- Write a Tractor and a TractorTest class (Tractor.java and TracktorTest.java)
- To the Tractor class
- Add Strings for both the make and model
- Add an Owner ID string
- Add a tractor value
- Add value for tractor power in terms of of horsepower .
- Add a String to hold the fuel type (diesel, gasoline, kerosene, electric, etc.), capacity and current fuel load
- Add a constructor that sets the make and model, along with the other instance variables
- Add methods to calculate the range based on fuel load and a fuel efficiency value (kilometers per liter)
- In the TractorTest program, use the 'new' operation with Tractor constructor to generate several instances of different tractor objects.
- Move all test code and place it into a run() method. Call run() from main.
- Add a method to print an attractive display of the import tractor data including power and range
- Have my code print the contents of all the tractors
- Write a TractorTaxation.java class
- Add a private ArrayList for holding Tractors, and add public convenience methods that support adding, getting and removing Tractors.
- Create a public method that loops through all Tractors in the array and prints an attractive listing.
- Add a private HashMap that stores Tractors by Owner ID, and a public method to allow retrieval by the ID .
- Add a method to calculate annual tax based on a rate-per-thousand of vehicle value – display with printed listing
- Create a FrontLoader class. Extend from Tractor class
- For the FrontLoader class, add member variables to track the front bucket size (width, capacity)
- Add an attractive print routine to the FrontLoader class that leverages the Tractor print routine from 2b, but add on information about the FrontLoader bucket (hint: use 'super' to access parent class methods with the same method name)
- Create a TractorIO class: This class will permit Tractor information to be stored (and possibly retrieved) from disk
- Create a private 'save' method for writing a single Tractor to an open file (pass the tractor and open file as parameters)
- Add public methods to load() and save() all Tractors to disk. For both methods, pass in a Tractor list and a filename as parameters.
- For all IO operations, use a try-catch() block to capture and print appropriate error messages
- Convert my TractorTaxation class to use the Singleton pattern
- Add a method to sort tractors by ID
- Print the results before and after sorting
- Add logging to TractorTaxation and TractorIO classes

- Log information messages for class creation, and for load/save methods
- Log a severe message if an error is captured in try-catch block
- Add a FileHandler to send log messages to disk