- Programming assignment 8
- I think pretty much everything was covered in “program 8 & lab12.pdf”
- You don't have to do the hideous black/red/yellow/turquoise border for the JPanels if you don't want to, the borders from the lab are fine
  - But if you've already replicated my colors, that's fine also
- You are to get the files, the sort file and the search files as commandline parameters
  - The sort file is the first commandline parameter
  - The search file is the second commandline parameter

- Your program will do the following
  - Based on a menu item selection, read the sort file, and populate an int[] with the values read
  - Based on a menu item selection, read the search file, and populate an int[] with the values read
  - Based on a menu item selection, exit the program
- For programming assignment 1 we implemented selection sort to sort a String[]
  - We will be re-using that method, except we will be updating it to sort an int[]
  - This update should be easy
    - Replace String[] with int[]
    - Assuming the String[] is named values
      - Replace “if(values[j].compareTo(values[min]) < 0)” with “if(values[j] < values[min])”
    - That should be pretty much it

- Your program will do the following (cont)
- Based on button clicks
  - Sort a copy of the sort int[] using your implementation of selection sort
  - Add key values to an initially empy binary search tree for the values in the sort int[]
  - Add the values in the sort int[] to an initially empty TreeSet<Integer>
  - Add the values in the sort int[] to an initially empty PriorityQueue<Integer>
  - Add the values in the sort int[] to an initially empty HashSet<Integer>
  - Add the values in the sort int[] to an initially empty ArrayList<Integer>
  - Add the values in the sort int[] to an initially empty ArrayList<Integer> and sort it using java.util.Collections.sort
  - Add the values in the sort int[] to an int[]
  - Sort a copy of the sort int[] using merge sort (optional for extra credit)

- If the sort ints/add to buttons are clicked multiple times, you should either instantiate a new BinarySearchTree or clear out the TreeSet/ProrityQueue/HashSet/ArrayList/int[]
- Based on button clicks
  - Seach for each of the search int[] values in the sorted int[] using your implementation of a binary search
  - Search the binary search tree for each of the key values in the search int[]
  - Search the TreeSet for each of the key values in the search int[]
  - Search the PriorityQueue for each of the key values in the search int[]
  - Search the HashSet for each of the key values in the search int[]
  - Search the ArrayList for each of the key values in the search int[]
  - Search the ArrayList for each of the key values in the search int[] using java.util.Collection.binarySearch
  - Search the int[] for each of the key values in the search int[]
  - Search the merge sorted int[] for each of the key values in the search int[] using a binay search (optional for extra credit)

- The buttons should initially all be disabled
- When the sort file menu item is selected, and the sort file is read, the buttons associated with sorting and adding should be enabled
- Each of the search buttons should be disabled until the search file is read and the applicable sort/add button is selected
  - If the search file is read prior to a sort/add button being selected, then the corresponding search button is enabled when the sort/add button is selected
  - If the sort/add button is selected prior to the search file being read, then the coprresponding search button should be enabled when the search file is read

- Submit all of the source files that are required to execute your program as attachments in an e-mail to me by midnight on the due date with a subject of “cs 140 program 8”
- When I grade your program
  - I will be copying your files into a folder
  - Execute “javac your_last_name_in_lower_case_program8.java” or “javac your_last_name_in_lower_case_p8.java”
  - Execute “java your_last_name_in_lower_case_program8 sort_data_100000.txt search_data_100000.txt” or “java your_last_name_in_lower_case_p8 sort_data_100000.txt search_data_100000.txt”
  - Click buttons and menu items

- When I grade your program (cont)
  - If your program doesn't compile?
  - Make sure that you don't use a package
    - That will cause it to not compile (or actually change the command to run it)
  - Make sure all of your filenames and class names are correct
    - So that your program compiles
  - Here's are the files in my submission
    - garrison_p8.java
    - BinarySearchTree.java
    - BinarySearchTreeFunctions.java
    - Node.java
    - NodeFunctions.java
  - Before you submit, try compiling and executing from the commandline with the files that you are submitting

- When I grade your program (cont)
  - When I test your program, I'm going to be looking for
    - Are your times similar to mine for the sorting and adding buttons
    - Are your times similar to mine for the searching buttons
    - Are the number of found search values the same as mine
    - Are your buttons enabled/disabled appropriately
    - Do you have the correct menu items and are they in the correct order
    - Do you have one menu and is it “file”
  - There shouldn't be any major shocks about what your grade for the program is
  - Get the subject correct
  - Get the program name correct
    - Either “your_last_name_in_lower_case”_program8.java or “your_last_name_in_lower_case”_p8.java

- If you look a the class files that are created when we compile the program, there are two class files related to our inner classes
  - garrison_p8$ButtonActionListener.class
  - garrison_p8$MenuItemActionListener.class