WESTERN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

SE2205b - Algorithms and Data Structures for Object-Oriented Design

Assignment 2
Due Date: March 30 , 2018

This is a teamwork assignment. You can work in groups of up to 2 students. If you are working within a team, please make sure that the UWO ID of **both** team members are in the final submission file name.

1 Goals

At the end of this assignment, you should be able to:

- Write Java code for a simple database manager that will store a set of bird species in an ordered dictionary.
- Understand how to implement dictionary ADT using a binary search tree.
- Use JavaFX and Scene Builder to build graphical user interface GUI application.

2 Description and Requirements

The most common objective of software applications is to store and retrieve data, in which the efficiency of organizing a collection of this data is a paramount. In this assignment, you will work on a simple interface for such a collection, called a dictionary, your dictionary will house a collection of bird species data records. You will consider the dictionary ADT that provides operations for storing records, finding records, and removing records from the collection.

Dictionaries depend on the concepts of a search key and comparable objects. To implement the dictionary's search function, we will require that keys be totally ordered. Ordering fields that are naturally multi-dimensional, such as a point in two or three dimensions, present special opportunities if we wish to take advantage of their multidimensional nature. Each record in your collection will be stored in the form of (DataKey, about, sound, image), where 'about' is string holding the information we collect about the bird, 'sound' is the audio filename of the bird's sound, 'image' is the filename of the bird's image, and 'DataKey' is a pair (birdName, birdSize), where 'birdName' is a string representing a unique identification of the bird, and 'birdSize' is an integer value as follows:

- birdSize = 1, if bird's size is small.
- birdSize = 2, if bird's size is medium.
- birdSize = 3, if bird's size is large.

When comparing two DataKey k1 = (birdName1, birdSize1) and k2 = (birdName2, birdSize2), we use the following rules:

- k1 = k2, if birdSize1 = birdSize2 and birdName1= birdName2

- k1 < k2, if birdSize1 < birdSize2 or
  if birdSize1 = birdSize2, and birdName1 precedes birdName1.

Consider, for example, a database storing the following birds' records:

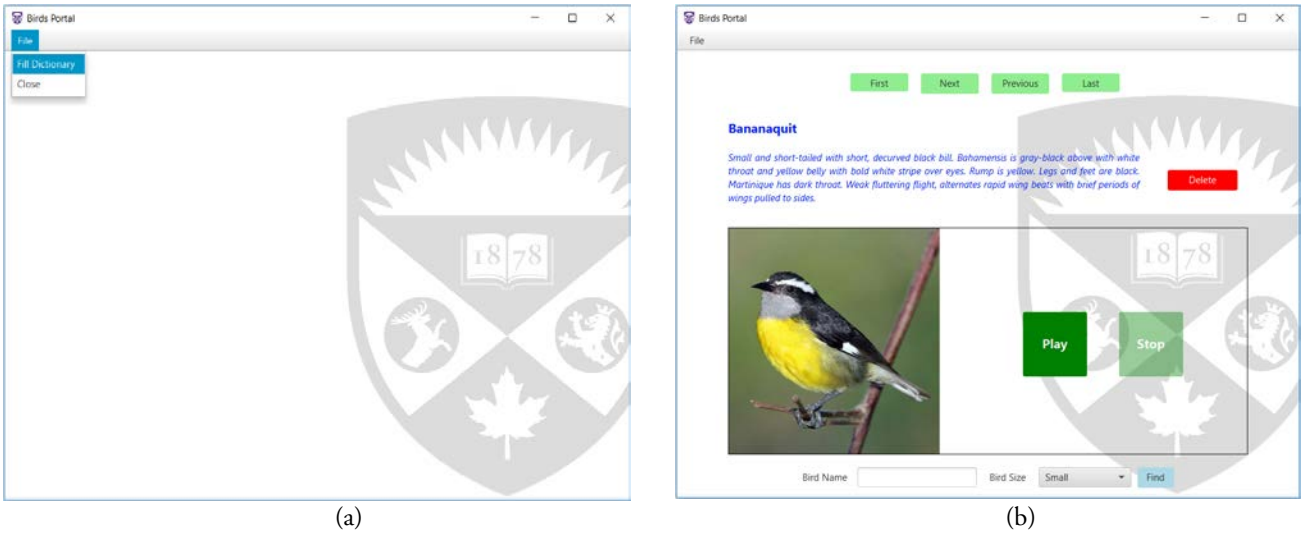
- Record1 = (("Rock Pigeon", 2), "Common throughout the continental United States, southern Canada, Mexico, and urban areas throughout the world.", "Rock Pigeon.mp3", "Rock Pigeon.jpg")
- Record2 = (("Ringed Kingfisher", 2), "Commonly found along the lower Rio Grande valley in south-eastern most Texas in the United States through Central America to Tierra del Fuego in South America.", "Ringed Kingfisher.mp3", "Ringed Kingfisher.jpg")
- Record3 = (("Northern Cardinal", 3), "This large crested finch has a vivid red body. The black mask and chin contrast with a heavy red bill.", "Northern Cardinal.mp3", "Northern Cardinal.jpg")

In the above dataset, we have Recod1 and Record2 are smaller than Record3 because their birdSize are smaller than the birdSize of Record3. And Record2 is smaller than Record1 because the birdName of Record2 alphabetically comes before the birdName of Record1, note that, both records have the same birdSize.

Your application will provide a GUI-based interface that will allow users to interact with the database through number of menu items, text fields and buttons. JavaFX and Scene Builder will be your programming language and the tools to build your application. Please use NetBeans IDE to develop your application. Figure 1, shows a sample run of the required application, the following describe how would your application behave.

2.1 The Main Menu

The application starts showing its initial state, where a blank screen with UWO logo in the background will be displayed in addition to one main menu at the top. The main menu of the application will include a 'File' menu that has two menu items 'Fill Dictionary' and 'Close'. When a user chooses File → Fill Dictionary option, the application will read the birds information from the given text file and populate the dictionary (the bird database) using the Insert method. When the dictionary is filled the first bird in the database will be displayed.



(a) (b)

Figure 1: (a) shows the initial stage of the application. (b) shows the application screen when the database is filled.

2.2 Search Bird

The application should provide the feature for searching the database for a specific bird by its name and size. The user need to enter the bird full name in Bird Name Textfield and choose the bird's size from the ComboBox and then click Find. The application will display the name, title, and the image of the bird if it exists or show an appropriate error message on the screen.

2.3 Play/Stop Sound

The application should be able to play the sound of the correct bird displayed on the screen. When the user clicks Play button the sound begins playing, until the audio file ends or the user click Stop button. This application supports MP3 audio files only.

2.4 Delete Bird

When the user clicks the 'Delete' button, the application should remove the current displayed bird from the database using the dictionary remove() method. After the record is deleted the available next or previous record will be displayed, if the deleted record was the last one in the database, the application will be reset to its initial state. That is a blank screen with UWO logo in the background in addition to the main menu at the top, the user can close the application or re-start again.

2.5 Next Bird

When the user clicks the 'Next' button, the application should display the bird information in the database that follows the current bird using the dictionary method successor (). If such a record does not exist because there is no record in the database with key larger than the current one then an appropriate message is displayed.

2.6 Previous Bird

When the user clicks the 'Previous' button, the application should display the bird information in the database that precedes the current bird using the dictionary method predecessor (). If such a record does not exist because there is no record in the database with key smaller than the current one then an appropriate message is displayed.

2.7 First Bird

When the user clicks the 'First' button, the application should display the bird information with smallest key in the database using the dictionary method smallest ().

2.8 Last Bird

When the user clicks the 'Last' button, the application should display the bird information with largest key in the database using the dictionary method largest ().

2.9 Close application

The application should exit, when the user chooses File → Close.

3 Objects Implementations

To make your life easier, a start-up application kit is given. It includes a template GUI interface, birds' images, audio files, the Java exceptions classes, the OrderDictionaryADT interface, and the units testing code. If you did not do it already, download the file Birds.zip and import it into your NetBeans IDE. You are required to implement the following Java classes: DataKey, BirdRecord, OrderedDictionary, and to modify the BirdController along with its Birds.fxml. Feel free to implement more classes if you need to, for example you may need to implement a Node class for your binary tree in addition to the form that will display the error messages.

3.1 DataKey

Each record in your database will be stored in the form of (DataKey, about, sound, image), and the DataKey is a pair (birdName, birdSize) as described above.

For this class you must implement the following public methods, however, you can implement any other methods that you want to in this class, but they must be declared as private methods.

1. The constructor(s) of the class.
2. public String getBirdName(), the method that returns the birdName of the DataKey.
3. public int getBirdSize(), the method that returns the type of the DataKey.
4. public int compareTo(DataKey k), the method that returns 0 if this DataKey is equal to k, returns -1 if this DataKey is smaller than k, and it returns 1 otherwise.

3.2 BirdRecord

This class represents a bird record in the database. Each record consists of the followings: a 'key' of type DataKey, 'about' of type String to hold the bird description, 'sound' of type String to hold the audio file name, 'image' of type string to hold the bird image name. For this class, you must implement the following public methods, however, you can implement any other methods that you want to in this class, but they must be declared as private methods.

1. The constructor(s) of the class.
2. public DataKey getDataKey(), the method that returns the key of the record.
3. public String getAbout(), the method that returns the bird description in the record.
4. public String getSound(), the method that returns the bird audio file name.
5. public String getImage(), the method that returns the bird image file name.

3.3 OrderedDictionary

You will build your database using a dictionary data structure. In this application, a binary tree will be used to implement your dictionary and not need to be balanced tree. The OrderedDictionary class should be a direct implementation to the given OrderedDictionaryADT interface. You must use a BirdRecord object to store the data contained in each node of the tree. In your binary search tree, only the internal nodes will store information. The leaves will be java objects storing null objects. The dictionary key for an internal node will be the DataKey object from the BirdRecord stored in that node. Again, you might want to implement a class Node to represent the nodes of the binary search tree. Each node will store a BirdRecord, and references to its left child, right child, and parent. When comparing two node keys you need to use method

compareTo() from class DataKey. Keep in mid that you must implement all the public methods declared in the given OrderedDictionaryADT interface, and to use the given dictionary exception class to handle the exceptional cases as it is mentioned in top of each method. Note that, you need to catch these exceptions in the BirdsController class and display their messages on the screen. You will need create another fxml file.

By implementing all required methods, test your application using the provided unit testing code. DO NOT CHANGE ANY LINE IN THE **ordereddictionarytest.java**. In the NetBean IDE, choose Run → Test Project. Fix the errors (if any) until all tests pass, as shown in Figure 2. You need to make sure that your dictionary operations are implemented correctly before you start building your application interface.

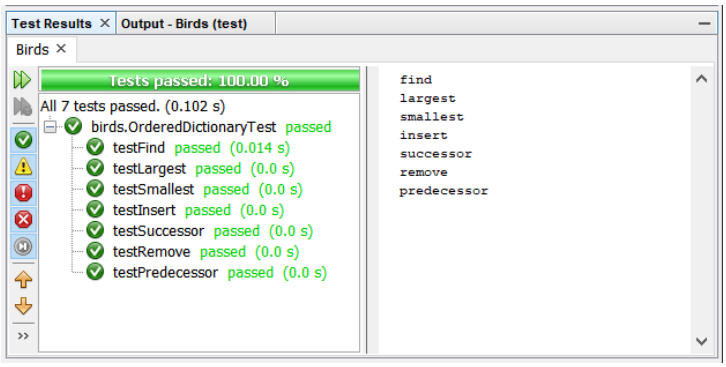


Figure 2

3.4 BirdsController

BirdsController is the class that interacts with the Birds.fxml file to show the main application interface and to control the behaviour of the entire application. In this class you need to implement at least the following public methods: first(), last(), next(), previous(), delete(), find(), play(), and stop() to handle the application buttons clicks. In addition to the method loadDictionary() to be called when the user choose File → Fill Dictionary. This method should be able to read the birds information from the given text file that has the following format. Each bird record has three lines of data, the first line contains a number that represent the bird size, the second line contains a string representing the bird name, the third line contains another string representing the bird's description. For example, the first two data records in the input file might be as follows:

Line1:	1
Line2:	Lucifer Hummingbird
Line3:	Small hummingbird, metallic green upperparts, head, sides, flared purple-red gorget (throat feathers) extending around sides of neck, white breast and belly. Tail is dark and deeply forked. Bill is long and decurved. Direct and hovering flight on rapid wing beats.
Line4:	3
Line5:	Gilded Flicker
Line6:	Large woodpecker with dark barred and spotted brown back, brown cap, pale gray face and throat, red moustache stripe, white rump, thick black crescent on upper breast, and black spotted, pale buff underparts. It was named for the gold color of its underwings and tail.

In this example, the first line has a number 1 to indicate that the nest two lines have a data about a bird of size 1 (Small), line2 has the name of the bird, and line3 has a description of the bird. Line4 is a number for the size of the nest bird record, followed by the bird name and description, and so on.

3.5 Birds.fxml

You are provided by an fxml file that contains basic elements for the application UI, and you need to add on your other UI element to have at the end your UI something similar to the one in Figure 1.

4 A Sample Application Run

You are provided by a video that demonstrates a sample run of the expected requirements of your application, please watch the video and make sure you fulfil the application requirements before you submit code.

Notice If you have any clarification questions or seeking guidelines in building your code please feel free to ask the instructor and/or any one of your TA during the designated lab hours in TEB-244.

5 Hand In

- Using NetBeans IDE export your project to ZIP and name it yourUwoId_Assignment2.zip. For example, if yourUWOId is aouda then name the archive file as aouda_Assignment2.zip. Use File → Export Project → to ZIP, then enter the zip file name.
- If you work in a team, both UwoIDs need to be in the ZIP file name, and only one submission per team.
- Submit this zip file to OWL assignment link at the due date mentioned above.