Adding To and Removing From an Integer List

File *IntegerList.java* contains a Java class representing a list of integers. The following public methods are provided:

- IntegerList(int size)—creates a new list of *size* elements. Elements are initialized to 0.
- void randomize()—fills the list with random integers between 1 and 100, inclusive.
- void print()—prints the array elements and indices

File *IntegerListTest.java* contains a Java program that provides menu-driven testing for the IntegerList class. Copy both files to your directory, and compile and run IntegerListTest to see how it works.

It is often necessary to add items to or remove items from a list. When the list is stored in an array, one way to do this is to create a new array of the appropriate size each time the number of elements changes, and copy the values over from the old array. However, this is rather inefficient. A more common strategy is to choose an initial size for the array and add elements until it is full, then double its size and continue adding elements until it is full, and so on. (It is also possible to decrease the size of the array if it falls under, say, half full, but we won't do that in this exercise.) The CDCollection class in Listing 7.8 of the text uses this strategy—it keeps track of the current size of the array and the number of elements already stored in it, and method *addCD* calls *increaseSize* if the array is full. Study that example.

1. Add this capability to the IntegerList class. You will need to add an *increaseSize* method plus instance variables to hold the current number of integers in the list and the current size of the array. Since you do not have any way to add elements to the list, you won't need to call *increaseSize* yet.

2. Add a method *void addElement(int newVal)* to the IntegerList class that adds an element to the list. At the beginning of *addElement*, check to see if the array is full. If so, call *increaseSize* before you do anything else.

   Add an option to the menu in IntegerListTest to test your new method.

3. Add a method *void removeFirst(int newVal)* to the IntegerList class that removes the first occurrence of a value from the list. If the value does not appear in the list, it should do nothing (but it's not an error). Removing an item should not change the size of the array, but note that the array values do need to remain contiguous, so when you remove a value you will have to shift everything after it down to fill up its space. Also remember to decrement the variable that keeps track of the number of elements.

   Add an option to the menu in IntegerListTest to test your new method.

4. Add a method *removeAll(int newVal)* to the IntegerList class that removes all occurrences of a value from the list. If the value does not appear in the list, it should do nothing (but it's not an error).

Add an option to the menu in IntegerListTest to test your new method.

```
// ****************************************************************

// IntegerList.java

//

// Define an IntegerList class with methods to create & fill

// a list of integers.

//

// ****************************************************************


public class IntegerList

{

    int[] list; //values in the list


    //-------------------------------------------------------

    //create a list of the given size

    //-------------------------------------------------------

    public IntegerList(int size)

    {

      list = new int[size];

    }



    //-------------------------------------------------------

    //fill array with integers between 1 and 100, inclusive

    //-------------------------------------------------------

    public void randomize()

    {

      for (int i=0; i<list.length; i++)

          list[i] = (int)(Math.random() * 100) + 1;
    }

    //-------------------------------------------------------

    //print array elements with indices

    //-------------------------------------------------------

    public void print()

    {

      for (int i=0; i<list.length; i++)

          System.out.println(i + ":\t" + list[i]);

    }

}




// ****************************************************************

// IntegerListTest.java

//

// Provide a menu-driven tester for the IntegerList class.

//

// ****************************************************************

import java.util.Scanner;


public class IntegerListTest

{

    static IntegerList list = new IntegerList(10);

    static Scanner scan = new Scanner(System.in);


    //-------------------------------------------------------

    // Create a list, then repeatedly print the menu and do what the

    // user asks until they quit

    //-------------------------------------------------------
```

```
public static void main(String[] args)
{
  printMenu();
  int choice = scan.nextInt();
  while (choice != 0)
      {
        dispatch(choice);
        printMenu();
        choice = scan.nextInt();
      }
}

//--------------------------------------
// Do what the menu item calls for
//--------------------------------------
public static void dispatch(int choice)
{
  int loc;
  switch(choice)
      {
      case 0:
        System.out.println("Bye!");
        break;
      case 1:
        System.out.println("How big should the list be?");
        int size = scan.nextInt();
        list = new IntegerList(size);
        list.randomize();
        break;
      case 2:
        list.print();
```

```
            break;
          default:
            System.out.println("Sorry, invalid choice");
          }
    }


    //----------------------------
    // Print the user's choices
    //----------------------------
    public static void printMenu()
    {
      System.out.println("\n   Menu   ");
      System.out.println("   ====");
      System.out.println("0: Quit");
      System.out.println("1: Create a new list (** do this first!! **)");
      System.out.println("2: Print the list");
      System.out.print("\nEnter your choice: ");
    }
}
```