LAB 3 REQUIREMENTS
40 points –team work

Place the following disclaimer at the beginning of each java file.

/* LAB #

***FALL 2013**

***STUDENT'S FIRST NAME, LAST NAME**

STUDENT'S FIRST NAME, LAST NAME

*** Following program is my own effort/work.**

***We did not copy this program or program segment from anyone or any website site. If proven otherwise we will accept the consequences and the actions that will be taken for cheating in this class.**

***/**

GENERAL TEAM REQUIREMENTS FOR LAB 3:

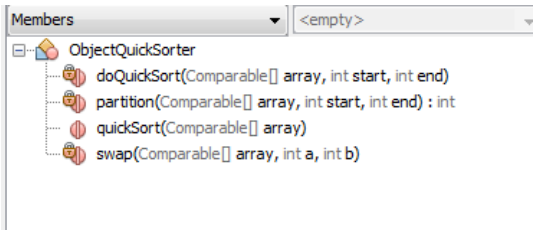
- You must divide your work fairly and equaly, please create a comment box on the top of your driver or the class to list the author of the method/methods.

A	B	C	D
LAB WORK DIST.	Susan Illinois	John California	
PointList Class	constructor I	isEmpty	
	constructor II	isFull	
	setup	gotoBeg	
	clear	gotoEnd	
	getCursor	gotoNext	
	insertBeginning	gotoPrior	

- If you choose to work on your own then decided to pick one solution to submit please state that on your comment box at the top of the client program as a comment box.
- If you choose to work on the lab together without dividing the work load then please reflect that at the top of the client program as a comment box.
- Your lab grade will be computed with the percentage of your contribution. If both team members divided the work as equally as possible then that is considered as 100% contribution from each team member. Work must be divided fairly to prevent one member completing all the work. Write your name on top of each method that you write as part of documentation.
- **Submit one lab project per team**. Both of you do not need to submit the work, agree upon yourselves on who will be submitting the work.

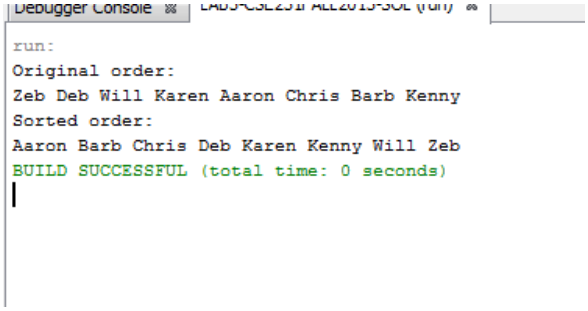
Problem 1: (10 points) Sorting Objects with the QUICK SORT Algorithm

Create an **ObjectQuickSorter** class that can sort **Comparable** objects.



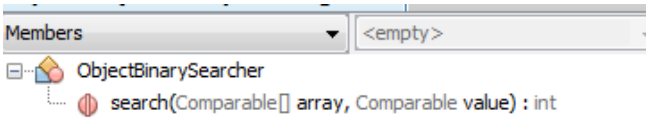
Demonstrate the class in a program that sorts an array of String objects.

Sample Run:



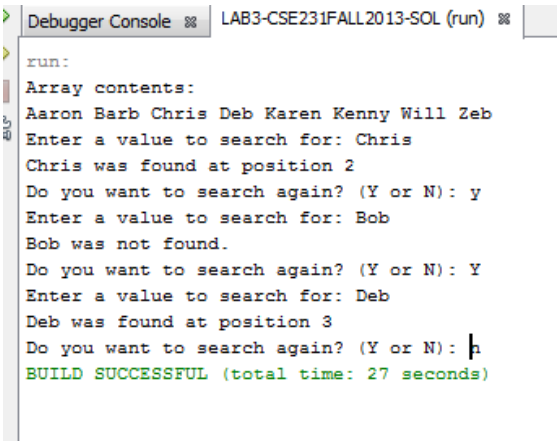
Problem 2:(10 points) Searching for Objects with the Binary Search Algorithm

Create an **ObjectBinarySearcher** class that can search an array of **Comparable** objects.



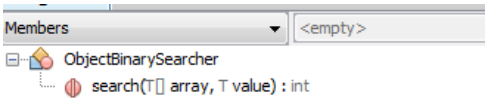
Demonstrate the class in a program that searches for a String in an array of String objects.

Sample Run:

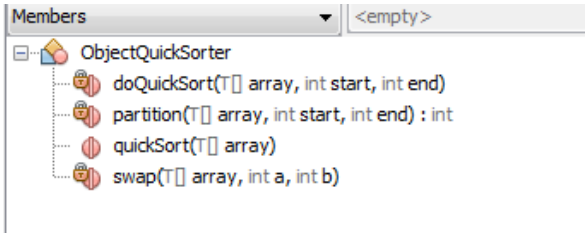


Problem 3:(10 points) Generic Binary Search

In Problem 2 you wrote an **ObjectBinarySearcher** class.
Modify the search method so it is a generic method. The method should use generic notation to accept an array of any objects that implement the Comparable interface.

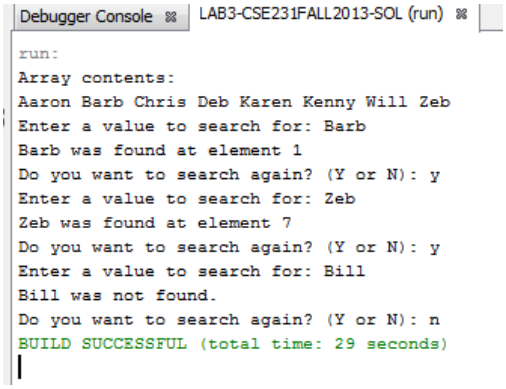


Since Binary search works on a sorted list, modify your ObjectQuickSorter class to be a generic class.



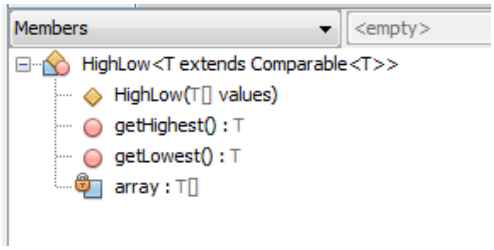
Demonstrate the class in an application.

Sample Run:



Problem 3:(10 points) Highest and Lowest Elements

Write a generic class with a type parameter constrained to any type that implements Comparable. The constructor should accept an array of such objects. The class should have methods that return the **highest** and **lowest** values in the array also please include toString method which is not shown in image below.



Demonstrate the class in an application

SUBMISSION:

Please make sure to organize your problems under their properly named folder.

For example problem 1 folder should only contain java files related to that problem. Repeat the same steps for the rest of the problems. Create a root folder named as Lab3-Lname1Lname2, zip the folder, submit only one per team.