Data Structures and Java, Assignment 4

Algorithms, Interfaces and Inheritance

Contents

[Assignment objectives 3](#_Toc413166536)

[Part 1: Interactive Application 3](#_Toc413166537)

[Code: 3](#_Toc413166538)

[1. MenuBase.java 3](#_Toc413166539)

[2. Menu.java 3](#_Toc413166540)

[3. IntegerArraySortInterface.java 4](#_Toc413166541)

[4. IntegerArraySort.java 4](#_Toc413166542)

[5. App.java 4](#_Toc413166543)

[Create .zip file for submission: 5](#_Toc413166544)

[Part 2: Analysis, Comparison of Bubble Sort and Selection Sort 5](#_Toc413166545)

[Plotting the performance of each algorithm: 5](#_Toc413166546)

[Creating the report comparing the two algorithms: 6](#_Toc413166547)

[Part 3: Algorithm research 6](#_Toc413166548)

[Part 4: Final submission 7](#_Toc413166549)

# Assignment objectives

The following concepts are to be applied during this assignment:

1. Composition
2. Inheritance
3. Interfaces
4. Algorithms
5. Calculating and conducting a running time analysis for different algorithms

# Part 1: Interactive Application

## Code:

Create a project called A4FirstLastName.

Five (5) .java files will be created and stored in a single package called A4FirstNameLastName.

### MenuBase.java

This class contains a simple template for a menu that does two things: 1) request the length for a systematically generated array of integers which will be sorted and 2) request a selection between two sorting algorithms (Bubble Sort vs. Selection Sort).

* Define three variables in this class:
  + Scanner console
  + int lengthOfArrayToSort
  + String sortAlgorithmChoice
* Define and implement two methods:
  + public int getLengthOfArrayToSort():should specifically ask the user to enter **any number**, store it in lengthOfArrayToSort, and return it
  + public String getSortAlgorithmChoice(): gets a String from the user, stores it in sortAlgorithmChoice, and returns it

### Menu.java

This class file contains a child class that should inherit from the MenuBase.java class. Therefore, it will inherit the non-private members from the parent class.

* **Override** the getLengthOfArrayToSort () method:
  + public int getLengthOfArrayToSort (): should specifically ask the user to enter a **non-negative number**, as opposed to any number, store it in lengthOfArrayToSort, and return it
    - Notice that the int variable lengthOfArrayToSort is inherited from the parent, so there is no need to redefine it in this class.
    - Notice that when a method is overridden, NetBeans reminds that an annotation is needed. Add the required annotation.
* Note that getSortAlgorithmChoice() will not be overridden.

### IntegerArraySortInterface.java

This **interface** file contains an interface that defines two method prototypes:

* public void bubbleSort(int[] arrayToSort);
* public void selectionSort(int[] arrayToSort);

### IntegerArraySort.java

This file implements the interface IntegerArraySortInterface. This means that it will have to provide code for the two method prototypes.

* Define two methods in this class:
  + public void bubbleSort(int[] arrayToSort)
  + public void selectionSort (int[] arrayToSort)
* Implement both methods based on the material covered in class related to the different sorting algorithms.

### App.java

This class contains the main() method.

* Use **composition** to create an instance of Menu and save it in a variable called appMenu.
  + Since an object instance of another class is being created, a constructor will be called. Since a constructor is not defined on the Menu class, the default one will be used in this case.
* Once the object called menu is created, call the getLengthOfArrayToSort () method to obtain a non-negative integer from the user and store it in, int lengthOfArrayToSortInput.
* Add a method called public static int[] generateIntegerArray(int n) to the App class that returns a systematically generated integer array of length n that is reverse sorted starting at n-1 and ending at 0. For example a call generateIntegerArray(5) would result in this array: {4, 3, 2, 1, 0}.
* Call generateIntegerArray() passing it lengthOfArrayToSortInput and store the resulting array in in int[] intArrayToSort
* Call getSortAlgorithmChoice () to obtain the user’s choice in algorithm (either “bubble” or “selection”) and store it in, String sortAlgorithmChoiceInput.
* Create an object instance of the IntegerArraySort class and save it in a variable sortAlgorithm.
  + To sort using bubble sort, call bubbleSort()
  + To sort using selection sort, call selectionSort()
* To output the running time for each algorithm you will need to compute the system time before each method is called, and again after the method has returned, as follows:

start = System.nanoTime();

// call a method here

end = system. nanoTime();

// end – start is the running time

* Output the running time for each of the sorting algorithms.
  + Here is an example of the output for Bubble Sort with an array of length 20000:

Please enter any non-negative number for the length of the array:

20000

Please enter the algorithm choice (bubble or selection):

bubble

Running time of bubble sort with array of length 20000 is: 731942068

* + Here is an example of the output for Selection Sort with an array of length 20000:

Please enter any non-negative number for the length of the array:

20000

Please enter the algorithm choice (bubble or selection):

selection

Running time of selection sort with array of length 20000 is: 326686119

## Create .zip file for submission:

Place the code created above into a .zip file called A4FirstLastName\_code.zip.

# Part 2: Analysis, Comparison of Bubble Sort and Selection Sort

## Plotting the performance of each algorithm:

* Comment out the code in your main method that interacts with the user using a comment block (i.e. /\* your code here \*/
* Create new code that the data that can be used in a plot as described below:
  + Perform a running time analysis for several values of the array length. Consider starting with a small value and increasing it until consistent, significant differences in the algorithms are observable. Create a line chart with running time (on the x-axis) and the length of the array (on the y-axis). This line chart should include one series for Bubble Sort and one series for Selection Sort.
* Recall that in part 1 of the assignment, the application requests the length of the array and the sorting algorithm choice from a user. For part 2, simply write a loop to automatically increments the test value for the length of the array, generates a reverse-ordered array of the appropriate length and computes running times for each algorithm. Consider writing this data to file using a comma-separated format. The generated .csv file(s) can then be imported into Excel (or another graphing utility) where the line chart can be generated.

## Creating the report comparing the two algorithms:

* In a pdf document named A4FirstLastname\_analysis.pdf, include the following:
  + Name
  + uNID
  + Date
  + Course number and title
  + Assignment title and part (i.e. Part 2: Analysis, Comparison of Bubble Sort and Merge Sort)
  + A plot depicting the performance of each algorithm (created above)
  + A description comparing the algorithms as outlined below:
    - Briefly describe the observed running time differences between the two algorithms.
    - Do some online research including Bubble Sort and Selection Sort:
      * Referring to at least one resource from the research process, explain the observed results.
      * Referring to at least one resource from the research process and/or an understanding of the algorithm:
        + In which scenario(s) would Bubble Sort perform best?
        + In which scenario(s) would Bubble Sort perform worst?
        + In which scenario(s) would Selection Sort perform best?
        + In which scenario(s) would Selection Sort perform worst?
* Please only submit the final .pdf containing the report
  + **Please do not submit the modified version of the code** that was used to create the report data.

# Part 3: Algorithm research

Choose an interesting algorithm/family of algorithms and perform research on it.

The following are suggestions:

* Search engines (E.g. Google)
* Product recommenders (E.g. Netflix, Amazon)
* Airline/hotel pricing
* Route optimization (E.g. networks, shipping)
* Music recommenders (E.g. Pandora)
* Encryption and other security algorithms
* Compression algorithms
* Social network analysis algorithms (E.g. linkedin)
* Sentiment and text analysis algorithms

Write a one-page single-spaced summary (A4FirtsLastname\_research.pdf) including some/all of the following:

* What is the algorithm?
* Which organizations use it?
* What is the purpose of the algorithm?
* Provide a brief history of the algorithm.
* Describe how the algorithm works. This need not be a purely technical discussion, however, describe how such an algorithm might be implemented (e.g. which types of data structures are likely to be used; which kind(s) of architecture(s) or platform(s) is(are) necessary to run the algorithm?
* What are some alternative algorithms used by competing firms?
  + Provide a brief comparison of these alternatives to the chosen algorithm.
* Discuss some recent extensions to the original algorithm that the originating organization is working on or has recently implemented.

Please cite all your sources at the end of the document. Note that at least one source needs to be cited. Note also, that sources are not included in the page requirement.

# Part 4: Final submission

Turn in a single zip file for the project called A4FirstLastName\_final.zip. Include the following in this zip file:

* Your code (A4FirstLastName\_code.zip)
* Your analysis report (A4FirstLastname\_analysis.pdf)
* Your algorithm research report (A4FirtsLastname\_research.pdf)