



# Assignment 8

Arizona State University - CSE205

## Assignment #8

**Due Date Wednesday, March 25th, 7:00pm**

***Important: This is an individual assignment. Please do not collaborate.***

No late assignment will be accepted.


Make sure that you write every line of your code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report to the Dean's office.

It must be submitted on-line (Course website).

Go to "GradeScope" and upload your Java files.

## Minimal Submitted Files

You are required, but not limited, to turn in the following source file:

**Assignment8.java** (<https://canvas.asu.edu/courses/44324/files/13629869/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629869/download?wrap=1>) (More code need to be added)

**Club.java** (<https://canvas.asu.edu/courses/44324/files/13629870/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629870/download?wrap=1>) (given by the instructor, it needs to be modified for this assignment)

**President.java** (<https://canvas.asu.edu/courses/44324/files/13629883/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629883/download?wrap=1>) (given by the instructor, it needs to be modified for this assignment)

ClubNameComparator.java

MemberNumberComparator.java

CurrentPresidentComparator.java

Sorts.java

ClubManagement.java

## Requirements to get full credits in Documentation

1. The assignment number, your name, StudentID, Lecture day/time, and a class description need to be included at the top of each file/class.

2. A description of each method is also needed.
3. Some additional comments inside of methods (especially for a "main" method) to explain code that are hard to follow should be written.

## New Skills to be Applied

In addition to what has been covered in previous assignments, the use of the following items, discussed in class, will probably be needed:

Sorting

Searching

Aggregate Objects

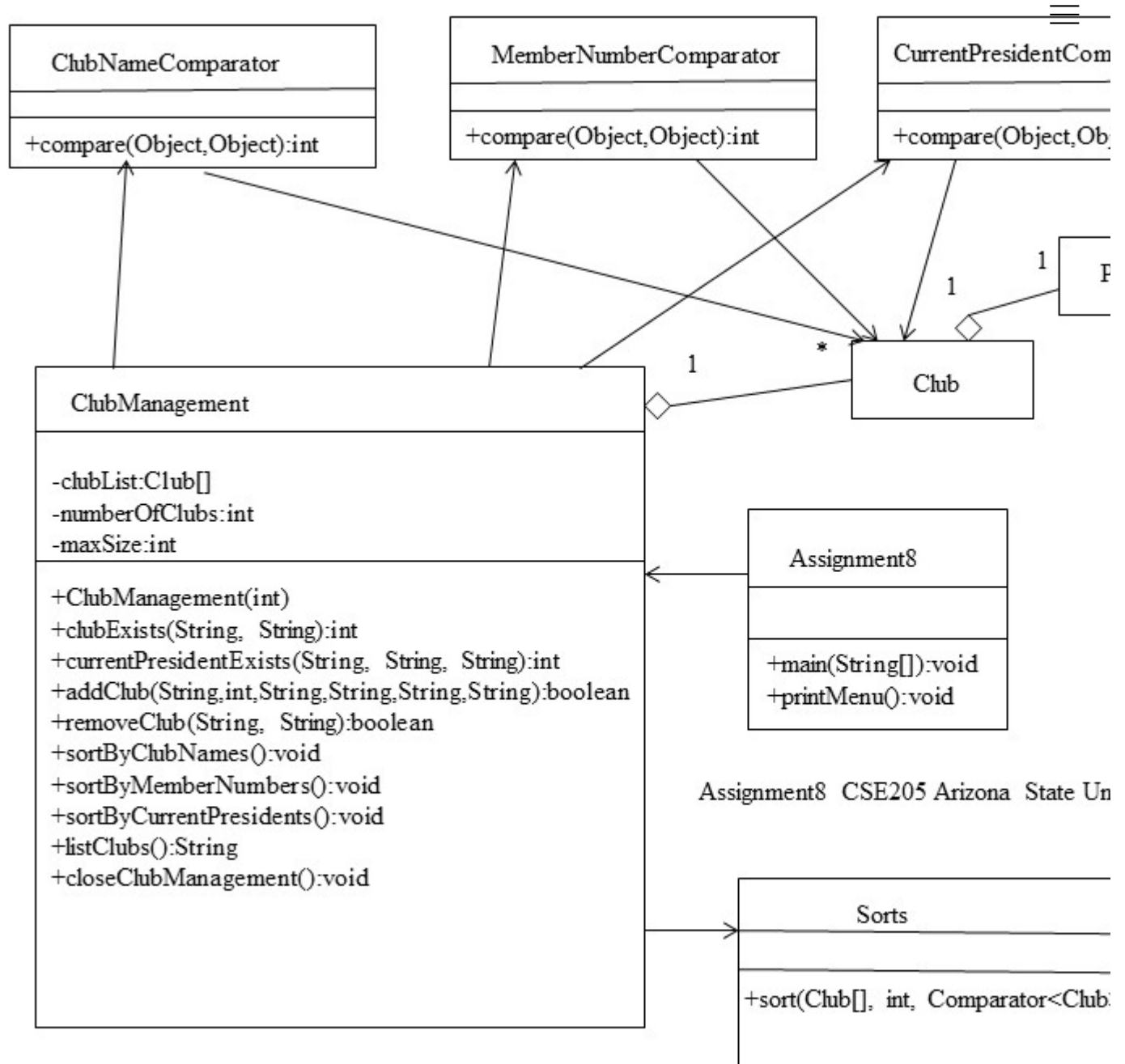
Interfaces

Serialization

File read/write

## Program Description

**Class Diagram:**



## President Class

The President class **implements the "Serializable" interface** so that its object can be stored. (The Serializable interface is defined in the "java.io" package.) It needs to define the following method:

```
public void copy(President other)
```

It needs to copy every member variable value from "other" parameter to their corresponding variable of the object itself using "this" reference. For instance, if we have two objects of President, president1 and president2, then, executing the following statement:

*president1.copy(president2)*



should resulting to have the object president1 to have the same companyName, locationCity, and locationState value as the ones for president2, but president1 and president2 should not become aliases.

## Club Class

The Club class **implements the "Serializable" interface** so that its object can be stored. (The Serializable interface is defined in the "java.io" package.) It needs to define the following method:

*public void copy(Club other)*

It needs to copy every member variable value from "other" parameter to their corresponding variable of the object itself using "this" reference.

## ClubNameComparator Class

The ClubNameComparator class **implements the "Comparator" interface (The Comparator interface is in "java.util" package.)** It needs to define the following method that was an inherited abstract method from Comparator interface:

*public int compare(Object first, Object second)*

(Note that you can also define:

*public int compare(Club first, Club second)*

instead by making the class implements Comparator<Club>.

If the first argument object has its club name less than that of the second argument, an int less than zero is returned. If the first argument object has its club name larger than that of the second argument, an int greater than zero is returned. If their club names are same, then 0 should be returned. Note that it should use String comparisons, such as using the compareTo method of the String class.

## MemberNumberComparator Class

The MemberNumberComparator class **implements the "Comparator" interface (The Comparator interface is in "java.util" package.)** It needs to define the following method that was an inherited abstract method from Comparator interface:

*public int compare(Object first, Object second)*

(Note that you can also define:

*public int compare(Club first, Club second)*

instead by making the class implements Comparator<Club>.

If the first argument object has its number of members less than that of the second argument, an int less than zero is returned. If the first argument object has its number of members larger than that of the second argument, an int greater than zero is returned. If their numbers of members are same, then 0 should be returned.

### CurrentPresidentComparator Class

The CurrentPresidentComparator class **implements the "Comparator" interface (The Comparator interface is in "java.util" package.)**. It needs to define the following method that was an inherited abstract method from Comparator interface:

```
public int compare(Object first, Object second)
```

(Note that you can also define:

```
public int compare(Club first, Club second)
```

instead by making the class implements Comparator<Club>.

If the first argument object has its last name less than that of the second argument, an int less than zero is returned. If the first argument object has its last name larger than that of the second argument, an int greater than zero is returned. If their last names are same, then their first names should be compared. (if the first argument object has its first name smaller (in alphabetical order -- compareTo method of the String class), then it should return a negative integer. If the first argument object has its first name larger, then it should return a positive integer. If their last names and first names are same, then 0 should be returned.

### Sorts Class

The Sorts class is a utility class that will be used to sort a list of Club objects. Sorting algorithms are described in the algorithm note posted under Notes section of the course web site. These algorithms sort numbers stored in an array. It is your job to modify it to sort an array of objects.

The Sorts class object will never be instantiated. It must have the following methods:

```
public static void sort(Club[] clubList, int size, Comparator<Club>)
```

Your sort method utilizes the compare method of the parameter Comparator object to sort. You can use one of Selection sort, Insertion Sort, MergeSort, or QuickSort. The parameter size specifies how many first elements should be sorted. Note that not all elements in the array should be sorted, since some of them will be null pointers.

### ClubManagement Class

The ClubManagement class has a list of Club objects that can be organized at the club management system. The club management system will be a fully encapsulated object. The ClubManagement class **implements the Serializable interface**.

It has the following attributes:



Attribute name	Attribute type	Description
clubList	an array of Club objects	A list of Club objects in the club management system
numberOfClubs	int	the number of Club objects created and stored in the clubList
maxSize	int	the maximum number of Club objects that can be stored in the clubList array. It is also the size of the clubList array.

The following public methods should be provided to interact with the club management system:

Method	Description
<i>ClubManagement(int maximumsize)</i>	<i>A Constructor of the ClubManagement class. Using the parameter value, it should initialize the member variable maxSize. Then it should instantiate an array of Club objects using the maxSize, and initialize each slot of the array to null for every index. It should also initialize the member variable numberOfClubs to 0.</i>
<i>int clubExists(String clubName, String university)</i>	<i>Search for a Club object by clubName and its university, and return the index of the object if found. Return -1 if not found. The parameters are the club name and the university of a Club object.</i>
<i>int currentPresidentExists(String</i>	<i>Search for a Club object in the club list that have the same president's first name, last name, and academic level</i>



<i>firstName, String lastName, String academicLevel)</i>	<i>as the parameter values and return the index of such object if found. Return -1 if not found.</i>
<i>boolean addClub(String clubName, int numberOfMembers, String university, String firstName, String lastName, String academicLevel)</i>	<i>Add a Club object to the club list and return true if such object was added successfully. Return false if an object with the same club name and university already exists or numberOfClubs is already same as maxSize, i.e., the array is full (the new object is not added).</i>
<i>boolean removeClub(String clubName, String university)</i>	<i>Remove a Club object of the parameter clubName and the university from the club list. Return true if the object was removed successfully. Return false if the object with the given club name and university does not exist.</i>
<i>void sortByClubNames()</i>	Sort the list of Club objects by club names. This method calls the sort method defined in the Sorts class, using the clubList, and its maxSize, an object of ClubNameComparator class.
<i>void sortByMemberNumbers()</i>	Sort the list of Club objects by their numbers of members. This method calls the sort method defined in the Sorts class, using the clubList, and its maxSize, an object of MemberNumberComparator class.
<i>void sortByCurrentPresidents()</i>	Sort the list of Club objects by their president information including its last names and first names. This method calls the sort method defined in the Sorts class, using the clubList, and its



	maxSize, an object of CurrentPresidentComparator class.
<i>String listClubs()</i>	List all Club objects in the club list. The returned string is the concatenation of each Club object information in the list for the number of numberOfClubs (do not use the size of the array here) <i>Hint: you can utilize Club's toString method to help you complete this method.</i> If there is no object in the list, This method should return the string containing "\nno club\n\n".
<i>void closeClubManagement()</i>	Closes the club management system by making the clubList empty. This can be done by making every slot of the clubList array to null, and also by setting numberOfClubs to be 0.

No input/output should occur in the club management system. User interaction should be handled only by the driver class (Assignment8 class).

You may add other methods to the class in order to make your life easier.

## Assignment8 Class

All input and output should be handled in Assignment8 class. The main method should start by displaying this updated menu in this exact format:

```
Choice\t\tAction\n
-----\t\t-----\n
A\t\tAdd Club\n
C\t\tCreate ClubManagement\n
D\t\tSearch by Club\n
E\t\tSearch by Current President\n
L\t\tList Clubs\n
N\t\tSort by Club Names\n
O\t\tSort by Club Member Numbers\n
P\t\tSort by Current Presidents\n
Q\t\tQuit\n
R\t\tRemove by Club\n
```





```
T\t\tClose ClubManagement\nU\t\tWrite Text to File\nV\t\tRead Text from File\nW\t\tSerialize ClubManagement to File\nX\t\tDeserialize ClubManagement from File\n?\t\tDisplay Help\n\n
```

Next, the following prompt should be displayed:

```
What action would you like to perform?\n
```

Read in the user input and execute the appropriate command. After the execution of each command, redisplay the prompt. Commands should be accepted in both lowercase and uppercase. The following commands are modified or new.

### **Add Club**

This part is already implemented in Assignment8.java file.  
Please see the case A in the main of Assignment8.java.

### **Create ClubManagement**

This part is already implemented in Assignment8.java file.  
Please see the case C in the main of Assignment8.java.

### **Search by Club**

This part is already implemented in Assignment8.java file.  
Please see the case D in the main of Assignment8.java.

### **Search by Current President**

This part is already implemented in Assignment8.java file.  
Please see the case E in the main of Assignment8.java.

### **Sort by Club Names**

This part is already implemented in Assignment8.java file.  
Please see the case N in the main of Assignment8.java.

### **Sort by Club Member Numbers**

This part is already implemented in Assignment8.java file.  
Please see the case O in the main of Assignment8.java.

### **Sort by Current Presidents**

This part is already implemented in Assignment8.java file.  
Please see the case P in the main of Assignment8.java.

## Remove by Club



This part is already implemented in Assignment8.java file.  
Please see the case R in the main of Assignment8.java.

## List Clubs

Each Club object information in the club list should be displayed using the toString method provided in the Club class. (and use listClubs( ) method in the ClubManagement class.)

This part is already implemented in Assignment8.java file.  
Please see the case L in the main of Assignment8.java.

## Close ClubManagement

Delete all Club objects. Then, display the following:

*club management system closed\n*

This part is already implemented in Assignment8.java file.  
Please see the case R in the main of Assignment8.java

## Write Text to File

Your program should display the following prompt:

*Please enter a file name to write:\n*

Read in the filename and create an appropriate object to get ready to read from the file. Then it should display the following prompts:

*Please enter a string to write in the file:\n*

Read in the string that a user types, say "input", then attach "\n" at the end of the string, and write it to the file. (i.e. input+"\n" string will be written in the file.)

If the operation is successful, display the following:

*FILENAME was written\n*

Replace FILENAME with the actual name of the file.

Use *try and catch* statements to catch IOException. The file should be closed in a finally statement.

## Read Text from File

Your program should display the following prompt:

*Please enter a file name to read:\n*

Read in the file name create appropriate objects to get ready to read from the file. If the operation is successful, display the following (replace FILENAME with the actual name of the file):



*FILENAME was read\n*

Then read only the first line in the file, and display:

*The first line of the file is:\n*

*CONTENT\n*

where CONTENT should be replaced by the actual first line in the file.

Your program should catch the exceptions if there are. (Use try and catch statement to catch, FileNotFoundException, and the rest of IOException.)

If the file name cannot be found, display

*FILENAME was not found\n*

where FILENAME is replaced by the actual file name.

### **Serialize ClubManagement to File**

Your program should display the following prompt:

*Please enter a file name to write:\n*

Read in the filename and write the serialized ClubManagement object (the variable clubManage1) out to it. **Note that any objects to be stored must implement Serializable interface.** The Serializable interface is defined in java.io.\* package. If the operation is successful, display the following:

*FILENAME was written\n*

Replace FILENAME with the actual name of the file.

Use *try and catch* statements to catch NotSerializableException and IOException.

### **Deserialize ClubManagement from File**

Your program should display the following prompt:

*Please enter a file name to read:\n*

Read in the file name and attempt to load the ClubManagement object from that file. Note that there is only one ClubManagement object (the variable clubManage1) in the Assignment8 class, and the first object read from the file should be assigned to the ClubManagement object. If the operation is successful, display the following (replace FILENAME with the actual name of the file):

*FILENAME was read\n*

Your program should catch the exceptions if there are.


(Use try and catch statement to catch ClassNotFoundException, FileNotFoundException, and the rest of IOException.)

See the output files for exception handling.

## Test Cases

Download the following input & output files, and save them in the same directory as Assignment5.java is located.

**input1.txt** (<https://canvas.asu.edu/courses/44324/files/13629873/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629873/download?wrap=1>)

**input2.txt** (<https://canvas.asu.edu/courses/44324/files/13629875/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629875/download?wrap=1>)

**input3.txt** (<https://canvas.asu.edu/courses/44324/files/13629876/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629876/download?wrap=1>)

**input4.txt** (<https://canvas.asu.edu/courses/44324/files/13629877/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629877/download?wrap=1>)

**output1.txt** (<https://canvas.asu.edu/courses/44324/files/13629879/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629879/download?wrap=1>)

**output2.txt** (<https://canvas.asu.edu/courses/44324/files/13629880/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629880/download?wrap=1>)

**output3.txt** (<https://canvas.asu.edu/courses/44324/files/13629881/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629881/download?wrap=1>)

**output4.txt** (<https://canvas.asu.edu/courses/44324/files/13629882/download?wrap=1>)   
(<https://canvas.asu.edu/courses/44324/files/13629882/download?wrap=1>)

## Error Handling

Your program should be robust enough to handle all test cases above.

-----

### ***What to turn in:***

-Submit your Assignment8.java, Club.java, President.java, ClubManagement.java, ClubNameComparator.java, MemberNumberComparator.java, CurrentPresidentComparator.java, Sorts.java files

via Gradescope. Make sure that your files are compiling and passing all test cases. You can submit multiple times until the assignment deadline.

**Grading Criteria:**

\_\_\_\_/ 5 Documentation (Each class file needs to have a header with your name, your information, and program description, each method needs its description and comments within your code)

\_\_\_\_/ 1 Indentation and spacing (easy to read)

\_\_\_\_/ 6 Required classes/methods and functionalities implemented

\_\_\_\_/ 8 Produces correct results (test cases – auto graded)

Total points: 20

-----

*Copyright © 2020,  
Arizona State University  
All rights reserved.*

**ASU disclaimer** [\\_\(http://www.asu.edu/asuweb/disclaimer/\)\\_](http://www.asu.edu/asuweb/disclaimer/)

Copying any content of this page and posting on the Internet will be a violation of the copy right.

