Tunbutr, Bryant

CSCI 220
Data Structures I

Lab Project #1

INTEGER SETS

Due Date
9/17/2013
Date Turned In
9/17/2013

Student Name:	____________ Bryant Tunbutr ______________________	Project Number:	________1_________
Project Name:	________IntegerSet_____	Eclipse Version:	___Kepler_________

Files: IntegerSetTest.java, IntegerSet.java

Project specification

This software is intended to manipulate data structures, specifically in a binary format of true or false using arrays.

The user may input numbers that get stored in two separate sets, that are then compared to each other.

Program is complete except I was unable to figure out how to do subsets.

I tried using while loops and searching for the not equal item, I tried using nested loops and looking for where it is not equal, but no success.

I think I learned many lessons:

It is very important to work on a big project like this one step at a time, i.e. first make sure the program can take in and display numbers. Then make sure it can do this for 2 sets. Work in small steps.

Write notes about what each method does.

Label every variable as clearly as possible to avoid confusion and work better.

I also learned that this project is an introduction to data structures.

I say this because the program runs very slowly and is inefficient.

I did research and saw that there are many other methods of storing and retrieving and using data such as HashSet, TreeSet and LinkedHashSet.

There are also Array Lists that have built in functionality.

A much better way to store this data is Linked Lists. This has pros such as it is easier to remove and add items and not having to worry about the order, i.e. with an array or vector the entire order would be changed from 2 to 1, 55 to 54, etc.

Search rate is also linear

The drawback is if there are a million or billion links, that would take longer to search.

But for this type of project, I would rather use a linked list.

Printout of program input/output

Input 1 for set 1
Input 1, 3 for set 2

Output

```
Test Case 1 with user input

Set 1 is made of elements :
{1, }

 Set 2 has elements:
{1, 3, }

 The union of the 2 sets is :
{1, 3, }

 The intersection of the 2 sets is :
{1, }

 Set 1 is NOT equal to Set 2


 Inserting 66 into Set 2


 Set 2 has elements
{1, 3, 66, }

 Deleting 66 from Set 2


 Set 2 has elements
{1, 3, }

 Test Case 2 with arrays

{1, 7, 9, 13, 23, 45, 55, 88, 111, }

{1, 7, 9, 23, 55, }


 Set 1 is made of elements :
{1, 7, 9, 13, 23, 45, 55, 88, 111, }

 Set 2 has elements:
{1, 7, 9, 23, 55, }
```

```
 The union of the 2 sets is :
{1, 7, 9, 13, 23, 45, 55, 88, 111, }

 The intersection of the 2 sets is :
{1, 7, 9, 23, 55, }

 Test Case 3 with same arrays

{1, 2, 4, 8, 16, 32, }

{1, 2, 4, 8, 16, 32, }



 The union of the 2 sets is :
{1, 2, 4, 8, 16, 32, }

 The intersection of the 2 sets is :
{1, 2, 4, 8, 16, 32, }



 Set 1 is EQUAL to Set 2


 Test Case 4 with almost same arrays

{2, 4, 8, 16, 32, 64, }

{1, 2, 4, 8, 16, 32, 64, 127, }



 Inserting 1 into Set 1


 Set 1 has elements
{1, 2, 4, 8, 16, 32, 64, }

 Deleting 127 from Set 2


 Set 2 has elements
{1, 2, 4, 8, 16, 32, 64, }

 Set 1 is EQUAL to Set 2
```

Source code

IntegerSetTest.java

```
import javax.swing.*;

import java.util.*;

public class IntegerSetTest
{
   private IntegerSet setOfIntegers1, setOfIntegers2, setOfIntegers3,
setOfIntegers4;
   private IntegerSet setOfIntegers7, setOfIntegers8;
   private IntegerSet setOfIntegers9, setOfIntegers10, setOfIntegers11,
setOfIntegers12;
   private IntegerSet setOfIntegers13, setOfIntegers14;

   private String displayString;

   // default constructor (empty set)
   public IntegerSetTest()
   {
      setOfIntegers1 = new IntegerSet();
      setOfIntegers2 = new IntegerSet();

      displayString = "";
   }

   public void initialize(){
      // prompt for set 1 values
      JOptionPane.showMessageDialog(null, "Enter Set 1");

      // run method and store values
      setOfIntegers1.IntegerParameter();

      // prompt for set 2 values
      JOptionPane.showMessageDialog(null, "Enter Set 2");

      // run method and store values
      setOfIntegers2.IntegerParameter();

      // run method for union
      setOfIntegers3 = setOfIntegers1.union2(setOfIntegers2);

      // run method intersection
      setOfIntegers4 = setOfIntegers1.intersect2(setOfIntegers2);

      // create string for display
      displayString += "Test Case 1 with user input " + "\n \n" +
      "Set 1 is made of elements :\n" + setOfIntegers1.print() +
            "\n\n Set 2 has elements: \n" + setOfIntegers2.print() +
            "\n\n The union of the 2 sets is : \n" + setOfIntegers3.print() +
            "\n\n The intersection of the 2 sets is : \n" +
setOfIntegers4.print();

      // test if 2 sets equal
      if ( setOfIntegers1.isEqual(setOfIntegers2))
      {
         displayString +=
         "\n\n Set 1 is EQUAL to Set 2 \n";
      }
      else
      {
         displayString +=
         "\n\n Set 1 is NOT equal to Set 2 \n";
      }

      // test for insertion & deletion
      displayString +=
            "\n\n Inserting 66 into Set 2 \n";
      setOfIntegers2.insertElement(66);

      displayString +=
            "\n\n Set 2 has elements \n" + setOfIntegers2.print();

      displayString +=
            "\n\n Deleting 66 from Set 2 \n";
      setOfIntegers2.removeElement(66);

      displayString +=
            "\n\n Set 2 has elements \n" + setOfIntegers2.print();

      // now testing constructor
      int[] arrayInt = {23, 7, 9, 13, 45, 88, 111, 55, 129, 1};
      IntegerSet setOfIntegers5 = new IntegerSet(arrayInt);

      int[] arrayInt2 = {23, 7, 9, 55, 129, 1};
      IntegerSet setOfIntegers6 = new IntegerSet(arrayInt2);

      displayString +=
            "\n\n Test Case 2 with arrays \n \n" + setOfIntegers5.print() + "\n\n"
+
      setOfIntegers6.print()+ "\n\n";

      // run method for union
      setOfIntegers7 = setOfIntegers5.union2(setOfIntegers6);

      // run method intersection
      setOfIntegers8 = setOfIntegers5.intersect2(setOfIntegers6);

      // create string for display
      displayString += "\n Set 1 is made of elements :\n" + setOfIntegers5.print()
+
            "\n\n Set 2 has elements: \n" + setOfIntegers6.print() +
            "\n\n The union of the 2 sets is : \n" + setOfIntegers7.print() +
            "\n\n The intersection of the 2 sets is : \n" +
setOfIntegers8.print();

      // now testing constructor
      int[] arrayInt3 = {1, 2, 4, 8, 16, 32};
      IntegerSet setOfIntegers9 = new IntegerSet(arrayInt3);

      int[] arrayInt4 = {1, 2, 4, 8, 16, 32};
      IntegerSet setOfIntegers10 = new IntegerSet(arrayInt4);

      displayString +=
            "\n\n Test Case 3 with same arrays \n \n" + setOfIntegers9.print() +
"\n\n" +
                  setOfIntegers10.print()+ "\n\n";

      // run method for union
      setOfIntegers11 = setOfIntegers9.union2(setOfIntegers10);

      // run method intersection
      setOfIntegers12 = setOfIntegers9.union2(setOfIntegers10);

      // create string for display
      displayString +=
            "\n\n The union of the 2 sets is : \n" + setOfIntegers11.print() +
            "\n\n The intersection of the 2 sets is : \n" +
setOfIntegers12.print()+"\n\n";

      // test if 2 sets equal
      if ( setOfIntegers9.isEqual(setOfIntegers10))
      {
         displayString +=
         "\n\n Set 1 is EQUAL to Set 2 \n";
      }
      else
      {
         displayString +=
         "\n\n Set 1 is NOT equal to Set 2 \n";
      }

      // now testing constructor
      int[] arrayInt5 = {2, 4, 8, 16, 32, 64};
      IntegerSet setOfIntegers13 = new IntegerSet(arrayInt5);

      int[] arrayInt6 = {1, 2, 4, 8, 16, 32, 64, 127};
      IntegerSet setOfIntegers14 = new IntegerSet(arrayInt6);

      displayString +=
            "\n\n Test Case 4 with almost same arrays \n \n" +
setOfIntegers13.print() + "\n\n" +
                  setOfIntegers14.print()+ "\n\n";

      // test for insertion & deletion
      displayString +=
            "\n\n Inserting 1 into Set 1 \n";
      setOfIntegers13.insertElement(1);

      displayString +=
            "\n\n Set 1 has elements \n" + setOfIntegers13.print();

      displayString +=
            "\n\n Deleting 127 from Set 2 \n";
      setOfIntegers14.removeElement(127);

      displayString +=
           "\n\n Set 2 has elements \n" + setOfIntegers14.print();

      // test if 2 sets equal
      if ( setOfIntegers13.isEqual(setOfIntegers14))
      {
         displayString +=
         "\n\n Set 1 is EQUAL to Set 2 \n";
      }
      else
      {
         displayString +=
         "\n\n Set 1 is NOT equal to Set 2 \n";
      }
      System.out.print(displayString);
   }

   public static void main(String[] args)
   {
      // run application
      IntegerSetTest myProgram = new IntegerSetTest();
      myProgram.initialize();
   }
}
```

IntegerSet.java

```
//folder/Project name: IntegerSet
//Programmer name: Bryant Tunbutr
//Date: 9/12/13
//Class name: IntegerSet
/*Project Description: This project will add, remove,
 * find the intersection, union, and whether the
 * set is a subset of the other set
*/

import java.util.Scanner;

import javax.swing.*;

import org.omg.CORBA.PUBLIC_MEMBER;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;

public class IntegerSet {

      // declare variables
      boolean[] setOfIntegersArrayBool;

      // declare constants
      final int MINIMUM_ARRAY_ELEMENT_INT = 0;
      final int ARRAY_SIZE_INT = 128;

      // three overload constructors
      // default constructor (empty set)
      public IntegerSet()
      {
         // declare array of booleans (true or false, 0 or 1 that space
         // is being held
         setOfIntegersArrayBool = new boolean[ARRAY_SIZE_INT];
      }

      // constructor with an integer parameter (set with one integer element)
      public void IntegerParameter()
      {
         // store user entered integer
         int userEnteredElementInt;

         // store integers from user into set
         do {
            // prompt user to enter integer or escape
            userEnteredElementInt = Integer.parseInt(JOptionPane.showInputDialog(
                  "Enter an integer : " + '\n'+ "or enter -1 to escape"));
            // check whether number is valid
            if (isItLegalToInsertOrDeleteElement(userEnteredElementInt))

               // if valid, add to array
               setOfIntegersArrayBool[userEnteredElementInt] = true;
         }
         // exit loop when -1 entered
         while (userEnteredElementInt != -1);
      }

      //  constructor with an integer array parameter (set with a list of 
elements)
      public IntegerSet( int arrayInt[] )
      {
         // array of booleans
         setOfIntegersArrayBool = new boolean[ARRAY_SIZE_INT];

         // run loop for length of array
         for(int i = 0; i < arrayInt.length; i++)
         {
            // insert element into array
            insertElement( arrayInt[i]);
         }
      }

      // checks whether it is legal to insert a user defined element
      public boolean isItLegalToInsertOrDeleteElement(int userEnteredElementInt)
      {
         // if it is between 0 and 127 it will return boolean of true
         return userEnteredElementInt >= MINIMUM_ARRAY_ELEMENT_INT && 
userEnteredElementInt <= ARRAY_SIZE_INT;
      }

      // inserts integer array (set with a list of elements)
      public void insertElement(int arrayIntegerToBeInsertedInt)
      {
         // first check to see if legal to insert integer
         if (isItLegalToInsertOrDeleteElement(arrayIntegerToBeInsertedInt))

            // if yes insert into array
            setOfIntegersArrayBool[arrayIntegerToBeInsertedInt] = true;
      }

      // deletes integer from array
      public void removeElement(int arrayIntegerToBeDeletedInt)
      {
         // first check to see if legal to insert integer
         if (isItLegalToInsertOrDeleteElement(arrayIntegerToBeDeletedInt))

            // if yes insert into array
            setOfIntegersArrayBool[arrayIntegerToBeDeletedInt] = false;
      }

      // prints a set as a list of elements in ascending order separated by commas
      public String print()
      {
         // what displays if array is empty
         String arrayString = "";

         // run loop for entire potential length of array
         for(int i = MINIMUM_ARRAY_ELEMENT_INT; i < ARRAY_SIZE_INT; i++)
         {
            // if there are values in the array
            if (setOfIntegersArrayBool[i]){

               // add each number with a comma
               arrayString += i + ", ";
            }
         }
         // finalize String
         arrayString = "{" + arrayString + "}";

         return arrayString; // return
      }

      // add elements from both sets
      public IntegerSet union2(IntegerSet integerSet)
      {
         IntegerSet union2Integers = new IntegerSet();

         // run loop for entire potential length of array
         for(int i = MINIMUM_ARRAY_ELEMENT_INT; i < ARRAY_SIZE_INT; i++)

            // gather every element in each array
            if (setOfIntegersArrayBool[i] || integerSet.setOfIntegersArrayBool[i])

               // add every element in each array to union2
               union2Integers.setOfIntegersArrayBool[i] = true;

         // return these integers
         return union2Integers;
      }

      // return elements that are common to both sets
      public IntegerSet intersect2(IntegerSet integerSet)
      {
         IntegerSet intersect2Integers = new IntegerSet();

         // run loop for entire potential length of array
         for(int i = MINIMUM_ARRAY_ELEMENT_INT; i < ARRAY_SIZE_INT; i++)

            // compare every element in each array & check for inequality
            if (setOfIntegersArrayBool[i] && integerSet.setOfIntegersArrayBool[i])


               // add every element in each array to intersect2
               intersect2Integers.setOfIntegersArrayBool[i] = true;

         // return these integers
         return intersect2Integers;
      }

      // checks whether two sets are equal
      public boolean isEqual(IntegerSet integerSet)
```

```
      {
         // run loop for entire potential length of array
         for(int i = MINIMUM_ARRAY_ELEMENT_INT; i < ARRAY_SIZE_INT; i++)

            // compare every element in each array & check for inequality
            if (setOfIntegersArrayBool[i] != integerSet.setOfIntegersArrayBool[i])

               return false; // if not equal return false for boolean

         return true; // sets are equal because loop has exited
      }
}
```