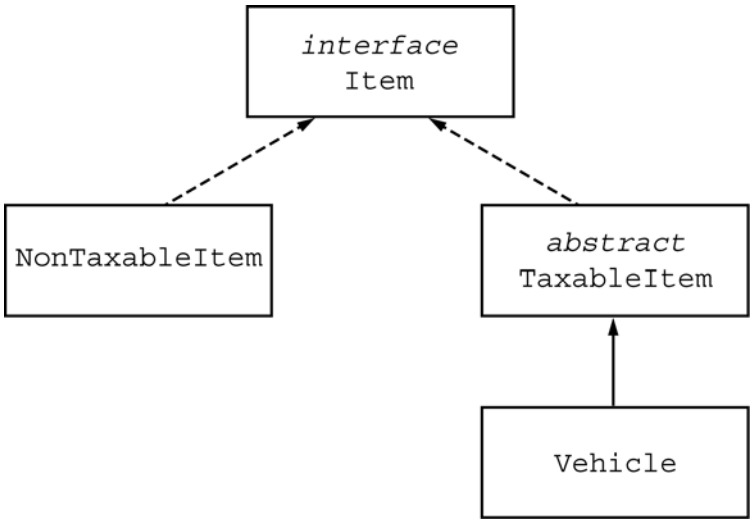
2. A set of classes is used to represent various items that are available for purchase. Items are either taxable or nontaxable. The purchase price of a taxable item is computed from its list price and its tax rate. The purchase price of a nontaxable item is simply its list price. Part of the class hierarchy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

}
```

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

(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()
```

(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.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

3. Consider the following incomplete class that stores information about a customer, which includes a name and unique ID (a positive integer). To facilitate sorting, customers are ordered alphabetically by name. If two or more customers have the same name, they are further ordered by ID number. A particular customer is "greater than" another customer if that particular customer appears later in the ordering than the other customer.

```
public class Customer
{
  // constructs a Customer with given name and ID number
  public Customer(String name, int idNum)
  {  /* implementation not shown */  }

  // returns the customer's name
  public String getName()
  {  /* implementation not shown */  }

  // returns the customer's id
  public int getID()
  {  /* implementation not shown */  }


  // returns 0 when this customer is equal to other;
  //   a positive integer when this customer is greater than other;
  //   a negative integer when this customer is less than other
  public int compareCustomer(Customer other)
  {  /* to be implemented in part (a) */  }

  // There may be fields, constructors, and methods that are not shown.
}
```

(a) Write the `Customer` method `compareCustomer`, which compares this customer to a given customer, `other`. Customers are ordered alphabetically by name, using the `compareTo` method of the `String` class. If the names of the two customers are the same, then the customers are ordered by ID number. Method `compareCustomer` should return a positive integer if this customer is greater than `other`, a negative integer if this customer is less than `other`, and 0 if they are the same.

For example, suppose we have the following `Customer` objects.

```
Customer c1 = new Customer("Smith", 1001);
Customer c2 = new Customer("Anderson", 1002);
Customer c3 = new Customer("Smith", 1003);
```

The following table shows the result of several calls to `compareCustomer`.

Method Call	Result
`c1.compareCustomer(c1)`	0
`c1.compareCustomer(c2)`	a positive integer
`c1.compareCustomer(c3)`	a negative integer

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

AP® COMPUTER SCIENCE A
2006 SCORING GUIDELINES

Question 2: Taxable Items (Design)

Part A: `purchasePrice` **2 1/2 points**

+1 call `getListPrice()`

+1 calculate correct purchase price (*no penalty if truncate/round to 2 decimal places*)

+1/2 return calculated price

Part B: `Vehicle` **6 1/2 points**

+1/2 `class Vehicle extends TaxableItem`
+1/2 `private double dealerCost`
+1/2 `private double dealerMarkup` (*no penalty if also store tax in field*)

+2 1/2 constructor
- **+1/2** `Vehicle(double ?, double ?, double ?)`
  `int/float` (*OK if match fields*)
- **+1** call parent constructor
  - **+1/2** attempt using `super`
  - **+1/2** correct call: `super(rate)` (*note: must be first line in method*)
- **+1** initialize dealer cost and markup fields
  - **+1/2** attempt (must use parameters on RHS or in mutator call)
  - **+1/2** correct

+1 `changeMarkup`
- **+1/2** `public void changeMarkup(double ?)`
  `int/float` (*OK if matches field; no penalty if returns reasonable value*)
- **+1/2** assign parameter to markup field

+1 1/2 `getListPrice`
- **+1** `public double getListPrice()`
- **+1/2** return sum of dealer cost and markup fields

Note: -1 usage if reimplement `purchasePrice` to do anything other than
`return super.purchasePrice();`

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

AP® COMPUTER SCIENCE A
2006 CANONICAL SOLUTIONS

Question 2: Taxable Items (Design)

PART A:

```
public double purchasePrice()
{
  return (1 + taxRate) * getListPrice();
}
```

PART B:

```
public class Vehicle extends TaxableItem
{
  private double dealerCost;
  private double dealerMarkup;

  public Vehicle(double cost, double markup, double rate)
  {
    super(rate);
    dealerCost = cost;
    dealerMarkup = markup;
  }

  public void changeMarkup(double newMarkup)
  {
    dealerMarkup = newMarkup;
  }

  public double getListPrice()
  {
    return dealerCost + dealerMarkup;
  }

}
```

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).