CS 435 Project 1

Brandon Ingli

February 7, 2020

FindCommonTime.java

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Finds a common time among several people.
 * @author Brandon Ingli
 * @version 1.0
 */
public class FindCommonTime {

  public static void main(String[] args) {

    if (args.length != 1) {
      System.out.println("Usage:  FindCommonTime [file name]");
      System.exit(-1);
    }

    // Open the file for reading
    String filename = args[0];
    File inputFile = new File(filename);
    Scanner input = null;
    try {
      input = new Scanner(inputFile);
    } catch (FileNotFoundException e) {
      System.out.println("Oops! File could not be opened.");
      e.printStackTrace();
      System.exit(-1);
    }

    ListsData data = new ListsData(); // Shared Memory

    // Read in the first list
```

```
35      int numFirstList = input.nextInt();
36      ArrayList<Integer> firstList = new ArrayList<Integer>();
37      for (int i = 0; i < numFirstList; i++) {
38        firstList.add(input.nextInt());
39      }
40
41      // Add lists as they appear
42      while (input.hasNextInt()) {
43        ArrayList<Integer> list = new ArrayList<Integer>();
44        int numOfInts = input.nextInt();
45        for (int j = 0; j < numOfInts; j++) {
46          list.add(input.nextInt());
47        }
48        data.addToLists(list);
49      }
50
51      // Create frames and threads
52      ArrayList<Thread> threads = new ArrayList<Thread>();
53      for (int i = 0; i < firstList.size(); i++) {
54        SearchLists sl = new SearchLists(data, firstList.get(i));
55        Thread t = new Thread(sl);
56        threads.add(t);
57      }
58
59      // Start the threads
60      for (int i = 0; i < threads.size(); i++) {
61        threads.get(i).start();
62      }
63
64      // Wait for the threads to finish
65      for (int i = 0; i < threads.size(); i++) {
66        try {
67          threads.get(i).join();
68        } catch (InterruptedException e) {
69          System.out.println("Oops. An InterruptedException has
                ↪ occurred!");
70          e.printStackTrace();
71          System.exit(-1);
72        }
73      }
74
75      // If no common meeting time, announce that fact.
76      if(!data.getFound()){
77        System.out.println("There is no common meeting time.");
78      }
79    }
80  }
```

SearchLists.java

```
1  import java.util.ArrayList;
2  /**
3   * Searches a number of lists for a common variable.
4   * @author Brandon Ingli
5   * @version 1.0
6   */
7  public class SearchLists implements Runnable{
8    /** Shared Memory */
9    private ListsData data;
10   /** Number to find in common between the data.lists */
11   private int toFind;
12
13   /**
14    * Constructor for a SearchLists object/"frame"
15    * @param data reference to shared memory
16    * @param toFind time to find in common
17    */
18   public SearchLists(ListsData data, int toFind){
19     this.data = data;
20     this.toFind = toFind;
21   }
22
23   public void run(){
24     boolean stillValid = true;
25     int i = 0;
26
27     // Go through each list while there is the possibility of a
       ↪ common time
28     while (stillValid && i < data.getLists().size()){
29       ArrayList<Integer> l = data.getListAt(i);
30       boolean inList = false;
31
32       // Go through the list, stopping if we find the common time
33       int j = 0;
34       while (!inList && j < l.size()){
35         inList = (l.get(j) == toFind);
36         j++;
37       }
38
39       stillValid = inList; // As long as we keep finding values,
           ↪ it still
40                            // holds that we have a valid common
                                 ↪ value.
41       i++;
42     }
```

```
43
44      // If we found a common time, announce that fact.
45      if(stillValid){
46        data.setFound(true);
47        System.out.println(toFind + " is a common meeting time.");
48      }
49    }
50  }
```

ListsData.java

```java
import java.util.ArrayList;

/**
 * Shared Data for the SearchLists Threads
 * @author Brandon Ingli
 * @version 1.0
 */
public class ListsData {
  /** True if a common time was found by one of the Threads */
  private boolean found = false;
  /** The lists other than the first one. */
  private ArrayList<ArrayList<Integer>> lists = new
     ↪ ArrayList<ArrayList<Integer>>();

  /** Set the value of the found variable */
  public void setFound(boolean val){
    found = val;
  }
  /**
   * Get the value of found
   * @return found
   */
  public boolean getFound(){
    return found;
  }

  /**
   * Add an ArrayList of Integers to the lists
   * @param list ArrayList<Integer> to add to lists
   */
  public void addToLists(ArrayList<Integer> list){
    lists.add(list);
  }
  /**
   * Gets the "Matrix" of lists
   * @return ArrayList<ArrayList<Integer>> of lists
   */
  public ArrayList<ArrayList<Integer>> getLists(){
    return lists;
  }
  /**
   * Gets one particular ArrayList from the "Matrix"
   * @param pos int position of the list to get
   * @return ArrayList<Integer> at position pos in lists
   */
```

```
45    public ArrayList<Integer> getListAt(int pos){
46      return lists.get(pos);
47    }
48    /**
49     * Resets the "Matrix" back to a new object
50     */
51    public void resetLists(){
52      lists = new ArrayList<ArrayList<Integer>>();
53    }
54  }
```