CSC 120: Play a Tune

Loops and Arrays

1 Introduction

This lab is designed to reinforce two of the topics covered in CSC120: arrays and loops. You will add your own code that effectively implements both of these topics in certain sections in the provided source code. The code you create will go in a source file called `PlayATune.java` and will be used by another source file called `PlayATuneHelper.java`. Note that you will not have to edit this second file.

2 Objective

The objective of this lab is to teach the student how to detect and correct errors that occur during the execution of a program. These errors will be confined primarily to the syntactical and logical varieties.

3 Activity

This section will guide you to successful completion of the lab. Two subsections fulfill this purpose, the first of which provides background information and the second of which lists sequential steps to follow.

3.1 Background Information

This section provides information you may require to finish the lab successfully. Refer to it as needed while you follow the steps outlined below.

3.1.1 Arrays

An array is an elementary data structure that stores homogeneous information in discrete locations called buckets or slots. Each bucket holds exactly one of the type the array was instantiated to hold. In Java, arrays may be explicitly declared, or you may use an `ArrayList`. In this project you will use both.

3.1.2 Loops

Loops are constructs that provide repetition until some specific condition occurs. Such a condition is called an exit condition. There are two main kinds of loops in Java: a `FOR` loop and a `WHILE` loop. `WHILE` loops take a boolean value as a parameter and execute the body of the loop until the boolean is false. This boolean value is called a sentinel value; so long as it is `true`, the loop will execute. Moreover, this parameter may be either an explicit boolean constant (i.e., `true` or `false`), a boolean variable reference, or a function call that returns a boolean type. `FOR` loops behave a little differently. They take three parameters, each separated by a semicolon (e.g., `for (i=0; i<10; i++)`). The first parameter is executed before the first iteration of the body of the loop. The second parameter is the exit condition and is checked before each iteration of the body of the loop. The final parameter is executed after completing each iteration of the body of the loop.

3.2 Implementation

This lab will allow you to create a simple Android application that will play a tune. You will be provided a Java source file named `PlayATune.java`. This file contains the skeleton framework that will be used by `PlayATuneHelper.java` in order to retrieve proper frequencies for musical notes.

To complete this lab successfully you will need to:

1. Create an array of type String named `notes`, instantiated to contain the letters C, D, E, F, G, A, and B.
2. Create an `ArrayList` variable of type `Integer` named `freq`.
3. Properly initialize the `ArrayList`.
4. Modify the `fillFreq()` method to copy the contents of `a` into `freq` using a `FOR` loop.
5. Using a `FOR` loop, modify the `getFreq()` method to search the `notes` array for the parameter `s` using the index of `s` within `notes` to return the value contained in `freq`. The method should return `-1` upon failure.
6. Using a `WHILE` loop, modify the `clearAll()` method so that it will remove all elements from `freq`.
7. Modify `getSize()` so that it returns the current size of `freq`

Once all tasks have been completed, compile your project and test it on the emulator and Android developer device. If successfully completed, you will be rewarded with a familiar tune.

4 Conclusion

After successful completion of this lab, the student will be able to initialize and utilize both explicit arrays and Java's `ArrayList` utility class. In addition, the student will understand how to store, retrieve, and remove elements from array buckets or slots. The student will also understand and be able to use both `FOR` and `WHILE` loops and understand the concept of iteration.

5 Deliverables

To submit your application, export your Eclipse project as a file system, zip all of the files into an archive and submit them online with the filename `<first_name_initial><last_name>-lab<lab#>.zip`.