# Lab 6: Managing a List

**Steven K. Andrianoff**
**Computer Science Department**
**St. Bonaventure University**
**Copyright, 2011**

**Objective:**

This lab introduces the ArrayList class. It also introduces the *compareTo* method and *Comparable interface*.

**Instructions:**

1. Begin by creating a new Java project named Lab6 in Eclipse. Create a main class named IntLister within the project. Add the file `Dice.java` to the project. Within the main method write code to generate 20 random integer values in the range 1 to 100 and place them in an array of ints. (Create a 100-sided Dice to generate the numbers.) Display the list of integers.

   Run the program to be sure your code is correct.

2. Now replace the array of ints with an ArrayList. Here is the code for creating the ArrayList.

   ```
   ArrayList<Integer> list = new ArrayList<Integer>();
   ```

   *list* is an ArrayList that stores Integer's.

   There are three ArrayList methods that we frequently use:

   `size()` - returns the current size of the list
   `get(i)` - returns the element at position i of the list (as with arrays the first position is 0)
   `add(item)` - adds item to the end of the list (and increases the size of the list by 1)

   Rewrite your main method to store the integer values you generate into the ArrayList you created. Rewrite the code you have for displaying the list to use the ArrayList.

   Run the program again to be sure your code is correct. Hand in the output with your lab write up.

3. Now change your code to store a list of words in the list rather than a list of integer values. Obtain your words from a text file.

   Create a new program in the Lab6 project named WordLister. In the main method create an ArrayList of Strings and then fill the list with words read from an external text file. Here is sample code (`WordReader.java`) to read words from a text file. (Note: this code does not do anything with the words read.) Use this code in your program to read the words from the file and store them in the ArrayList. Add code to display the resulting list after it has been built.

   Run the program using the text file `sample.txt` and hand in the output with your lab write up.

   Since the list of words is an object of type ArrayList<String>, what is the result of the statement

   ```
   System.out.println(list);
   ```

   Add this statement to your program, run the program, and hand in the output with your lab write up.

4. Next we want to create a list of BankAccounts.

   Create a new program in the Lab6 project named AccountLister. First, add your BankAccount class from Lab 5 to your Lab6 project. In the main method of AccountLister create an ArrayList of BankAccounts and then fill the list with BankAccounts created from data read from an external text file. Here (`AccountReader.java`) is code that you can use to help you read data for a list of BankAccounts from an external data file. Note that my code assumes the balance is a double not an int and take careful note of the order of the arguments in my BankAccount constructor - yours may be different. Add code to display the resulting list after it has been built.

   Create a data file with data for at least 11 bank accounts.

   Run the program with this data file and hand in the output with your lab write up.

5. At this point of the lab we will change gears and look at how to compare objects. We have seen how one can use the equals method to compare two objects for equality. We have also seen how to write an equals method for a class we construct. Now we examine how to compare two objects to determine which one is larger (or smaller).

   Start by creating a new main class called ThingComparer that defines two integer variables. Store a random number from the range 1-100 in each variable. Then use an if-else if statement that displays the value that is larger or displays the statement that they have the same value – this should be a 3-way selection. Run this program several times and observe its behavior.

   Now what happens if you replace the int variables with two String variables and store "one" in the first variable and "two" in the second? Make these changes, run the program, and record your results.

6. Open up the API for the String class. Examine the list of String methods and determine which method should be used to see which of two Strings is larger.
   What is the signature of this method?
   What are the possible return values?
   How is the result determined?

   Modify your main method to print the result of using this method to compare the following pairs of strings:
   "one" with "two"
   "two" with "one"
   "this" with "that"
   "that" with "this"
   "this" with "this"
   "the" with "then"

   What does it mean for one String to be "smaller" than another String?

7. The compareTo method can be used to determine which of two Strings "comes first". For example, to ask if *str1* "comes before" *str2* one would write the statement:

```
if (str1.compareTo(str2) < 0) . . .
```

   Modify your main method to ask the user to enter two different words, storing each word in a String variable. Use an if-else if statement that displays the word which "comes first" or displays the statement that they have the same value.

   Run the program for each of the pairs of words from Step 6 and record your results. Hand in your results with your lab write up.

8. Modify your main program to create two BankAccount objects. Compare the two BankAccounts using compareTo. What problem do you encounter? Why?

   Add the compareTo method to the BankAccount class. Here (`compareTo.txt`) is an example of how the

compareTo method would be written for the Person class where the comparison is based on age. That is, Person *p1* is smaller than Person *p2* if *p1*'s age is smaller than *p2*'s age. In the case of BankAccount compare the BankAccounts based on their balance.

Now modify your code to create two BankAccount objects and compare them using the compareTo you have written. As before use an if-else if statement that displays the BankAccount which is smaller or displays the statement that they are the same.

Run the program with three sets of data to illustrate each of the three possible outcomes. In your lab write up describe the data you used with the results.

At this point print your program `ThingComparer.java` and hand it in with your lab.

9. Return to your program `IntLister.java` from Step 2. At the end of the program add the following code snippet:

```
Object[] oList = list.toArray();
Arrays.sort(oList);
list.clear();
for (int i = 0; i < oList.length; i++)
{
    list.add((Integer) oList[i]);
}
```

Following this code display your list again. What do you notice about the result?

Run the program and record your results. Hand in your results with your lab write up.

Print out `IntLister.java` and hand it in with your lab write up.

Now modify your program WordLister from Step 3 in the same way. (You will need to make a couple of minor adjustments to get it to work correctly.) What do you notice about the result?

Run the program and record your results. Hand in your results with your lab write up.

Print out `WordLister.java` and hand it in with your lab write up.

10. Return to your program AccountLister from Step 4 and make the same modification. Record the error that you get.

This is easily fixed. Add the following line to the end of the class definition of BankAccount:

```
implements Comparable<BankAccount>
```

Run the program and record your results. Hand in your results with your lab write up.

Print out `AccountLister.java` and hand it in with your lab write up.

**Hand in:**

The write-up you hand in for this lab should include:

- the answers to the questions in Steps 6, 8, and 9.
- printed copies of the output from Steps 2, 3, 4, and 9.
- discussion of results from Steps 5, 7, 8, 9, and 10.
- the printed copy of the source code requested in Steps 8, 9, and 10.

**Help Policy:**

Help Policy in Effect for This Assignment: Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 132.

You may use the following materials produced by other students: materials produced by members of your group.

You may use the following materials produced by other students: NONE.