COEN 160/275 OO Analysis, Design and Programming Winter 2014

Assignment 1 (200pts) **Due: 31st Jan, 10PM**

Note: This is an individual assignment.

This is a **console-based program** and should work **without** a GUI.

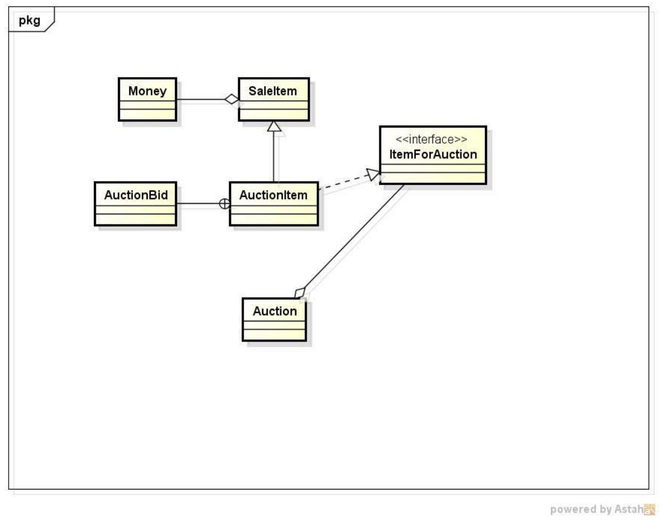
In this assignment, you will

- Become familiar with a Java IDE (recommended IDE is Eclipse) to develop your Java programs.
- Learn to use Java packages.
- Learn to develop Java classes.
- Learn to use interfaces, inner classes and enums.
- Learn to use composition and inheritance for reusability.
- Learn to use arrays and array processing
- Learn to use Java API classes for example, Date.
- Learn to use Javadocs

You will define and develop a small application consisting of classes for an auction-based sale.

In this application, you will define classes to represent **AuctionItem**s on which potential buyers can make bids and **Auction** objects which can maintain and manage a list of AuctionItems.

See the UML Class diagram below to see how the different classes are related to each other.



You will define classes (and interfaces), to represent a **SaleItem, ItemForAuction**, **AuctionItem**, **Money** and **Auction**.

Create a package called *auction*. Include classes Money, SaleItem, ItemForAuction, AuctionItem and Auction in this package. Include class Tester in a package called *assign1*.

Part 1 (100 pts)

Define a class called Money to represent the price of an item, as given below:

Class Money

Data Members:

- currencyUnits: String (default value – dollars)
- amount: double

Methods:

- A constructor, Money (amount:double) to initialize the amount. Define any other constructors (including the default constructor) that you may find necessary.
- **getAmount()** : double   returns amount
- **setAmount(amount: double)**
- **getAmountInEuro(): double**   returns the value of amount (stored in dollars) in euros.

  Note: You may use this conversion, 1 US dollar = 0.7 Euro

Class Tester

Define this class in a package called **assign1**. Define the main() in this class and test all your classes, one by one, as you define them, from the main() in this class.

To test each class, create instances of each class using every constructor you have defined and call each of the methods on these instances and check if they are returning the expected values.

Define a Java interface, ItemForAction to represent a type for AuctionItems.

interface ItemForAuction

- makeABid (bidAmount:double, email: String) : boolean
- makeABidInEuro (bidAmountInEuro:double, email: String): boolean
- sell(): void

- sell (double amount)
- cancelSale(): void
- showPrice(): void
- showPriceInEuro(): void
- showItemStatus : void
- getSoldPrice(): double
- showSaleInfo() : void

Class SaleItem

Data Members

- itemName: String
- price: Money
- status: {AVAILABLE, SOLD) Default value AVAILABLE. You may use either Strings or enums.
- dateSold :Date

Methods

- **SaleItem (itemName: String, price: double)** – creates the data member, price (of type Money) using the parameter, price (double)
- Other necessary Constructors for initialization
- **getPrice() :** double - returns the value of amount in dollars.
- **sell ()** : Should change the status to SOLD and set the dateSold to current date.
- **showItemStatus()**: void – should display the itemName, price, status and dateSold (if it is sold).
- All the necessary access methods.

Class AuctionItem extends SaleItem implements ItemForAuction

This class represents an item that is on auction. It holds information about the item itself (itemName, price and so on), current bids and buyer's info. It contains an inner class called AuctionBid which maintains info about the current bid.

Define an inner class called AuctionBid as follows:

Class AuctionBid

Data Members

- bidAmount: Money
- MINBID : a constant object of type Money initialized in the constructor.
- bidderEmail: String
- bidDate : Date

Methods

- **showBidStatus(): void** – display the bidAmount, MINBID amount, bidderEmail and bidDate. If there is no valid bid at the time, display a message, "No current bids".
- **clear() : void –** Sets the bidAmount, bidderEmail and bidDate to null and change the MINBID to 0.0.

Data Members (of AuctionItem)

- currentBid: AuctionBid
- soldPrice: Money
- buyerInfo: String

Methods (AuctionItem)

- **AuctionItem (itemName: String, price: double, minbid: double)** – Should initialize the itemName and price using the supeclass constructor. Should create the instance of the data member, **currentBid** with minbid and set the other data members in currentBid to null values.

- **makeABid (bidAmount:double, email: String) :** if the bidAmount is >= currentBid.MINBID and bidAmount > currentBid.bidAmount, currentBid.bidAmount is changed to the parameter, bidAmount; currentBid.bidderEmail is set to the parameter, email and the currentBid.bidDate is set to the current date; if the bidAmount is set, returns a true. Otherwise, a false.
- **makeABidInEuro (bidAmountInEuro:double, email: String)** : Same as makeABid(). *Make sure you convert the bidAmountInEuro to dollars before you do the comparisons.*

- **sell ():** This results in selling the item at the currentBid.bidAmount, if it is not already SOLD. If the status of the item is SOLD, display's a message, with itemName and "Item not available".

Otherwise, set the *soldPrice* to the currentBid.bidAmount and *buyerInfo* to currentBid. bidderEmail. Should change the status to SOLD and set the dateSold to currentDate (same as superclass method, sell()).

- **sell (double amount,String buyerEmail) :** If the status of the item is SOLD, display a message, with itemName and "Item not available". Otherwise, should set the *soldPrice* to the parameter amount (should convert the double into a Money object) and *buyerInfo* to the parameter, buyerEmail. Should change the status to SOLD and set the dateSold to currentDate (same as superclass method, sell()).
- **showItemStatus()** : Should display itemName, price, status (available or sold) and bid status (call currentBid.showBidStatus()).
- **getSoldPrice()**: double – returns the soldPrice
- **cancelSale():** Should clear the data in currentBid and set the buyerInfo and soldPrice to null.
- **showPriceInEuro():** should display the price of the item in Euro
- **getBuyerInfo():String –** return the value in buyerInfo.
- **showSaleInfo():** void – displays the itemName, soldPrice, date sold, and buyer's info (email).

Part 2 (50 pts)

Class Auction

This class represents an Auction site that maintains a list of AuctionItems and provides the methods to display the items in the list, find the status of a specific item, find the item with the highest bid etc.

Data Members

auctionList – a list of objects of type ItemForAuction (interface you have defined earlier). You may choose to implement the list as ab array ([]) or as an ArrayList.

Methods

Auction() : a constructor. Create the auctionList.

addAuctionItem (itemName: String, price: double, minbid: double): Creates an AuctionItem instance with the given parameters and adds it to auctionList.

addAuctionItem(item: ItemForAuction): adds the item to the auctionList

showAllAuctionItems(): Shows the itemName, price and currentBid information of each item in the auctionList.

showAuctionItemsByStatus(status: enum or String): Shows the list of items that are available or sold, based on the parameter value.

showAuctionItemStatus (itemName: String): void – should search the list for the AuctionItem with itemName (ignore case) and show its bid status.

showAuctionItemwithHighestBid () - should search the list for the AuctionItem for an item with the highest bid amount and display information of that item.

Note: You are free to define any other additional data members/methods.

Helpful Hints

Printing a Date object: You can print a Date object as shown below (this is only one of several possible ways to print a date in Java).

Example code:

```
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
String currentDate = dateFormat.format(date);
System.out.println(currentDate);
```

Test Case (50 pts): You must include the following test case in the main() and capture the output. You are free to include any test cases of your choice **in addition** to the one given.

```
AuctionItem item1 = new AuctionItem("Carnations - A
Painting",780.00,500.00);

item1.showItemStatus();

item1.makeABid(300.00, "smith@decor.com");
item1.showItemStatus();

item1.makeABid(510.00, "smith@decor.com");
item1.showItemStatus();

item1.makeABid(600.00, "jones@interiors.com");
item1.showItemStatus();

item1.sell();
item1.showSaleInfo();

item1.showItemStatus();

item1.sell(item1.getItemPrice());

item1.showItemStatus();

item1.cancelSale();

item1.makeABid(520.00, "jones@interiors.com");

item1.showItemStatus();

item1.sell(item1.getItemPrice(),”clark@museums.com”);

item1.showSaleInfo();

item1.showItemStatus();
```

// Test the Auction class

```
Auction auction1 = new Auction();

Auction1.addAuctionItem(item1);

Auction1.addAuctionItem(new AuctionItem(“IPAD 1”,100.00, 70.00));

// add at least 3 more AuctionItem instances of your choice
```

```
showAllAuctionItems();
showAuctionItemsByStatus(SOLD);
showAuctionItemsByStatus(AVAILABLE);
showAuctionItemStatus ("carnations – a painting");
showAuctionItemwithHighestBid ();
```

What to submit

1. Javadocs for all the classes in a package. **Include your name and section (TR or WF) in each file.**
2. Source code. **Create a Zip file of all your source files, using Eclipse.**

   Creating a Jar or a Zip file in Eclipse

   Ref: http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.jdt.doc.user/tasks/tasks-33.htm
3. Test cases and output. **To capture output, run your program and capture the screen shot(s) of your output window.**
4. A readme file. This file should identify the Java environment that you have used and any limitations of your program.

Point distribution

- A correctly working program with source code + test case: **65%**
- Output: **15%**
- Javadocs: **10%**

5. Using correct naming conventions and including comments (where necessary): In naming classes, packages, data members, methods and so on, you should follow the recommended Java conventions. See : http://java.about.com/od/javasyntax/a/nameconventions.htm **8%**

- Correctly naming the zipped file that includes a readme with your name + section: **2%**

How to submit

You should create a zip file with the items in 1-5. Name your zip file as **assign1_*yourFirstInitialLastName***. The readme file and the source code should include your student id, your name, course number, assignment number and date of submission.

Submit the zip file via Canvas.