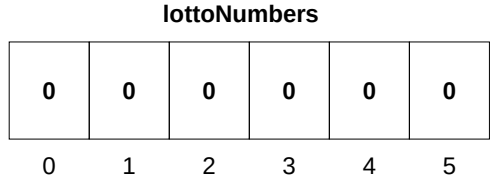
ArrayLists

Introduction to OO Programming

Array revision

- Arrays allow the programmer to work with a group of values of the same type.

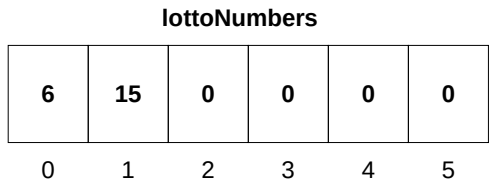
```
int[] lottoNumbers = new int [6];
```



Array revision

- Individual array elements can be accessed using the subscript or index.

```
int[] lottoNumbers = new int [6];
lottoNumbers[0] = 6;
lottoNumbers[1] = 15;
```

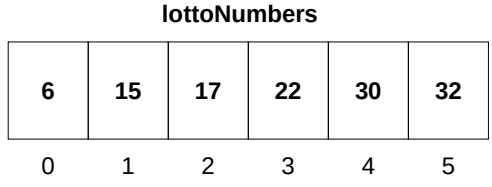


Array revision

- A for loop can be used to access all array elements in turn.

```
int[] lottoNumbers = new int[] {6, 15, 17, 22, 30, 32};

      for(int i=0; i < lottoNumbers.length; i++)
      {
         System.out.print(lottoNumbers[i] + " ");
      }
```



Array revision

- Arrays are declared to hold values of the same type. This can be a primitive type or an object type.

```
double[] prices = new double[5];

String[] theBand = new String[] {"Bob", "Joe", "Mary"};
or
String[] theBand = {"Bob", "Joe", "Mary"};

BankAccount[] accounts = new BankAccount[1000];
```

ArrayList

- The `ArrayList` class is designed to store a series or list of objects of the same type.
- An `ArrayList` is a resizable container class which provides methods for the manipulation and storage of its contents.
- An `ArrayList` can only contain objects. It cannot contain primitives.

ArrayLists

- The `ArrayList` class manages a sequence or list of objects
- Advantages over Arrays
  - Can grow and shrink as needed
  - `ArrayList` class supplies methods for many common tasks, such as inserting and removing elements

ArrayList

- Like an array, an `ArrayList` has an index which starts at 0. This index allows you to refer to any element in an `ArrayList`.
- The index in an `ArrayList` goes from 0 to n – 1 where n is the number of elements in the `ArrayList`.
- To create an `ArrayList`, you need to import the class from the package `java.util`.

```
import java.util.ArrayList;
```

ArrayList declaration

- When declaring an `ArrayList`, the type of element which it will hold must be specified.
- This is included in angle brackets directly after the word `ArrayList`:

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

ArrayList <Employee> staff = new ArrayList <Employee>();
```

- **General Form**

```
ArrayList <T> list = new ArrayList <T>();
```

where `T` is any `Class Type`

`ArrayList` methods

- When you construct an empty `ArrayList` object – it has size 0
- `add()` method adds an object to the end of the array list
- The size increases after each call to `add()`
- `size()` method returns the number of elements in list

Length and Size

The Java syntax for determining the number of elements in
an array,
an array list and
a string
is not consistent

DataType	Find no. of elements
Arrays (int[ ] values)	values.length
ArrayList list	list.size()
String s	s.length()

Adding elements

- The `add( )` method is used to add elements to an `ArrayList`

```
ArrayList <String> names = new ArrayList <String>();
String firstBeatle = new String("John");

names.add(firstBeatle);
names.add("Paul");
```

boolean	add(E e) Appends the specified element to the end of this list.

Adding elements

- The `add( )` method is overloaded to allow the insertion of elements at a particular position in an ArrayList

```
ArrayList <String> names = new ArrayList <String>();
names.add("John");
names.add("Paul");
names.add("Ringo");
names.add(2, "George");
```

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.

Retrieving elements

- The `get( )` method is used to retrieve elements from an ArrayList
- The return type of the `get( )` method is the type of element the ArrayList contains.

```
ArrayList <String> names = new ArrayList <String>();
names.add("John");
names.add("Paul");

String s = names.get(0);
Or
System.out.println(names.get(0));
```

E	get(int index) Returns the element at the specified position in this list.

The `size()` method

- The `size( )` method returns the number of elements in an ArrayList

```
ArrayList <String> names = new ArrayList <String>();
names.add("John");
names.add("Paul");

System.out.println(names.size());
Or
int noOfElements = names.size();
```

int	size() Returns the number of elements in this list.

Accessing all ArrayList elements

- A `for` loop can be used along with the `size( )` method to access all the elements in an ArrayList.

```
ArrayList <Employee> computingDept = new ArrayList
  <Employee>();

computingDept.add(firstWorker);
computingDept.add(secondWorker);
computingDept.add(thirdWorker);

    for(int i = 0; i < computingDept.size(); i++)
    {
       System.out.println(computingDept.get(i));
    }
```

This will only print if class has a `toString()`

The `set()` method

- The `set( )` method overwrites an existing element with a new one.

```
ArrayList <String> names = new ArrayList <String>();
names.add("John");
names.add("Paul");
names.add("George");
names.set(2, "Mick");
```

E	set(int index, E element) Replaces the element at the specified position in this list with the specified element.

The `remove()` method

- The `remove( )` method deletes an element at a given position.
- All other elements are moved by one space, and the size of the ArrayList is decreased by one.

```
ArrayList <Employee> computingDept = new ArrayList
  <Employee>();

computingDept.add(new Employee("Mary", 20000));
computingDept.add(new Employee("Ann", 21000));
computingDept.add(new Employee("Tom", 20000));
computingDept.remove(2);
```

E	remove(int index) Removes the element at the specified position in this list.

The `remove()` method

- The `remove( )` method is overloaded to allow you to specify the element to remove
- This is an optional operation, which means that it can be called whether or not the element is present.

```
ArrayList <Employee> computingDept = new ArrayList
  <Employee>();

computingDept.add(firstWorker);
computingDept.add(secondWorker);
computingDept.remove(firstWorker);
```

boolean	**remove**(Object o) Removes a single instance of the specified element from this list, if it is present

Manipulating retrieved elements

- The `get( )` method is used to retrieve elements from an ArrayList
- It is possible to call the methods of these objects to perform operations on the returned values

```
ArrayList <String> names = new ArrayList <String>();
names.add("John");
names.add("Paul");

System.out.println(names.get(0).length());
System.out.println(names.get(0).charAt(0));
```

Summary

- An `ArrayList` is a resizable container class which provides methods for the manipulation and storage of its contents.
- An `ArrayList` can only contain objects. It cannot contain primitives.
- Like an array, an `ArrayList` has an index which starts at 0. This allows you to refer to any element in an `ArrayList`.
- The `ArrayList` class provides methods which allow you to access and manipulate its elements.