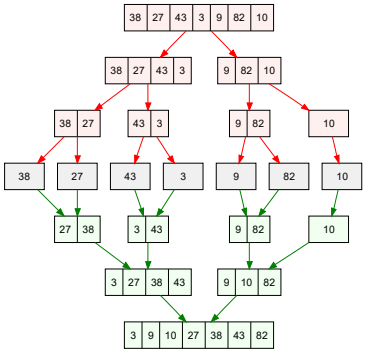
P0: MergeSort



Background

The publisher of the Cormen textbook has engaged you to write a reference implementation of the MergeSort algorithm presented in the book. They are looking for code that is faithful to the design while adapted appropriately for the Java programming idiom.

Assignment

Create a `Sorter` class that implements the merge sort algorithm described in Cormen (2009), pp 30—37.

Feature	Specification	Information
Package	`edu.metrostate.ics340.p0.aannnn.merge`	aannnn is your Student Identifier (all lower case)
Class	`Sorter`	
Constructors	N/A	Utility class with only static methods.
Methods	`public static <T extends Comparable<T>> void sort (T [] items)`	Given an array of items, sorts the array using the merge sort algorithm as described in Cormen(2009).
Input	`T [] items`	Array of Comparable values of parameterized type T Precondition: *items* cannot be null
Output	`void`	N/A

Project Requirements

	Requirements
Submission	Your submission shall be an exported Eclipse project archive zip file

	Requirements

	• **Project type**: Java Archive zip *with sources*
	• **Project Name**: P0_*AAnnnn*_MergeSort, where
	  ○ *AAnnnn* is your "student identifier" where
	  ▪ *AA:*your initials
	  ▪ *nnnn:* the 4 digits embedded in your StarID
	The zip file must contain your Java sources.
	Your classes must be in a package named with the following prefix:
	`edu.metrostate.ics340.p0.`*`aannnn.merge`*
	where *aannnn* is your student identifier as described above

Code	The Constructors and methods defined in the capabilities table will be public, spelled as specified, and with the return type as specified (void otherwise). Your submissions will be tested with an automated testing framework and therefore must adhere to the specification. Also ensure the Eclipse project is running at code Java code level 14. Your code **must** be free of compile-time errors. Your code must comply with Java coding conventions per the course Content / Resources in D2L. All public members must have Javadoc comments. You are encouraged to develop any helper methods or classes as you deem fit. It is generally advisable to make them *non-public*—only specified methods should be public.
Tests	You must also provide a test class, separate from the calculator, showing your test cases. It does not have to be a JUnit test, but that would be *highly* recommended. Your tests should demonstrate how you ensure your work meets the specification. The tests should validate preconditions and boundary cases.
APIs	You may use: • Apache Commons Lang, v 3.12 • Apache Commons Collections v4.4 • Google Guava v 31.0.1