

Solution to Section #5

Portions of this handout by Eric Roberts

1. Word count

```
/*
 * File: WordCount.java
 * -----
 * Counts the characters, words, and lines in a file.
 */

import acm.program.*;
import java.io.*;

public class WordCount extends ConsoleProgram {

    public void run() {
        int lines = 0;
        int words = 0;
        int chars = 0;
        BufferedReader rd = openFileReader("File: ");
        try {
            String line;
            do {
                line = rd.readLine();
                if (line != null) {
                    lines++;
                    words += countWords(line);
                    chars += line.length();
                }
            } while (line != null);
            rd.close();
        } catch (IOException ex) {
            println("An I/O exception has occurred");
        }
        println("Lines = " + lines);
        println("Words = " + words);
        println("Chars = " + chars);
    }

    /**
     * Asks the user for the name of an input file and returns a
     * BufferedReader attached to its contents.  If the file does
     * not exist, the user is given another chance to try.
     */
    private BufferedReader openFileReader(String prompt) {
        BufferedReader rd = null;
        while (rd == null) {
            String name = readLine(prompt);
            try {
                rd = new BufferedReader(new FileReader(name));
            } catch (IOException ex) {
                println("Can't open that file.");
            }
        }
        return rd;
    }
}
```

```

/**
 * Counts the words (consecutive strings of letters and/or digits)
 * in the input line.
 */
private int countWords(String line) {
    boolean inWord = false;
    int words = 0;
    for (int i = 0; i < line.length(); i++) {
        char ch = line.charAt(i);
        if (Character.isLetterOrDigit(ch)) {
            inWord = true;
        } else {
            if (inWord) words++;
            inWord = false;
        }
    }
    if (inWord) words++;
    return words;
}
}

```

2. How Unique!

```

/*
 * File: UniqueNames.java
 * -----
 * This program asks the user for a list of names (one per line)
 * until the user enters a blank line. Then the program prints
 * out the list of names entered, where each name is listed only
 * once (i.e., uniquely)
 */
public class UniqueNames extends ConsoleProgram {

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

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

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

```

3. A Christmas Carol

```
/**
 * File: Employee.java
 * -----
 * Class which describes the Employee variable type.
 */
public class Employee {

    public Employee(String newName, int newId) {
        name = newName;
        taxId = newId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public int getTaxId() {
        return taxId;
    }

    /* Employee instance variables */
    String name;
    int taxId;
    String title;
    boolean active;
    int salary;
}
```

```
/**
 * File ChristmasCarol.java
 * -----
 * Practices using the Employee class by creating variables of
 * type Employee.
 */
public class ChristmasCarol extends ConsoleProgram{
    public void run() {
        Employee ceo = new Employee("Ebenezer Scrooge", 161803399);
        Employee partner = new Employee("Jacob Marley", 271828182);
        Employee clerk = new Employee("Bob Cratchit", 314159265);

        ceo.setTitle("Ceo");
        partner.setTitle("Former Partner");
        clerk.setTitle("Clerk");

        ceo.setActive(true);
        partner.setActive(false);
        clerk.setActive(true);

        ceo.setSalary(1000);
        partner.setSalary(0);
        clerk.setSalary(25);
    }
}
}
```

4. Histogram

```
/*
 * File: Histogram.java
 * -----
 * This program reads a list of exam scores, with one score per line.
 * It then displays a histogram of those scores, divided into the
 * ranges 0-9, 10-19, 20-29, and so forth, up to the range containing
 * only the value 100.
 */

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

public class Histogram extends ConsoleProgram {

    public void run() {
        initHistogram();
        readScoresIntoHistogram();
        printHistogram();
    }

    /* Initializes the histogram array */
    private void initHistogram() {
        histogramArray = new int[11];
        for (int i = 0; i <= 10; i++) {
            histogramArray[i] = 0;
        }
    }
}
```

```

/* Reads the exam scores, updating the histogram */
private void readScoresIntoHistogram() {
    try {
        BufferedReader rd
            = new BufferedReader(new FileReader(DATA_FILE));
        String line;
        do {
            line = rd.readLine();
            if (line != null) {
                int score = Integer.parseInt(line);
                if (score < 0 || score > 100) {
                    throw new Exception("That score is out of range");
                } else {
                    int range = score / 10;
                    histogramArray[range]++;
                }
            }
        } while (line != null);
    } catch (IOException ex) {
        throw new Exception(ex);
    }
}

/* Displays the histogram */
private void printHistogram() {
    for (int range = 0; range <= 10; range++) {
        String label;
        switch (range) {
            case 0: label = "00-09"; break;
            case 10: label = " 100"; break;
            default:
                label = (10 * range) + "-" + (10 * range + 9);
                break;
        }
        String stars = createStars(histogramArray[range]);
        println(label + ": " + stars);
    }
}

/* Creates a string consisting of n stars */
private String createStars(int n) {
    String stars = "";
    for (int i = 0; i < n; i++) {
        stars += "*";
    }
    return stars;
}

/* Private instance variables */
private int[] histogramArray;

/* Name of the data file */
private static final String DATA_FILE = "MidtermScores.txt";
}

```

5. Tracing method execution

The output of `Halloween.java` is:

```
skellington 10
```