420-202-RE Assignment 1-B Due Date: 13 February 2017

Objectives

1. Give you continuous practice at writing and using classes.
2. Practice writing and using both normal and default constructors.
3. Give you practice with writing and the use of *copy* constructors.
4. Override the **toString** method in order to provide string representation for *invoking* objects.
5. Override the **equals** method to provide comparison between an *invoking* object and a *supplied* object.
6. Practice using arrays of references.
7. Practice reading from input text files.

Deep Copying Fruit Objects

Equip your **Fruit** class (from assignment 1-a) with a *copy* constructor.

```
public Fruit(Fruit otherFruit) { /* copy otherFruit to 'this' */ }
```

Modeling a Fruit List

Using your **Fruit** class from your assignment 1-a, develop a class **FruitList** to model a list of **Fruit** objects. Your **FruitList** class should store only two instance variables:

- An array of references to **Fruit** objects.
- The number of **Fruit** references actually stored in the array.

The database of fruit information is kept in a text file named **FruitDatabase.txt**. Each fruit record consists of five lines, as shown by the following sample fruit record:

```
Blueberries                          ← fruit name
One cup of blueberries               ← portion size
1.1                                  ← protein (grams)
84                                   ← calories
3.6                                  ← fiber (grams)
```

Thus the database file **FruitDatabase.txt** must contain multiples of 5 lines, with each five lines representing a fruit record.

The public interface of the class should includes the following methods.

1. A default constructor that creates a fruit list with default initial capacity of **10**.
2. A normal constructor that takes the initial array size as parameter and creates a fruit list of that size.
3. A **size** method that returns the number of elements in the fruit list.
4. A **capacity** method that returns the capacity of the fruit list.
5. An **addToEnd** method that takes a **Fruit** reference as parameter. It duplicates the **Fruit** object referenced and inserts its reference at the end of the list.
6. An **addToFront** method that takes a **Fruit** reference as parameter. It duplicates the **Fruit** object referenced and inserts its reference at the front of the list.
7. A **load** method that takes the name of a fruit database file. It reads the fruit records (if any) from the given file, creates a **Fruit** object for each record, and adds the references created to the end of the fruit list.

   Obviously, the input process must end when either all fruit records have been processed or the fruit list is full.

   If the given file does not exist, the method should display an error message to that effect, and then return, leaving the fruit list unchanged.

   If the given file does exist, the method may assume that the input file contains valid fruit records.
8. A **maxCalorie** method that returns the **Fruit** object in the list with maximum calories.
9. A **minCalorie** method that returns the **Fruit** object in the list with minimum calories.
10. A **maxProtein** method that returns the **Fruit** object in the list with maximum protein.
11. A **minProtein** method that returns the **Fruit** object in the list with minimum protein.
12. A **maxFiber** method that returns the **Fruit** object in the list with maximum fiber.
13. A **minFiber** method that returns the **Fruit** object in the list with minimum fiber.
14. A **totalProtein** method that returns the sum of protein values of the fruits in the list.
15. A **totalCalorie** method that returns the sum of calories of the fruits in the list.
16. A **totalFiber** method that returns the sum of fiber values of the fruits in the list.
17. A **toString** override that returns a string representation of the fruit list formatted as shown in the sample run on page 4.
18. An **equals** override that determines whether two **Fruit** objects are equal.

Test Driving Class FruitList

Test Constructors

Source code

```
// test normal constructor
        // create a fruit basket of capacity 15
        FruitList fruitBasket = new FruitList(15);

// Test our file reader and the 'addToEnd' method
        // load fruit basket with from input file
        fruitBasket.load("FruitDtabase.txt");
        // Note: FruitDtabase.txt has exactly 13 fruit records
        // So our fruit basket has 13 fruits on it
```

output

```
Sucessfully opened input file named FruitDtabase.txt
Fruit records processed: 13
Fruit list size: 13 Capacity: 15
```

Test toString(), and other Methods

Source code

```
10
11          System.out.println();
12
13  // Test toString(), totalCalories, totalProtein, totalFiber, size, capacity
14          // list fruits on fruit basket
15          System.out.println(fruitBasket);
16
```

output

```
4
5           Name  Protein  Calories  Fiber  Portion size
6           ====  =======  ========  =====  ============
7         Olives     0.07     10.00   0.30  One tablespoon of ripe olives
8           Kiwi     0.79     42.00   2.10  One medium kiwi (69 grams)
9        Papayas     0.85     55.00   2.50  One cup of cubed fresh papaya
10        Banana     1.29    105.00   3.10  One medium banana
11       Avocado     4.02    322.00  13.50  One medium avocado
12  Gooseberries     1.32     66.00   6.50  One cup of gooseberries
13    Watermelon     1.74     86.00   1.10  One medium wedge (slice) of watermelon
14   Pomegranate     4.71    234.00  11.30  One fresh pomegranate
15         Peach     1.36     58.00   2.20  One medium peach (with skin)
16    Grapefruit     1.45     74.00   2.50  One cup of grapefruit sections
17         Pears     0.68    103.00   5.50  One medium pear
18         Lemon     0.92     24.00   2.40  One lemon without peel
19       Raisins     1.32    129.00   1.60  One small box of raisins (1.5 ozs)
20          ====  =======  ========  =====
21         Total    20.52   1308.00  54.60
22  Size     : 13
23  Capacity: 15
24
```

Test Method 'insertAtFront'

Source code

```
// Test the 'insertAtFront' method
    fruitBasket.insertAtFront(
        new Fruit("Blueberries", "One cup of blueberries", 1.1, 84, 3.6));
    fruitBasket.insertAtFront(
        new Fruit("Lychees", "One cup of fresh lychees", 1.58, 125, 2.5));

     System.out.println(fruitBasket); // list fruits on fruit basket

```

output

```
        Name  Protein  Calories  Fiber  Portion size
        ====  =======  ========  =====  ============
     Lychees     1.58    125.00   2.50  One cup of fresh lychees
 Blueberries     1.10     84.00   3.60  One cup of blueberries
      Olives     0.07     10.00   0.30  One tablespoon of ripe olives
        Kiwi     0.79     42.00   2.10  One medium kiwi (69 grams)
     Papayas     0.85     55.00   2.50  One cup of cubed fresh papaya
      Banana     1.29    105.00   3.10  One medium banana
     Avocado     4.02    322.00  13.50  One medium avocado
Gooseberries     1.32     66.00   6.50  One cup of gooseberries
  Watermelon     1.74     86.00   1.10  One medium wedge (slice) of watermelon
 Pomegranate     4.71    234.00  11.30  One fresh pomegranate
       Peach     1.36     58.00   2.20  One medium peach (with skin)
  Grapefruit     1.45     74.00   2.50  One cup of grapefruit sections
       Pears     0.68    103.00   5.50  One medium pear
       Lemon     0.92     24.00   2.40  One lemon without peel
     Raisins     1.32    129.00   1.60  One small box of raisins (1.5 ozs)
        ====  =======  ========  =====
       Total    23.20   1517.00  60.70
Size     : 15
Capacity: 15
```

Test Method ‘insertAtFront’ on Full List

Source code

```
25      // Note: our fruit basket is now full
26  // Test the 'insertAtFront' method on full fruit basket
27      fruitBasket.insertAtFront(new Fruit("Lime", "One lime", 0.47, 20, 1.9));
28
```

output

```
46
47  Error - cannot add to a full fruitlist.
48
```

Test the ‘minimum’ and ‘maximum’ Methods

Source code

```
29        System.out.println("\nFruit with minimum Protein: \n"
30                + "--------------------------- \n"
31                + fruitBasket.minProtein());
32
33        System.out.println("\nFruit with maximum Protein: \n"
34                + "--------------------------- \n"
35                + fruitBasket.maxProtein());
36
37        System.out.println("\nFruit with minimum Calorie: \n"
38                + "--------------------------- \n"
39                + fruitBasket.minCalories());
40
41        System.out.println("\nFruit with maximum Calorie: \n"
42                + "--------------------------- \n"
43                + fruitBasket.maxCalories());
44
45        System.out.println("\nFruit with minimum Fiber: \n"
46                + "-------------------------\n"
47                + fruitBasket.minFiber());
48
49        System.out.println("\nFruit with maximum Fiber: \n"
50                + "------------------------- \n"
51                + fruitBasket.maxFiber());
```

output

```
Fruit with minimum Protein:
---------------------------
Fruit name   :Olives
Portion size: One tablespoon of ripe olives
Protein      : 0.07 grams
Calories     : 10
Fiber        : 0.3 grams

Fruit with maximum Protein:
---------------------------
Fruit name   :Pomegranate
Portion size: One fresh pomegranate
Protein      : 4.71 grams
Calories     : 234
Fiber        : 11.3 grams

Fruit with minimum Calorie:
---------------------------
Fruit name   :Olives
Portion size: One tablespoon of ripe olives
Protein      : 0.07 grams
Calories     : 10
Fiber        : 0.3 grams

Fruit with maximum Calorie:
---------------------------
Fruit name   :Avocado
Portion size: One medium avocado
Protein      : 4.02 grams
Calories     : 322
Fiber        : 13.5 grams

Fruit with minimum Fiber:
-------------------------
Fruit name   :Olives
Portion size: One tablespoon of ripe olives
Protein      : 0.07 grams
Calories     : 10
Fiber        : 0.3 grams

Fruit with maximum Fiber:
-------------------------
Fruit name   :Avocado
Portion size: One medium avocado
Protein      : 4.02 grams
Calories     : 322
Fiber        : 13.5 grams
```

Test Default Copy Constructor

Source code

```
53
54          System.out.println();
55  // Test the default constructor and the equals override
56          FruitList fruitBowl = new FruitList();
57          fruitBowl.load("FruitDtabase.txt");
58          System.out.println();
59          System.out.println(fruitBowl);
60          System.out.println();
61          if(fruitBowl.equals(fruitBasket))
62              System.out.println("fruit bowl and fruit basket are equal");
63          else
64              System.out.println("fruit bowl and fruit basket are NOT equal");
```

output

```
96
97  Sucessfully opened input file named FruitDtabase.txt
98  Fruit records processed: 10
99  Fruit list size: 10 Capacity: 10
100
101           Name   Protein   Calories   Fiber   Portion size
102           ====   =======   ========   =====   ============
103         Olives      0.07      10.00    0.30   One tablespoon of ripe olives
104           Kiwi      0.79      42.00    2.10   One medium kiwi (69 grams)
105        Papayas      0.85      55.00    2.50   One cup of cubed fresh papaya
106         Banana      1.29     105.00    3.10   One medium banana
107        Avocado      4.02     322.00   13.50   One medium avocado
108  Gooseberries      1.32      66.00    6.50   One cup of gooseberries
109    Watermelon      1.74      86.00    1.10   One medium wedge (slice) of watermelon
110   Pomegranate      4.71     234.00   11.30   One fresh pomegranate
111          Peach      1.36      58.00    2.20   One medium peach (with skin)
112    Grapefruit      1.45      74.00    2.50   One cup of grapefruit sections
113           ====   =======   ========   =====
114          Total     17.60    1052.00   45.10
115  Size     : 10
116  Capacity: 10
117
118
119  fruit bowl and fruit basket are NOT equal
```

Evaluation Criteria

Correctness of execution of your program	60%
Proper use of required Java concepts	20%
Java API documentation style	10%
Comments on nontrivial steps in code, Choice of meaningful variable names, Indentation and readability of program	10%