**Instructions/Design**

You will be modifying (refactoring) the Firm.java and Sorting.java programs  from Chapter 10 of our textbook.

* Download **Firm.java** and **Sorting.java** from [SourceCode.zip](https://ccsf.instructure.com/courses/38292/files/3443200/download?wrap=1) and make sure you get those programs working before you proceed.
* **Note:** You MAY NOT use any of the collection classes (such ArrayList) for this assignment.  You must 'roll your own' arrays.
* Here is an updated class hierarchy (original in Chapter 10 of our book) showing the needed updates: [StaffMemberHierarchyPDF.pdfPreview the document](https://ccsf.instructure.com/courses/38292/files/3443054/download?wrap=1)

**Modification 1: *Payable* Interface**

Modify the Firm program such that it expands its use of polymorphism by creating an *interface*called*Payable*.  You will move the *abstract* pay method from *StaffMember* into *Payable.***Reminder**: Don't forget to have **StaffMember** implement the interfaces

**Modification 2: *VacationTime* Interface**

Modify the Firm program such that all employees can be given different vacation options depending on their classification. Create an *interface* called ***VacationTime***containing one*abstract* method called **vacation** that returns the number of vacation days a person has (return an *int*).   
  
Give all employees a standard number of vacation days (14) by initializing a final/constant STANDARD\_VACATION = 14 inside *VacationTime*.

* 1. Then*override* the **vacation** method in the various classes derived from ***StaffMember***as follows:  
       
     **Volunteer** (who are not Employees): 0  
       
     **Employee**: STANDARD\_VACATION (vacation example for Employee: ***return STANDARD\_VACATION***)  
       
     **Hourly**: STANDARD\_VACATION - 7  
       
     **Executive**: STANDARD\_VACATION + extraVacation  
     **NOTE:** You will need to add*extraVacation* as instance variable to ***Executive*** and pass it in as an argument to the constructor.
  2. Call **vacation** method inside the loop already used by**payday** method so we see each StaffMember's vacation days.

*AFTER you get the above working, continue on to...*

**Modification 3: Sorting**

     Update the ***Sorting***class so that both sorting algorithms put the values in *descending order*.  THIS IS ONLY CHANGE TO Sorting.java NEEDED.  
  
**NOTE:** For sorting to work ***StaffMember*** needs to OVERRIDE **compareTo**method (we will use names for comparing).  This means ***StaffMember*** needs to *implement* the ***Comparable<StaffMember>***interface so the Sorting.java methods work.  
  
**HINT:** add a **getName (returns a String)** method to ***StaffMember***.

**Modification 4: Update Sorts**

    Sort the *a*rray of StaffMembers based on name using the updated selection OR insertion sort from **Sorting.java**. Define a **sort** method in the class that has direct access to the array.

**Modification 5: Update main()**

     Call the **sort** method in **main**before calling**payday.** Here is what main() should look like eventually:

public class Firm

{

//-------------------------------------------------------

// Creates a staff of employees for a firm and pays them.

//-------------------------------------------------------

public static void main(String[] args)

{

Staff personnel = new Staff();

personnel.sort(); //call sort method before payday

personnel.payday();

}

}

**Hints on Implementation**

* **Constructor**for***Executive*** needs to be passed the number of vacation days.  No other changes needed to ***StaffMember***
* All**StaffMembers**need to*override* the **vacation** method implemented from VacationTime interface
* Use a constant/final for vacation days. This way if the **Firm** becomes more generous in future, we only need to update one line of code.
* ***StaffMember***needs to **OVERRIDE** **compareTo**method (we will use *names* for comparing).  This means you need to implement the ***Comparable*** interface so the methods in Sorting.java work.  
  **NOE**: To help with this: add a **getName** method to StaffMember class.
* The **Staff** class needs to include a method:

**public void sort()**{  
   // 2 lines of code here  
}

Here is an example of what those 2 lines look like in **PhoneList.java** from Chapter 10:

Sorting sorts = new Sorting();  
sorts.selectionSort(friends);

**But you need to update that code above:**

1. Instantiate the Sorting class (using ***StaffMember***inside the angle brackets) as we are sorting an array of ***StaffMembers*** NOT Contacts.
2. Have the Sorting object **sorts** call the selection sort OR insertion sort method (passing the array of StaffMembers -  **staffList** - as the argument. **Note:** Sorting is a generic class and we use <> to specify parameter types in generic class creation

|  |
| --- |
| **Note:** See **PhoneList.java** and**Sorting.java** in Chapter 10 for example of using a generic class and also for  the insertion and selection sort methods. Also see **Contact.java** for overriding the *compareTo* method. |

**Input**

There is no user input for this assignment.

**Sample Output**

Name: Woody  
Address: 789 Off Rocker  
Phone: 555-0000  
Social Security Number: 010-20-3040  
Paid: 1169.23  
Vacation days available: 14  
-----------------------------------  
Name: Sam  
Address: 123 Main Line  
Phone: 555-0469  
Social Security Number: 123-45-6789  
Paid: 2923.07  
Vacation days available: 26  
-----------------------------------  
Name: Norm  
Address: 987 Suds Blvd.  
Phone: 555-8374  
Thanks!  
Vacation days available: 0  
-----------------------------------  
Name: Diane  
Address: 678 Fifth Ave.  
Phone: 555-0690  
Social Security Number: 958-47-3625  
Current hours: 40  
Paid: 422.0  
Vacation days available: 7  
-----------------------------------  
Name: Cliff  
Address: 321 Duds Lane  
Phone: 555-7282  
Thanks!  
Vacation days available: 0  
-----------------------------------  
Name: Carla  
Address: 456 Off Line  
Phone: 555-0101  
Social Security Number: 987-65-4321  
Paid: 1246.15  
Vacation days available: 14  
-----------------------------------