# Summer 2015

# **Wedding Planner Application**

OOP244 Assignment

**Milestone 4: OnShelf and CustomMade classeS V1.0**

Please download the source files provided for this milestone.

<https://github.com/Seneca-OOP244/FP2154MS4>

Add to your solution all the classes that you have implemented in the first three milestones.

**Part 1: ErrorMessage Class**

Examine the source code for the **ErrorMessage** class. This class is already coded and ready to use.

The **ErrorMessage** class is a very simple class. It is just a container for a string of 80 characters that holds an error message.

This class has the following member functions:

**ErrorMessage();** A constructor that creates an empty error message

**void clear();** Resets the error message to an empty string.

**bool isClear()const;** Returns true if the error message is empty (No Error)

**void message(const char\* s);** Sets the error message to the received string (**s**)

**const char\* message() const;** Returns the address of the message held by the object.

We use an **ErrorMessage** object to capture the status of the **OnShelf** and **CustomMade** objects. If an error occurs during console entry, those objects set their **ErrorMessage** object to the proper message, to be shown later if needed. Using this class, we can find out if an **OnShelf** or **CustomMade** object is in an erroneous state and take appropriate action.

**Part 2: OnShelf Class**

Implement the **OnShelf** class as a class derived from your abstract **Good** class. **OnShelf** is a concrete version of your **Good** class.

Private member variables

This class has only one private member variable of type **ErrorMessage**, named **\_err**.

### Constructor:

##### This class has only one default constructor that sets:

##### UPC to an empty string

##### Name to an empty string

##### Price to zero

##### Quantity Needed to 1

##### Taxed to true

### Public member functions

##### This class implements all four pure virtual methods of the **ReadWrite** class. The signatures of these member functions are identical to those of the **ReadWrite** class.

### fstream& OnShelf::store(fstream& file)const;

This member function uses the insertion operator (<<) to write the character **“S”** and a comma **“,”** to the **file** object. This function then writes without any special formatting or spaces, all of the member variables of the **Good** base class, comma separated, in following order:

upc, name, price, taxed, quantity, quantity needed

and ends them with a new line.

Finally, this function return a reference to the file object.

For example:

S,1234,Candle,123.45,1,1,5<Newline>

### fstream& OnShelf::load(fstream& file);

This member function uses the extraction operator (>>),**ignore()** and **getline()** functions on the **istream** class to read all of the fields from the **file** object and sets the member variables of the **Good** base class using its setter functions.

When reading the fields, this function assumes that the record does not have the “**S,”** at the beginning, so it starts reading from the upc field.

This function does not perform any error detection. It returns a reference to the **file** object.

Hint: create temporary variables of type double, int and string and read the fields one by one, skipping the commas. After each read, set the member variables using the setter methods.

### ostream& OnShelf::display(ostream& ostr, bool linear) const;

This function first determines if the object has any errors using the **isClear()** member function.

If the **\_err** member variable is not clear (Some Error), this function prints the **\_err** using the **ostr** object and returns a reference to the **ostr** object.

If the **\_err** member variable is clear (No Error), this function prints the **Good** object in a format determined by the value of **linear**:

### Linear is true:

Prints the **Good** values separated by the vertical bar “|” character in following format:

1234 |Candle | 139.50| t | 1| 5|

**Upc:**  left-justified in MAX\_UPC\_LEN characters  
**Name:**  left-justified 20 characters wide  
**Cost:**  (not the price) right-justified, 2 digits after the decimal point 7 characters wide  
**Taxed:** “t” if it is taxed, space if it is not taxed, 3 characters wide  
**Qty on hand:**  right-justified 4 characters wide  
**Quantity needed:** right-justified 4 characters wide  
**One Bar** and **NO NEW LINE**

### Linear is false:

Prints one member variable per line in the following format:

Upc: 1234  
Name: Candle  
Price: 123.45  
Price after tax: 139.50   
Quantity On Hand: 1  
Quantity Needed: 5  
<Newline>

OR if not taxed

Upc: 1234  
Name: Candle  
Price: 123.45  
Price after tax: N/A   
Quantity On Hand: 1  
Quantity Needed: 5  
<Newline>

Finally, this function returns a reference to the **ostr** object.

### istream& OnShelf::conInput(istream& istr);

This function receives a reference to an **istream** object (**istr**) and extracts console input from the object in exactly the following format:

On Shelf Good Entry:  
Upc: 1234<ENTER>  
Name: Candle<ENTER>  
Price: 123.45<ENTER>  
Taxed: y<Enter>  
Quantity On Hand: 1<ENTER>  
Quantity Needed: 5<ENTER>

If **istr** is in a **fail** state, this function does nothing and returns a reference to the **istr** object.

If at any stage of the input **istr** fails (cannot read further), this function sets **\_err**  to the proper error message, skips the rest of the input process, does not set anything in the **Good** part of the object and does not display any error message.

The possible error messages are:

failure at Price Entry: **Invalid Price Entry**failure at Taxed Entry: **Invalid Taxed Entry, (y)es or (n)o**  
failure at Quantity Entry: **Invalid Quantity Entry**  
failure at Quantity Needed Entry: **Invalid Quantity Needed Entry**

Since the other member variables are text, **istr** cannot fail on those variables and there are no error messages associated with them.

For the Taxed entry, only ‘Y’,’y’,’N’ and ‘n’ are acceptable values. Validate the entry and if none of the four values are entered, set the **\_err** to the appropriate massage. Then, for consistency with an istream failure, manually set failure mode on the **istr** object by calling this function:

**istr.setstate(ios::failbit);**

Make sure that at the end of the input entry this function does not read the last new line or flush the keyboard buffer.

Finally, this function returns a reference to the **istr** object.

**Part 3: CustomMade Class**

*Please note that the CustomMade and OnShelf classes have nearly identical logic. The only difference is that the CustomMade class has one extra member variable: the date that the object is to be received.*

Implement the CustomMade class as a class derived from your abstract **Good** class. **CustomMade** is a concrete version of your **Good** class.

### Private member variables

This class has two private member variables:

* An **ErrorMessage** object, named **\_err**.
* A **Date** object, called **\_delivery**

### Constructor:

##### This class has only one default constructor that sets:

##### UPC to an empty string

##### Name to an empty string

##### Price to zero

##### Quantity Needed to 1

##### Taxed to true

## **Public member functions:**

#### Public Accessors (setters and getters)

##### **const Date& delivery() const;**

This function returns an unmodifiable reference to the **\_delivery** member variable.

##### **void delivery(const Date &value);**

This function sets the **\_delivery** attribute to the incoming value.

### Pure virtual method implementations:

##### This class implements all four pure virtual methods of the **ReadWrite** class. The signatures of these member functions are identical to those of the **ReadWrite** class.

### fstream& CustomMade::store(fstream& file)const;

This member function uses the insertion operator (<<) to write the character **“C”** and a comma **“,”** to the **file** object. This function then writes without any special formatting or spaces, all of the member variables of the **Good** base class, comma separated, in following order:

upc, name, price, taxed, quantity, quantity needed, expiry date

and ends them with a new line.

Finally, this function return a reference to the file object.

For example:

C,1234,Centerpieces,150.51,1,1,5,2017/10/12<Newline>

### fstream& OnShelf::load(fstream& file);

This member function uses the extraction operator (>>),**ignore()** and **getline()** functions on the **istream** class to read all of the fields from the **file** object and sets the member variables of the **Good** base class using its setter functions.

When reading the fields, this function assumes that the record does not have the “**C,”** at the beginning, so it starts reading from the upc field.

This function does not perform any error detection. It returns a reference to the **file** object.

Hint: create temporary variables of type double, int and string and read the fields one by one, skipping the commas. After each read, set the member variables using the setter methods.

### ostream& CustomMade::display(ostream& ostr, bool linear) const;

This function first determines if the object has any errors using the **isClear()** member function.

If the **\_err** member variable is not clear (Some Error), this function prints the **\_err** using the **ostr** object and returns a reference to the **ostr** object.

If the **\_err** member variable is clear (No Error), this function prints the **Good** object in a format determined by the value of **linear**:

### Linear is true:

Prints the **Good** values separated by the vertical bar “|” character in following format:

1234 |Centerpiece | 170.08| t | 10| 35|2017/10/12

**Upc:**  left-justified in MAX\_UPC\_LEN characters  
**Name:**  left-justified 20 characters wide  
**Cost:**  (not the price) right-justified, 2 digits after the decimal point 7 characters wide  
**Taxed:** “t” if it is taxed, space if it is not taxed, 3 characters wide  
**Qty on hand:**  right-justified 4 characters wide  
**Quantity needed:** right-justified 4 characters wide  
**Delivery Date:** prints as is  
**One Bar** and **NO NEW LINE**

### Linear is false:

Prints one member variable per line in the following format:

upc: 1234  
name: water  
price: 150.51  
Price after tax: 170.08  
Quantity On hand: 10  
Quantity Needed: 35  
Expiry date: 2017/10/12  
<Newline>

Or if not taxed:

upc: 1234  
name: water  
price: 150.50  
Price after tax: N/A  
Quantity On hand: 10  
Quantity Needed: 35  
Expiry date: 2017/10/12  
<Newline>

Finally, this function returns a reference to the **ostr** object.

### istream& CustomMade::conInput(istream& istr);

This function receives a reference to an **istream** object (**istr**) and extracts console input from the object in exactly the following format:

Custom Made Good Entry:  
Upc: 1234<ENTER>  
Name: Centerpiece<ENTER>  
Price: 150.51<ENTER>  
Taxed: y<ENTER>  
Quantity On hand: 1<ENTER>  
Quantity Needed: 5<ENTER>  
Expiry date (YYYY/MM/DD) : 2017/10/12<ENTER>

If **istr** is in a **fail** state, this function does nothing and returns a reference to the **istr** object.

If at any stage of the input **istr** fails (cannot read further), this function sets **\_err**  to the proper error message, skips the rest of the input process, does not set anything in the **Good** part of the object and does not display any error message.

The possible error messages are:

failure at Price Entry: **Invalid Price Entry**failure at Taxed Entry: **Invalid Taxed Entry, (y)es or (n)o**  
failure at Quantity Entry: **Invalid Quantity Entry**  
failure at Quantity Needed Entry: **Invalid Quantity Needed Entry**

If the Expiry date entry fails, then depending of the error code stored in your Date object, set the error message in **\_err** to:  
CIN\_FAILED: **Invalid Date Entry**

YEAR\_ERROR: **Invalid Year in Date Entry**

MON\_ERROR: **Invalid Month in Date Entry**

DAY\_ERROR: **Invalid Day in Date Entry**

Since the other member variables are text, **istr** cannot fail on those variables and there are no error messages associated with them.

For the Taxed entry, only ‘Y’,’y’,’N’ and ‘n’ are acceptable values. Validate the entry and if none of the four values are entered, set the **\_err** to the appropriate massage. Then, for consistency with an istream failure, manually set failure mode on the **istr** object by calling this function:

**istr.setstate(ios::failbit);**

Make sure that at the end of the input entry this function does not read the last new line or flush the keyboard buffer.

Finally, this function returns a reference to the **istr** object.

# **Submission:**

Please refer to your professor’s instructions for submission.