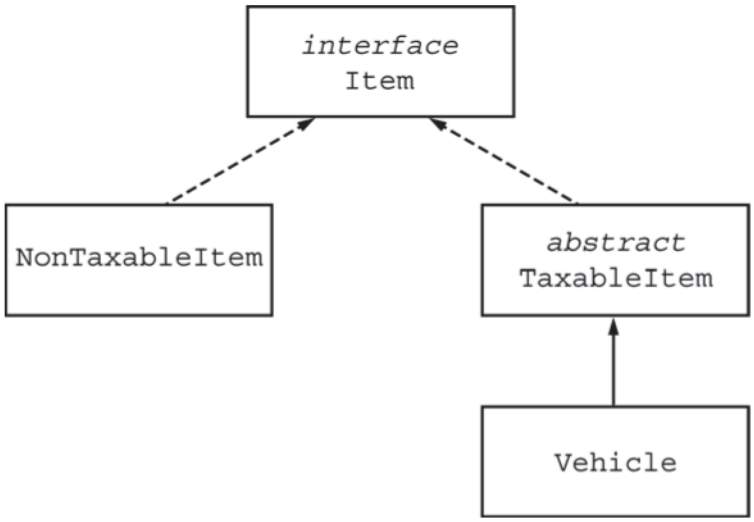
A set of classes is used to represent various items that are available for purchase. Items are either taxable or nontaxable. The purchase price of a taxable item is computed from its list price and its tax rate. The purchase price of a nontaxable item is simply its list price. Part of the class hierarchy is shown in the diagram below.



The definitions of the `Item` interface and the `TaxableItem` class are shown below.

```
public interface Item
{
  double purchasePrice();
}


public abstract class TaxableItem implements Item
{
  private double taxRate;


  public abstract double getListPrice();


  public TaxableItem(double rate)
  {  taxRate = rate;  }


  // returns the price of the item including the tax
  public double purchasePrice()
  {  /* to be implemented in part (a) */  }

}
```

(a) Write the `TaxableItem` method `purchasePrice`. The purchase price of a `TaxableItem` is its list price plus the tax on the item. The tax is computed by multiplying the list price by the tax rate. For example, if the tax rate is 0.10 (representing 10%), the purchase price of an item with a list price of $6.50 would be $7.15.

Complete method `purchasePrice` below.

```
// returns the price of the item including the tax
public double purchasePrice()
```

Testing code link:
https://www.jdoodle.com/a/26nc

(b) Create the `Vehicle` class, which extends the `TaxableItem` class. A vehicle has two parts to its list price: a dealer cost and dealer markup. The list price of a vehicle is the sum of the dealer cost and the dealer markup.

For example, if a vehicle has a dealer cost of $20,000.00, a dealer markup of $2,500.00, and a tax rate of 0.10, then the list price of the vehicle would be $22,500.00 and the purchase price (including tax) would be $24,750.00. If the dealer markup were changed to $1,000.00, then the list price of the vehicle would be $21,000.00 and the purchase price would be $23,100.00.

Your class should have a constructor that takes dealer cost, the dealer markup, and the tax rate as parameters. Provide any private instance variables needed and implement all necessary methods. Also provide a public method `changeMarkup`, which changes the dealer markup to the value of its parameter.