CIS263AA Programming Assignment 13

Assignment Goal:

In this chapter you learned about generics. Generics can reduce the need for overloaded methods.

Assignment Specifications:

Create a program call **SortArrays** that sorts numeric arrays of different types. Create a method called **sortData** that accepts a single-dimension array as a generic parameter and sorts the array into descending sequence. The array should accept only number types. Use a simple bubble sort to perform the sort. Create a method that displays the sorted data. Note: I know you can use the Arrays.sort() method but, for this assignment, you should use a bubble sort.

Define compile-time arrays consisting of 10 elements each for the following data types: float, double, byte, int. Make up random data for each of the elements, or, if you want to experiment, use Java’s random number generator to create the data.

In your output method, list the data type you are sorting by using obj.getClass().getName() method.

Deliverables (what you are to submit):

1. Planning document for SortArrays.
   1. Program Outline
   2. Methods
   3. Data Items
   4. Sample Output
   5. Test Data
2. Your completed project folder in zip format.

CIS263AA Programming Assignment 13 (SortArrays)

Name: \_\_\_\_\_\_Daniel Cender\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Program Outline:** < This is an outline of what your program is to do. Be detailed. >

**Methods:** < This is a list of methods you will define in your program. >

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Access Modifier(1)** | **Method Name** | **Parameters (dataType identifier)** | **Return Type (2)** |
| Main method: | public static | main() |  | void |
| Sort Data | public static | sortData() | T[] array | void |
| Display Array | public | displayArray() | Object[] array | void |

1. Access Modifier: local, public, private, protected

2. Return Type: void, string, char, byte, short, integer, long, double, float, boolean, object, etc.

**Data Items:** < This is a list of fields (variables, constants, and objects you will need. >

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Item** | **Source (1)** | **Access Modifier (2)** | **Data Type (3)** | **Identifier** | **Notes** |
| Length incrementor | calculated | local | int | a |  |
| Index Accessor | calculated | local | int | b |  |
| Temporary Placement | calculated | static | Generic | t |  |
| Integer Array | instance | local | int | numArray |  |
| Byte Array | instance | local | byte | byteArray |  |
| Float Array | instance | local | float | floatArray |  |
| Double Array | instance | local | double | doubleArray |  |

1. Source (where the data comes from): calculated, input, constant, parameter, instance, object

2. Access Modifier: local, public, private, protected

3. Data Type: string, char, byte, short, integer, long, double, float, boolean, object, etc.

**Sample Output:** < What will the user see? >

(Array type) was passed to Sort.

Array Contents:

99999.99

99999.99

99999.99

99999.99

99999.99

99999.99

99999.99

99999.99

99999.99

99999.99

(x4 arrays)

**Test Data:** < How will you prove your program works? >

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | Integer | Byte | Float | Double |
| **Declared Data:** |  |  |  |  |
| array[0] | 2 | 34 | 34.654 | 2346765 |
| array[1] | 6 | 65 | 34 | 454 |
| array[2] | 7 | 2 | 65.34 | 234 |
| array[3] | 43 | 6 | 234.654 | 765 |
| array[4] | 5 | 4 | 75 | 34 |
| array[5] | 65 | 7 | 34 | 654.34 |
| array[6] | 234 | 23 | 4444 | 655.533 |
| array[7] | 78 | 54 | 23563 | 2333 |
| array[8] | 3456 | 6 | 2 | 54.2233543 |
| array[9] | 23 | 3 | 777 | 50 |
| **Sorted Output:** |  |  |  |  |
| array[0] | 2 | 2 | 2 | 34 |
| array[1] | 5 | 3 | 34 | 50 |
| array[2] | 6 | 4 | 34 | 54.2233543 |
| array[3] | 7 | 6 | 34.654 | 234 |
| array[4] | 23 | 6 | 65.34 | 454 |
| array[5] | 43 | 7 | 75.0 | 654.34 |
| array[6] | 65 | 23 | 234.654 | 655.533 |
| array[7] | 78 | 34 | 777.0 | 765 |
| array[8] | 234 | 54 | 4444 | 2333 |
| array[9] | 3456 | 65 | 23563 | 2346765 |

Note: You made more or fewer test cases depending on your application.