Project 14

CS 1323, Fall 2015

# Learning Objectives

1. Insert elements into an ArrayList, preventing duplicates from being inserted using the Collections. binarySearch() method. (20 points)
2. Perform linear search for partial matches in an ArrayList using the startsWith() method in the String class. (20 points)
3. Create a nested loop to perform auto-completion. (20 points)
4. Use at least four methods, with well-chosen parameters and return types. (20 points)
5. Create a menu driven interface. (10 points)

10 points will be awarded for the documentation of your program. That means using good names for variables, proper and consistent indentation of code, and meaningful use of whitespace.

**This project is essentially the same as Project 10, only using ArrayList objects instead of arrays. You are welcome to use your or *my* Project 10 code as a starting point for this project. *The grade from this project will replace your lowest laboratory grade.* I recommend that you do this project BEFORE the midterm examination to help solidify your understanding of ArrayList objects.**

Section 10: Have the TAs check your assignment by the end of the laboratory or submit it on Janux by 11:59 p.m. on Monday, December 2. ***For this week only, you may work on this project on your own without coming to the laboratory.***

Sections 1 and 995: Submit your program on Janux by 11:59 p.m. on Monday, December 2.

# Description

You have probably noticed that many pieces of software use auto-completion to accelerate data entry. Examples of programs that typically use auto-completion include eclipse, text messaging, looking up contacts in an address book, and email addresses in Outlook. In this project we will create a program that auto-completes email addresses.

An interface for the program is shown below (user input is in bold)

1. Enter a new email address

2. Find an existing email address

3. Quit

What is your choice?

**1**

Enter the email address:

[**dtrytten@ou.edu**](mailto:dtrytten@ou.edu)

Insertion successful

// Repeat menu

What is your choice?

**1**

Enter the email address:

[**dtrytten@ou.edu**](mailto:dtrytten@ou.edu)

That email address is already inserted

// Repeat menu

What is your choice?

**1**

Enter the email address:

[**dt2@ou.edu**](mailto:dt2@ou.edu)

Insertion successful

//Repeat menu

What is your choice?

**2**

Enter the first letters, one at a time

**d**

[dtrytten@ou.edu](mailto:dtrytten@ou.edu)

[dt2@ou.edu](mailto:dt2@ou.edu)

**t**

[dtrytten@ou.edu](mailto:dtrytten@ou.edu)

[dt2@ou.edu](mailto:dt2@ou.edu)

**2**

Found [dt2@ou.edu](mailto:dt2@ou.edu)

// Repeat menu

What is your choice?

3

Bye.

This program uses an ArrayList object.

## Inserting Data

When you go to insert data in this array, you need to be sure that it isn’t already there. This is best done using Collections.binarySearch(). Remember that you must have the data in sorted order for binary search to work. This can be done with a method like Collections.sort(), although this is somewhat inefficient. It also can be done by performing insertion using the same algorithm we used in Laboratory 10.

## Class Constants

Creating a menu driven interface is relatively simple, except that the parameter passing gets ugly without one programming trick. If you declare constants like ADD, QUIT, and SEARCH in the main program, you have to bring them into other methods as parameters. An easy work around for this is to create class data. This is done by putting the parameters in the class body, but not inside of any method. Make them static. When they are declared this way, they may be used in any method inside of the class.

**public** **class** AutoCompletion

{

**public static** **final** **int** *ADD* = 1;

**public** **static** **final** **int** *SEARCH* = 2;

**public** **static** **final** **int** *QUIT* = 3;

}