Due:
Activity (in-lab): Monday, October 31, 2011 by the end of lab
Homework: Tuesday, November 1, 2011 by 11:59 PM on Web-CAT

Goals:
By the end of this project you should be able to do the following:

- Use the protected modifier for inheritance
- Use the super reference to refer to a parent class
- Override methods in children classes

Directions:
For this assignment, you will need to create a class called InventoryItem that represents an inventory item in a store. InventoryItem should have the following characteristics:

InventoryItem (30%)

- Variables:
  - A **double** called price using the protected visibility modifier
  - A **String** called name using the protected visibility modifier
  - A private static **double** called taxRate (initialized to 0)
- Constructor:
  - Sets the values for the instance variables name and price:
    `public InventoryItem(String nameIn, double priceIn)`
- Methods:
  - `public String getName() {`
    - Returns the customer name
  - `public double calculateCost() {`
    - Returns the price including tax: price * (1 + tax)
  - `public static void setTaxRate(double taxRateIn){`
    - Sets the tax rate
  - `public String toString() {`
    - Return a String representation with name, price, and price with tax
      Example: "Computer: $789.02"
      `return name + ": $" + __________();`

```
InventoryItem.setTaxRate(0.08);
InventoryItem s1 = new InventoryItem ("Birdseed", 7.99);
s1
Birdseed: 8.6292
InventoryItem s2 = new InventoryItem ("Picture", 10.99);
s2
Picture: 11.869200000000001
```

ElectronicsItem (20%)

- Create a class called ElectronicsItem that inherits from InventoryItem. Electronics items will have all of the characteristics of inventory items and will take into account shipping costs.
```
public class ElectronicsItem extends InventoryItem {
```

- Add a **protected double** instance variable for the **weight** of the item. Add a public constant to represent the shipping cost per pound:
```
private static final ________ SHIPPING_COST = 1.5;
```

- Add a constructor to ElectronicsItem that takes a **String for name (nameIn)**, a **double for price (priceIn)**, and a **double for weight (weightIn)**. Invoke the constructor for InventoryItem using the super reserved word:
```
super(nameIn, priceIn);
```

  Add code to set the weight of the item.

- **Override** the calculateCost method to take into account shipping.

```
  public double calculateCost() {
    return super.calculateCost() + (SHIPPING_COST * weight);
  }
```

```
InventoryItem.setTaxRate(0.08);
ElectronicsItem e = new ElectronicsItem("Monitor", 100, 10.0);
e
Monitor: $123.0
```

OnlineTextItem (15%)

- Create a class to represent an online text item that users can buy (such as an electronic book or journal article). This class does not need to be instantiated as it is just a concept that represents a number of items, so it is an abstract class (more on abstract classes on Wednesday's lecture):
```
public abstract class OnlineTextItem extends InventoryItem {
```

- Write a constructor that will only call the constructor of the parent class (InventoryItem) using the super reserved word:
```
  public _____________(String nameIn, double priceIn) {
    super(_________, ____________);
  }
```

- These items are not taxed, so you'll need to override the calculateCost method to only return the price. **The price variable has been inherited from InventoryItem**:
```
  public ________ calculateCost() {
    return price;
  }
```

OnlineArticle (10%)

- Because we cannot instantiate OnlineTextItem, we will need to create subclasses that inherit from OnlineTextItem. An OnlineArticle is a electronic text that keeps track of word count in addition to everything that is done by OnlineTextItem and InventoryItem:

```
public class OnlineArticle extends OnlineTextItem {
    private int wordCount;
```

- As with your other classes, you will need a **constructor** that calls the constructor of the parent class and **initializes wordCount to 0**.
- Add a **setWordCount** method to set the wordCount variable.

OnlineBook (10%)

- OnlineBook will need to inherit from OnlineTextItem and will need to include a variable for the author's name:

```
public class OnlineBook extends OnlineTextItem {
    protected String author;
```

- Create a constructor that calls the constructor of the parent class and initializes the author String to "Author Not Listed".

- **Override** the toString method so that the author's name is listed after the name of the book in the format "book name – author's name : $price (see interactions output below).

- Add a setAuthor method to set the author's name and then try the following in the interactions pane:

```
OnlineBook book = new OnlineBook("A Novel Novel", 9.99);
book
A Novel Novel - Author Not Listed: 9.99
book.setAuthor("Jane Lane");
book
A Novel Novel - Jane Lane: 9.99
```

InventoryDriver (10%)

- For the driver program (main method only) set the tax rate to 0.05 instantiate and print each of the following objects:
  InventoryItem – name: Oil change kit, price: $39.99
  ElectronicsItem – name: Cordless phone, price: $80.00, weight: 1.8 lbs
  OnlineArticle – name: Java News, price: $8.50, wordcount: 700 words
  OnlineBook – name: Java for Noobs, price: $13.37, author: Lauren G

Project File (5%)

- Create a project file and add all of your java files above.
- Click on the UML diagram to see the class hierarchy.
- Right-click in the UML diagram and select Layout > Tree Down then click All.

Homework

- Create a child class that inherits from InventoryItem called SeasonalItem. Its constructor should take the name of the item and the price of the item. The constructor should also specify that the item is out of season (see below).
- Seasonal items are either in season (e.g. pumpkins in October) or out of season (example: Peeps anytime outside of April).
- Add the following methods:
    - setInSeason: takes a boolean parameter (true if the item is in season, false otherwise)
    - isInSeason: returns true only if the item is in season (no parameters)
- Override the `calculateCost` method as follows:
    - If the item is in season, calculate the price as in inventory item
    - If the item is not in season, return the price (with no tax) multiplied by 0.9 (discounted item).
- Override the toString method to append *In Stock* to the end of the output (as shown in InventoryItem) only if the item is in stock.
- **In order to receive credit, you will have to submit at least InventoryItem and SeasonalItem (submitting additional files will not affect your grade).**