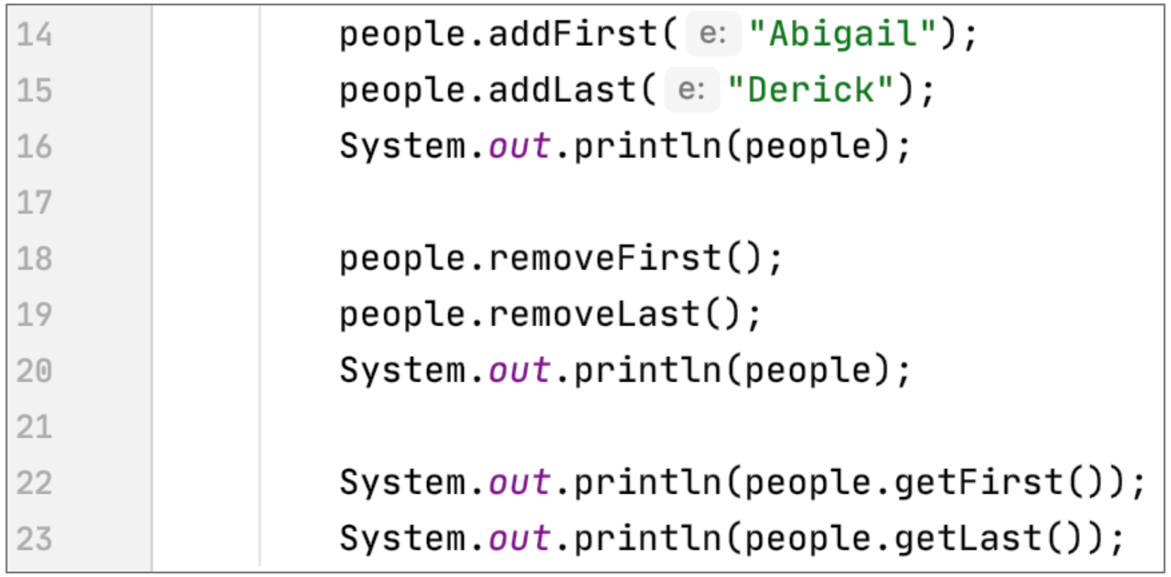
```java
import java.util.LinkedList;

public class Demo {

    public static void main(String[] args) {

        LinkedList<String> people = new LinkedList<String>();
        people.add("Adrian");
        people.add("Belle");
        people.add("Charles");
        people.add("Delia");
        System.out.println(people);

        System.out.println(people.get(0));
        System.out.println(people.get(2));

        people.set(1, "Bonnie");
        System.out.println(people);
```



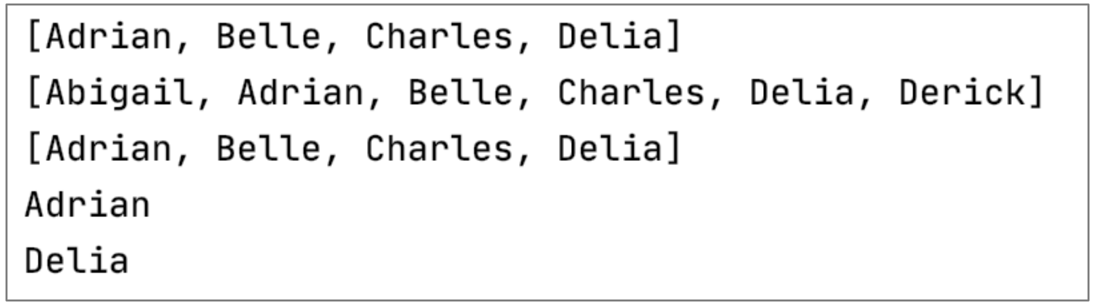
```
[Adrian, Belle, Charles, Delia]
Adrian
Charles
[Adrian, Bonnie, Charles, Delia]
```

Class `LinkedList`

- …plus additional methods at either end of the list



```
        people.addFirst( e: "Abigail");
        people.addLast( e: "Derick");
        System.out.println(people);

        people.removeFirst();
        people.removeLast();
        System.out.println(people);

        System.out.println(people.getFirst());
        System.out.println(people.getLast());
```



```
[Adrian, Belle, Charles, Delia]
[Abigail, Adrian, Belle, Charles, Delia, Derick]
[Adrian, Belle, Charles, Delia]
Adrian
Delia
```

ArrayList vs LinkedList

- The `ArrayList` class contains a regular Array type. When a new element is added, it is placed into the array. If the array is not big enough, a new one is created, the values are copied over and the old one removed.
- The `LinkedList` class is organized as a linked chain structure where each element is stored in a container. Each container stores the element value plus a link to the next container in the list.
    - `ArrayList` is best for storing and retrieving data that is fairly static in nature
    - `LinkedList` is best for manipulating data where the items stored will frequently change.

Scenario

- In your Anytown project, add a new class `ArrayListBag` that implements `BagInterface` using an `ArrayList` as the storage medium.
    - Make as much use as you can of `ArrayList` methods in your implementation
    - Modify the file ***BagOfBuildingsTest.java*** to operate on an instance of a `ArrayListBag`
    - Run the `main()` method in ***BagOfBuildingsTest.java*** and trace the diagnostic comments provided to ensure that your `ArrayListBag` implementation is working as expected

Challenge

- The Mayor of Anytown has requested that volunteers register for a "Tidy Town" project which organises litter collection and general street maintenance. For ease of reporting, all volunteers are entered in the form "Surname, Firstname" and are maintained in alphabetical order.
    - In your `Anytown` project, implement the class `TidyTown` which prompts the user to enter names until the name "xxx"(which should not be stored) is entered.
    - The names should be stored in either an `ArrayList` or `LinkedList` structure (your choice!)
    - Each name entered should be added to the structure such that the list of names is always maintained in alphabetical order.
    - Once the "xxx" value has been entered, the list of names should be displayed alphabetically by traversing the list structure.