HW 03: Java Generics

Due Feb 20, 2022 by 11:59pm **Points** 1 **Submitting** a file upload **File Types** zip
Available until Feb 20, 2022 at 11:59pm

This assignment was locked Feb 20, 2022 at 11:59pm.

Purpose:

The purpose of this assignment is to have you practice using Java Generics to create your own generic class. You are **required** to do this assignment in Java, since Generics is a Java only topic. Do not forget to include the header comment at the top of each .java file you submit.

Detailed Instructions:

For this assignment, you will be designing a custom Array class using generics and adding additional features not available to normal arrays. Use the following code as the basis for your class and add the required methods. You MUST use Java Generics to implement the MyArray class. Please place all code in a package called **hw03**.

```
public class MyArray<E extends Comparable<E>> {
    //No other data fields necessary.
    private E[] data;

    public MyArray(int size) {
        this.data = (E[])(new Comparable[size]);
    }

    //This constructor can accept an array or a comma-separated list of values.
    public MyArray(E ... elements) {
        this.data = (E[])(new Comparable[elements.length]);
        //Make a deep copy to prevent shared references.
        System.arraycopy(elements, 0, this.data, 0, elements.length);
    }

    //Add other methods.
}
```

Additional Methods:

- **`get(index)`**:
  - This method shall take an integer index as a parameter and return the data at the given index.
- **`get(start, end)`**:
  - This method shall take two integer parameters `start` and `end`.
  - This method shall return a new MyArray object with values between indices `start` and `end` inclusive.

- **`put(index, value)`**:
  - This method shall take an integer index and a value and place the value at the given index.
- **`put(start, end, any number of comma separated values)`**:
  - This method shall take 3 parameters, the `start` index, `end` index, and a variable length parameter list of values of any length.
  - This method shall take the values and place them into the array replacing the values between the `start` index position and the `end` index position (inclusive).
- **`equals()`**:
  - Override the equals method.
  - This method shall take a single MyArray object as input and return whether or not this MyArray object is equal to the parameter MyArray object.
  - Two my array objects are equal if they have the same number of elements, all with the same data type, in the exact same order without needing to be sorted.
- **`max()`**: This method shall take no parameters and find and return the maximum value in the array.
- **`min()`**: This method shall take no parameters and find and return the minimum value in the array.
- **`reverse()`**:
  - This method shall take no parameters and reverse the array.
  - NOTE: It does not return a new array that is the reverse, it simply reverses the existing array.
- **`shuffle()`**:
  - This method shall randomly shuffle the array.
  - NOTE: It does not return a new array, it shuffles the existing array.
- **`leftShift(shiftDistance)`**:
  - This method shall take an integer value as a parameter and left shift the elements in the array by the given number of positions.
  - This is a circular shift so the beginning should wrap around to the end.
  - NOTE: This method does not return a new MyArray.
- **`rightShift(shiftDistance)`**:
  - This method shall take an integer value as a parameter and right shift the elements in the array by the given number of positions.
  - This is a circular shift so the end should wrap around to the beginning.
  - NOTE: This method does not return a new MyArray.
- **`size()`**: This method returns the current size of the array.
- **`toString()`**: Implement the `toString()` method so that it will print a String representation of the array.
  - The toString method should display each value of the array on one line and should separate each value with a comma and one space.
  - i.e. array 1 2 3 4 5 would display as: 1, 2, 3, 4, 5
  - Be sure not to have a trailing comma or space, otherwise your code will not pass my validation.
- **`sort()`**:

- This method shall sort the array and shall be implemented with the following sorting algorithm (given in pseudocode).
- NOTE: This method does not return a new array, just sorts the existing array.

```
while the list is not sorted:
   for each adjacent pair of items:
      if the pair of items are out of order:
         swap the pair of items
```

Testings:

- Include a Main class demonstrating that all functionality of your MyArray class works.
  - NOTE: If you fail to demonstrate that a method works, then you will receive no credit for that method, regardless of whether or not that method was correctly implemented.
- HINT: Be sure to test with Strings and a custom class because what works for numbers may not necessarily work for other data types.

Video Presentation Checklist:

- **Explain** your implementation of the following methods from your MyArray class:
  - sort()
  - equals()
  - put(start, end, ...)
  - leftShift()
- **Demonstrate** that all constructors and methods from your MyArray class are fully implemented and produce the correct output. Be sure to show results using Integer, Strings, and a custom class of your choice (should be an object other than numbers, strings, or characters.)

Submission of Deliverables:

- You will need to turn in all **.java** files related to this assignment.
- Please **zip** your **src** folder from your Eclipse project and turn in the entire **.zip** file to Canvas by the due date and time.
  - Your src folder should include the hw03 package and all the .java files.
- **DOCUMENTATION:** Make sure that you document your code with JavaDoc comments.
- **VIDEO: See above.**