Method from the java.util.Arrays class:

`static String toString(T[] array)`	Returns a `String` representation of any type (`T[]`) array.
`static void sort(T[] array)`	sorts the specified array in memory type T must be Comparable or Comparable`<T>`
`static int[] copyOf(int[] orig, int newLength)`	Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static <T> T[] copyOf(T[] orig, int newLength)`	Copies the specified array, truncating or padding with `null`s (if necessary) so the copy has the specified length.

Arrays: array of Objects (of type Item)

1. Declare and create an *array* named `a` that stores 8 instances of type `Item`.

2. Write code that creates and adds (assigns) some `Item`s to your *array*.
   Assume `class Item` has a constructor that requires a single `String` argument.

3. Write code that displays if two `Item` elements at index m and n are the same.
   Assume that `class Item` *@Overrides* the `boolean equals(Object o)` method.

4. Write a code fragment that displays the items in the array named `a`.
   Tip: Use a loop, or use the `Arrays` *toString* method.

5. Write a *class method* that returns the number of (non-null) items in an array of `Item`s.
   Assume: The array of items is passed in a parameter.

Methods from the java.util.ArrayList class (*REMEMBER 0-based indexing is used): Note the E's below are replaced with the particular `ArrayList`'s element type.

`int size()`	Returns the number of used elements in this list.
`E[] toArray()`	Returns an array of the specified type of this ArrayList
`boolean contains(E item)`	Returns true iff the specified item is in this list, otherwise false.
`int indexOf(E item)`	Returns the index of the specified item if it is in this list, otherwise `-1`
`E get(int index)`	Returns the item at the specified index in this list. `throws IndexOutBoundsException` if invalid index.
`void add(E item)`	Adds the specified item to the end of this list.
`void add(int index, E item)`	Adds the specified item by inserting it into this list at the specified index.
`E remove(int index)`	Removes and returns the item from the specified index.
`boolean remove(E item)`	Returns `true` iff the specified item was removed from this list, otherwise `false`.
`void sort(Comparable[] item)`	Sorts the items in the array list and places them in the parameter. It's better Returns `true` iff the specified item was removed from this list, otherwise `false`.

ArrayList: an Auto-Expanding Array

Must `import java.util.ArrayList;`

1. Declare and create an `ArrayList` named `alist` that stores instance of type `Item`.

2. Add some `Item`s to your `ArrayList`.

3. Display the number of items in the `ArrayList`.

4. Get the first item in the list.

5. Get the last item in the list.

6. Does `get(int)` remove the item from the list?

7. Declare and create an `ArrayList` of integers, and one of characters, and one of doubles..

Arrays of Objects

When an array of a reference type is created; element values are null until the individual objects are created and assigned as elements of the array.
Use brackets `[ ]` to indicate which element.

1. What is output by this code fragment if `Widget` is defined as shown? Hint: Draw the Object (memory) diagram of the state of memory allocation at the end of execution of the previous question's code fragment.

```
// Code Fragment
Widget [] w;

w = new Widget[5];

w[0] = new Widget();

w[1] = w[0];

w[2] = new Widget( "Blackberry" );

w[3] = new Widget( "iPhone" );

String s = "" + w[0];

w[4] = w[0];

w[4].setName( "Samsung" );

s += w[0].getID()+" " + w[1].getID()+" ";
s += w[2].getID()+" " + w[3].getID()+" ";
s += w[4].getID();

s += " = " + Widget.getCount();

System.out.println( s );

w[0] = new Widget();

System.out.println( Widget.getCount() );
System.out.println( w[0] == w[1] );
System.out.println( w[1] == w[4] );
System.out.println( w[3].equals( new
    Widget("iPhone") ) );

System.out.println(
    w[3].getName().equals( (new
    Widget("iPhone")).getName() ) );

System.out.println(
    w[3].getName().equals( "iPhone" ) );
```

```
public class Widget {

    public static final String COMPANY =
        "ACME";
    private static int count = 0;
    private final long ID;
    private String name;

    public Widget( String name ) {
        ID = 9000 + ++count;
        this.name = name;
    }

    public Widget() {
        this( "W" + count );
    }

    public static String getCompany() {
        return COMPANY;
    }

    public static int getCount() {
        return count;
    }

    public long getID() {
        return ID;
    }

    public String getName() {
        return name;
    }

    public void setName( String newName )
    {
        name = newName;
    }

}
```