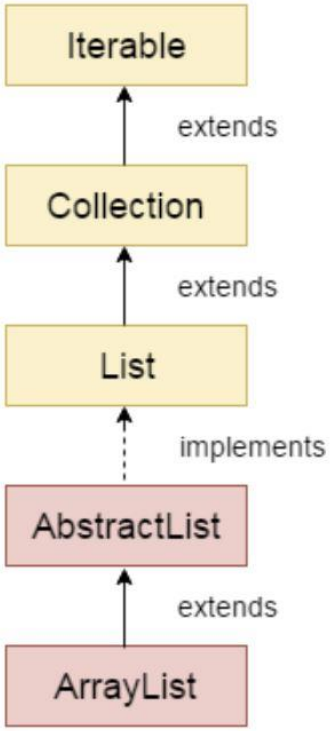
Bluegrass Community and Technical College

Programming Requirements Document

Java Data Structures – List Interface and ArrayList Class

NARRATIVE DESCRIPTION

This week we start our study of common data structures used in computer science. We will begin our study of Java collections and data structures looking at the **List** interface and the **ArrayList** implementation of **List**.



One very important thing to remember about the ArrayList is it holds **objects** (Strings, objects instantiated from user-defined classes, etc.) and not primitive data items (byte, int, short, long, char, boolean, float, double).

For this assignment, you may store String objects in the ArrayList or you may create a Volunteer class and store Volunteer objects in the ArrayList. The choice is entirely up to you.

A local community college is hosting an Hour of Code (code.org) and is seeking volunteers to help.

The program you create this week will help maintain a list of the volunteers. A small CSV file has been created of the current volunteers who have agreed to help (Volunteers.csv). The file contains 3 fields of data:

- Volunteer's last name
- Volunteer's first name
- Volunteer's email address

Step 1: Read the Data in Volunteers.csv and store in an ArrayList

Read the data in Volunteers.csv and store the elements in an ArrayList object.

Below is a sampling of the records in the CSV file

```
Adams,McKenzie,adamsm8800@kctcs.edu
Baldwin,Anya,baldwina7896@kctcs.edu
Black,Colin,blackc8111@kctcs.edu
```

You may store a record in the ArrayList as one of the two options below:

1. **String objects** with comma-delimited fields. For example, the first String stored in the ArrayList would be:

```
Adams,McKenzie,adamsm8800@kctcs.edu
```

2. **Volunteer objects.** You may choose to create a Volunteer class:

Volunteer
- lastName : String - firstName : String - email : String
+ Volunteer ( ): + Volunteer (String lastName, String firstName, String email):
+ getLastName( ) : String + getFirstname( ) : String + getEmail( ) : String + setLastName(String lastName) : void + setFirstname(String firstName) : void + setEmail(String email) : void + toString( ) : String

Using this technique, you would instantiate Volunteer objects and place them in the ArrayList.

BONUS POINTS: You can earn 2 extra bonus points on the assignment if you create the Volunteer class, use Volunteer objects for processing, and store Volunteer objects in the array.

Step 2: Create a GUI to Maintain the ArrayList

Create a Java GUI application to maintain the data in the ArrayList. The user should be able to:

- **Display the current volunteers in the ArrayList –** List the index for a record and the data for the record. The index will help the user if they choose to delete a record later. The data should be displayed in an easy to read manner (not the comma-delimited format if using Strings in the ArrayList).

- **Add a volunteer to the list** – the user will enter data for a new volunteer and the data will added to the ArrayList. If you are using Strings, format the data with the commas so that the format is formatted exactly as current data in the ArrayList. If you are using Volunteer objects, instantiate a new object and add it to the ArrayList. The CSV file is currently sorted alphabetically. It would be best to maintain a sorted list in the ArrayList. However, since this is your first application using an ArrayList, you may choose to add the new volunteer at the end of the list or in a sorted position. Whatever data is currently displayed in the GUI should be current as actions are taken.

- **Remove a member from the list** – Allow the user to enter an index for the ArrayList. It should display the volunteer's information on the screen and ask the user if this is the correct record to delete. If the user confirms this is correct, remove the entry from the ArrayList. The user may also choose to cancel the delete. Whatever data is currently displayed in the GUI should be current as actions are taken.

- Exit the GUI – If changes have been made to the ArrayList, ask the user if they want to save the changes. If they do, create a new CSV file named "Volunteers2.csv". In a real scenario, you would replace the original file but this allow you to test your program without corrupting the original file.

Zip together all Java files needed to test your solution plus the Volunteer.csv file.

New and Reviewed Concepts Assessed and Illustrated

- File class
- List interface
- ArrayList class
- Reading from text files

Software Requirements

Standard Requirements from previous weeks are required even though not explicitly listed (documentation, programming standards, clean code, etc.)

R1: The ArrayList is populated with the current data from Volunteers.csv (Strings or Volunteer objects).
R2: A GUI is used for the applications.
R3: Elements in the ArrayList are displayed properly in the GUI in a professional format. Record are up to date in the display as actions are taken (add, delete, etc.)
R4: Elements can be added to the ArrayList (at the end or in sorted position).
R5: Elements can be deleted from the ArrayList using the index (position).
R6: The user of the application can choose whether or not to save changes in a file named Volunteers2.csv. The file is created and formatted properly.

Security Considerations

Incorporate security in all classes created as learned in class.

Special Notes

None.

Change Request Form

Students who wish to obtain written permission to alter the assignment or to use features/statements/structures before they are introduced in class, must complete a *Change Request Form* (link in Blackboard in left-hand navigation bar) and follow all guidelines provided there. Please follow these procedures.