Sarai Gould & Laura Cruz-Albrecht
CS 106A

Section #6
August 2, 2019

Solutions to Section #6

1. How Unique!

```
import acm.program.*;
import java.util.*;

public class UniqueNames extends ConsoleProgram {

   public void run() {
      ArrayList<String> list = new ArrayList<String>();
      while (true) {
         String name = readLine("Enter name: ");
         if (name.equals("")) {
            break;
         }
         if (!list.contains(name)) {
            list.add(name);
         }
      }

      println("Unique name list contains:");
      printList(list);
   }

   /* Prints out contents of ArrayList, one element per line */
   private void printList(ArrayList<String> list) {
      for(int i = 0; i < list.size(); i++) {
         println(list.get(i));
      }
   }
}
```

2. Remove Even Length

```
public void removeEvenLength(ArrayList<String> list) {
      for (int i = list.size() - 1; i >= 0; i--) {
            if (list.get(i).length() % 2 == 0) {
                  list.remove(i);
            }
      }
}
```

3. Switch Pairs

```
private void switchPairs(ArrayList<String> list) {
      for (int i = 0; i < list.size() - 1; i += 2) {
            String first = list.remove(i);
            list.add(i + 1, first);
      }
}
```

4. Name Counts

```
/* File: CountNames.java
 * --------------------
 * This program shows an example of using a HashMap.  It reads a
 * list of names from the user and list out how many times each name
 * appeared in the list.
 */
import acm.program.*;
import java.util.*;

public class CountNames extends ConsoleProgram {
    public void run() {
        HashMap<String,Integer> nameMap = new
            HashMap<String,Integer>();
        readNames(nameMap);
        printMap(nameMap);
    }

    /*
     * Reads a list of names from the user, storing names and how many
     * times each appeared in the map that is passed in as a parameter.
     */
    private void readNames(HashMap<String,Integer> map) {
        while (true) {
            String name = readLine("Enter name: ");
            if (name.equals("")) {
                break;
            }

            /* See if that name previously appeared in the map.  Update
             * count if it did, or create a new count if it didn't.
             */
            if (map.containsKey(name)) {
                // auto-unboxing: gets an int instead of Integer
                int oldCount = map.get(name);
                // auto-boxing: convert int to Integer automatically
                map.put(name, oldCount + 1);
            } else {
                // auto-boxing: convert int to Integer automatically
                map.put(name, 1);
            }
        }
    }

    /*
     * Prints out list of entries (and associated counts) from the map
     * that is passed in as a parameter.
     */
    private void printMap(HashMap<String,Integer> map) {
        for (String key : map.keySet()) {
            int count = map.get(key);  // auto-unboxing
            println("Entry [" + key + "] has count " + count);
        }
    }
}
```

5. Mutual Friends

```
private HashMap<String, Integer> mutualFriends(
        HashMap<String, Integer> phonebook1,
        HashMap<String, Integer> phonebook2) {

    HashMap<String, Integer> result =
            new HashMap<String, Integer>();

    for (String name : phonebook1.keySet()) {
        int phoneNum = phonebook1.get(name);
        if (phonebook2.containsKey(name) &&
                phoneNum == phonebook2.get(name)) {
            result.put(name, phoneNum);
        }
    }
    return result;
}
```

6. Reverse

```
private HashMap<String, Integer> reverse(HashMap<Integer, String> map){
    HashMap<String, Integer> result = new HashMap<String, Integer>();
    for (int key : map.keySet()) {
        String value = map.get(key);
        result.put(value, key);
    }
    return result;
}
```

7. Student

```
// Student object reps. a Stanford student w/ name, ID, and unit count.
public class Student {
    private String name;            /* The student's name */
    private int ID;                 /* The student's ID number */
    private double unitsEarned;     /* number of units student has */

    /** Constant: the number of units required for graduation */
    public static final double UNITS_TO_GRADUATE = 180.0;

    // Creates a new student object with given name, ID, and 0 units.
    public Student(String studentName, int studentID) {
        name = studentName; ID = studentID; unitsEarned = 0;
    }

    // Returns the name of this student.
    public String getName() {
        return name;
    }

    // Returns the ID number of this student.
    public int getID() {
        return ID;
    }
```

```
    // Returns the number of units earned.
    public double getUnits() {
        return unitsEarned;
    }

    // Increments the earned units by the given number of units.
    public void incrementUnits(double additionalUnits) {
        unitsEarned += additionalUnits;
    }

    // Returns whether or not the student has enough units to graduate.
    public boolean hasEnoughUnits() {
        return unitsEarned >= UNITS_TO_GRADUATE;
    }

    // Creates a string IDing this student, such as "Marty (#223)".
    public String toString() {
        return name + " (#" + ID + ")";
    }
}
```

8. Paper Plane Airport

```
/*
 * File: Airport.java
 * ------------------
 * This program manages and dispatches Airplanes.
 */

import acm.program.*;
import java.util.*;

public class Airport extends ConsoleProgram {

   ArrayList<Airplane> planes;

   public void run() {
      planes = new ArrayList<Airplane>();

      // build 3 airplanes
      for (int i = 0; i < 3; i++) {
         println("Airport log: adding plane");
         Airplane plane = new Airplane();
         planes.add(plane);
      }

      // tell 2 to depart
      for (int i = 0; i < 2; i++) {
         dispatchPlane();
      }

      // build one more plane - can do this in 1 line below, or like
above
      println("Airport log: adding plane");
      planes.add(new Airplane());

      // tell all planes to depart
```

```
      while (!planes.isEmpty()) {
         dispatchPlane();
      }
   }

   private void dispatchPlane() {
      println("Airport log: dispatching plane");
      Airplane plane = planes.get(0);

      // just an example of error-checking using Airplane's "getter"
method
      if (plane.isAirborne()) {
         println("Airport log: ERROR - plane already airborne");
      }
      plane.takeOff();
      planes.remove(0);
   }
}
```

```
/*
 * File: Airplane.java
 * ---------------------------
 * This program implements the Airplane class used by the Paper Plane
 * Airport in Airport.java.
 */

public class Airplane {

      private boolean airborne;

      public Airplane() {
            foldInHalf();
            foldWings();
            this.airborne = false;
      }

      public boolean isAirborne() {
            return airborne;
      }

      public void takeOff() {
            System.out.println("Airplane log: dispatching plane");
            this.airborne = true;
      }

      private void foldInHalf() {
            System.out.println("Airplane log: folded plane in half!");
      }

      private void foldWings() {
            System.out.println("Airplane log: folded plane wings!");
      }
}
```

9. Subclassing GCanvas

```
/*
 * File: RandomCirclesCanvas.java
 * ------------------------
 * This GCanvas subclass adds the ability to also draw random circles.
 * Each circle has a randomly chosen color, a randomly chosen
 * radius between 5 and 50 pixels, and a randomly chosen
 * position on the canvas, subject to the condition that
 * the entire circle must fit inside the canvas without
 * extending past the edge.
 */

import acm.graphics.*;
import acm.util.*;

public class RandomCirclesCanvas extends GCanvas {
      private static final double MIN_RADIUS = 5;
      private static final double MAX_RADIUS = 50;

      public void drawRandomCircle() {
            double r = rgen.nextDouble(MIN_RADIUS, MAX_RADIUS);
            double x = rgen.nextDouble(0, getWidth() - 2 * r);
            double y = rgen.nextDouble(0, getHeight() - 2 * r);
            GOval circle = new GOval(x, y, 2 * r, 2 * r);
            circle.setFilled(true);
            circle.setColor(rgen.nextColor());
            add(circle); // adds it to ourself!
      }

      /* Private instance variable */
      private RandomGenerator rgen = RandomGenerator.getInstance();
}
```