

# Project Four B: Squirrel Explorer

This project is the second one of the two week-long projects (Project Four B) that will replace Project Four. As [Project Four A \(https://canvas.wisc.edu/courses/244796/pages/project-four-a-hue-finder\)](https://canvas.wisc.edu/courses/244796/pages/project-four-a-hue-finder), it is **OPTIONAL**, but we recommend that you complete it. The Projects portion of your final grade will only include contributions from either Project Four A, Project Four B, or both if those scores improve your overall Project score.

Project Four B is an individual project. All code that you submit must be written by yourself. The rules for Project Four A also apply to this project, so make sure you review them on the [Project Four A page \(https://canvas.wisc.edu/courses/244796/pages/project-four-a-hue-finder\)](https://canvas.wisc.edu/courses/244796/pages/project-four-a-hue-finder).

## Project Introduction

The project allows users to browse the [NY Squirrel Census \(https://www.thesquirrelcensus.com/\)](https://www.thesquirrelcensus.com/), which contains data about squirrel sightings with details about the sighting and the squirrel. Users are able to view a summarization of the data (squirrel sightings over time) in a bar chart visualization, and zoom in on details by changing the time range displayed as well as selecting different attributes of the sightings to be displayed in the bar chart along with the overall sighting counts.

You are provided with a front end written with JavaFX that implements this functionality. It makes use of a skip list implementation that stores the squirrel sightings ordered by the date and time of the sighting to speed up range queries on the data. The skip list implements an ADT called SortedMultiMap, and it misses an implementation for four of its methods. You will implement all four of these methods so that the provided test methods for them pass. After completing the skip list, you will be able to run and test the front end. In its current state, it only has one selector for a data attribute (squirrel age). You will implement additional selectors and organize them visually in a GridPane.

## Setup

1. Set up a new project in your favorite IDE.
2. Download and extract the contents of the provided [starter package \(http://pages.cs.wisc.edu/~cs400/p4b\\_starter\\_files.zip\)](http://pages.cs.wisc.edu/~cs400/p4b_starter_files.zip) and import the files in the package into your project.
3. Set up the JavaFX library for your project (for [Eclipse \(https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-eclipse\)](https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-eclipse), [VS Code \(https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-vs-code\)](https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-vs-code) or [IntelliJ \(https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-intellij\)](https://canvas.wisc.edu/courses/244796/pages/installing-javaafx-in-intellij)).
4. Review the contents of the provided files

- SortedMultiMap.java - an interface that supports mapping key, value pairs and that allows multiple values per key and keeps keys ordered
- SkipList.java - an implementation of SortedMultiMap based on a skip list that misses 4 of the methods
- SkipListTest.java - a test class for SkipList.java with 11 test of which 7 fail with the starter implementation of SkipList.java
- DataInstance.java - an implementation of a generic data instance that maps string attribute names to attribute values
- SquirrelSighting.java - a subclass of DataInstance that has an additional date and time attribute
- App.java - the main program and UI code for the application
- nyc\_squirrel.csv - a copy of part of the nyc squirrel census data
- junit5.jar - class archive for the junit5 library

## SkipList: getSmallest and getLargest Methods

5. The test testSmallestKey and testLargestKey in the provided test class file. After reading through the SkipList.java file and familiarizing yourself with the skip list implementation, implement those two methods. We recommend that you start with the getSmallest method and then the getLargest method. getSmallest is supposed to return the smallest key in the skip list, while getLargest should return the largest key in it.
6. After implementing both methods, make sure that the test for both of the methods pass and you are confident that both implementations are working correctly.

## SkipList: getAll and getRange Methods

7. There are 5 additional test methods that fail in the test class. Those methods all test the getAll and getRange methods. Implement both methods so that all 5 tests pass. We recommend that you start with the getRange method. Once you have implemented getRange, you will be able to base your getAll implementation in it by calling from getRange. The getRange method requires an Iterator as the return type. You can implement the Iterator interface as an anonymous class inside the getRange method. The interface has two methods that need to be implemented, hasNext and next. The former returns a boolean indicating whether the Iterator has another element, and the latter will return the element. Once hasNext returns false and there is no element left to return, next is expected to throw a NoSuchElementException exception.  
 Tip: We recommend that you base your implementation of getRange in the existing protected helper method getInternalClosest, which will return the node that either contains the key that the Iterator will start with, or a node before that. Once you have identified the exact node that the Iterator returns as its first element, store it in a final local variable in the getRange method so that it can be used within the anonymous class that implements Iterator.
8. After implementing both methods, make sure that the provided tests for both methods pass and you are confident that both implementations are working correctly.

# RadioButtons in GridPane

9. After implementing all four methods in SkipList.java, you should be able to run App.java with the JavaFX front end. After running the app, you can load the provided data set using the **File -> Load** menu item in the app. Familiarize yourself with the front end by using it to explore the squirrel data.
10. Currently, there is only one data attribute selector for the age attribute at the bottom of the screen. Your job is to extend those selectors, and add additional radio buttons (read more about the JavaFX radio button control [on this page](https://docs.oracle.com/javafx/2/ui_controls/radio-button.htm) [\\_ \(https://docs.oracle.com/javafx/2/ui\\_controls/radio-button.htm\)](https://docs.oracle.com/javafx/2/ui_controls/radio-button.htm)) that can be used to explore the data. For this, review the provided data file and add radio buttons for attributes that seem interesting to explore. Add at least 5 additional radio buttons. When you extend the number of radio buttons, you will find that the current layout mechanism we use (a JavaFX Group with absolute positioning of the radio buttons) is not very flexible when adding more controls to the section. Change the layout mechanism of the radio buttons by using a JavaFX GridPane layout manager to organize the radio buttons in a grid. For the dimensions (number of rows / columns) of the grid, choose a setting that looks best with your particular selection of attributes.

## Submit

11. Log in to [gradescope.com](https://www.gradescope.com/) [\\_ \(https://www.gradescope.com/\)](https://www.gradescope.com/) to submit your solution. Make sure your submission contains the following two files from the starter package with your code:
  - SkipList.java
  - App.java

## Grading Rubric

Criteria	Points
Implement the getSmallest and getLargest Methods for the SkipList class	10
Implement getAll and getRange Methods for the SkipList class	20
Create a GridPane and 5 or More Radio Buttons to Select Attributes	15
<b>Total Points Possible</b>	<b>45</b>