COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710
Software Construction

Fall 2017

Lab 4
Simple Cash Register Application
with Inheritance and Virtual Functions

Due: December 8, 2017 11:55pm

Points Possible: 100
Due: Written Portion: November 29th, 2017 by 11:55 pm. Submission via Canvas (25 points)
Program: December 8th, 2017 by 11:55 pm via Canvas (75 points)

No late assignments will be accepted.
No collaboration between students. Students should NOT share any project code with each other. Collaborations in any form will be treated as a serious violation of the University's academic integrity code.

Goals:

- To develop a simple cash register application taking advantage of inheritance
- To use virtual functions to simplify implementation of list with multiple types
- To understand the use of vectors that contain different type of objects
- To perform Object-Oriented Analysis, Design, and Testing
- To learn the use of vector class template

Process – 25 points:
Create a text, doc, or .pdf file named "<username>-4p" (for example, mine might read "lim-4p.txt") and provide each of the following. Please submit a text file, a .doc file or .pdf file (if you want to use diagrams or pictures, use .doc or .pdf). You are free to use tools like Visio to help you, but the final output needs to be .txt, .doc, or .pdf.

1. **Analysis:** Prepare use cases. Remember, these use cases describe how the user interacts with a cash register system (what they do, what the systems does in response, etc.). Your use cases should have enough basic details such that someone unfamiliar with the system can have an understanding of what is happening in the cash register system interface. They should not include internal technical details that the user is not (and should not) be aware of. *You must include a use case diagram show relationships between use cases. Your use case descriptions must include the following: Use case name, description, assumptions, actors, pre-conditions, post-conditions, steps (basic flow), variations (alternate flow), non-functional requirements, and issues.*
2. **Design**:
   a. Create a Class Diagram. Be sure to include:
      1) The name and purpose of the classes
      2) The member variables and the functions of the class
      3) Show the interactions between classes (for example, ownership or dependency)
      4) Any relevant notes that don't fit into the previous categories can be added
   b. Create the data flow diagrams. Show all the entities and processes that comprise the overall system and show how information flows from and to each process.

3. **Testing**: Develop lists of *specific* test cases OR a driver will substitute for this phase:
      1) For the system at large. In other words, describe inputs for "nominal" usage. You may need several scenarios. In addition, suggest scenarios for abnormal usage and show what your program should do (for example, entering a negative number for a menu might ask the user to try again).

2) For each object. (Later, these tests can be automated easily using a simple driver function in the object)

4. Implement the simple cash register application

5. **Test Results**: ***After developing the cash register system***, actually try all of your test cases (both system and unit testing). Show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this. Note: Driver output will substitute for this phase.

Program Portion:

In this Lab assignment, you will implement a simple cash register application that allows the cashier to perform the following functions:

1. Enter all items purchased and their prices
2. Compute different *types of sale*, such as regular sale and contractor sale, and then calculates the sub-total amount
3. Calculate tax from the sub-total amount to obtain the total amount payable
4. Process different *types of payments*, including cash, check and credit card payments
5. Print all the sales recorded in the cash register

You will use inheritance and virtual functions to simplify the definition of the classes that contain the above variations in functions for each of the different sales types and payment types (cash, check or credit card).

In Function (1), the cashier must itemize all the items bought by the customer with their prices and calculate the subtotal. When all the items have been entered, the cashier can end the itemization using the special option '*'. This function will first print on the receipt the sale banner and number and then prints the items and prices and sub-total.

In Function (2), called `process_sale`, the sale must be one of the two sales types: regular sale or contractor sale. It must be implemented as a virtual function, where the function `process_sale` must be overridden in the two derived classes to process regular sale and contractor sale. In regular sale, the sub-total calculated in Function (1) above is used for the next function. In contractor sales, a discount of 15% is deducted from the previous subtotal from Function (1) and a new sub-total is used. You will implement each of the two different types of sale as derived class from a base `Sale` class. It will print on the receipt the sale type (regular or contractor), new subtotal, etc.

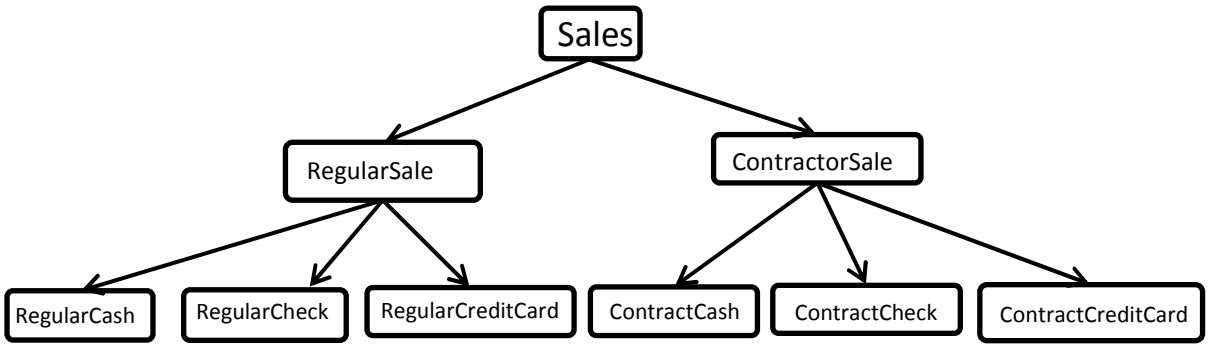
In Function (3), an 8% tax will be calculated from the sub-total to obtain the total amount.

Function (4) is called `process_payment` and is used for processing different types of payments, including cash, check and credit card payments. It must be implemented as a virtual function, where the function `process_payment` must be overridden in the three derived classes to process cash, check and credit card payments. In cash payments, the cashier will enter the cash amount received from the customer and then computes the change. In check payment, the cashier must enter the name on the check and the driver's license number. In credit card payment, the cashier must enter the name on the card, expiration date, and credit card number. All these sales information for each customer will be stored in member variables of the sales object that is of a type based on the types of sale and payment, i.e. regular or contractor and cash, check or credit card sales.

In Function (5), all sales records will be printed from vector that contains all the sales. For each of the different types of sales information, the virtual function `print_sale` function will first print the sale banner and number and then prints the items, sales types (regular or contractor), subtotal, tax, etc. Then it must print the appropriate payment information based on the derived class: regular cash, regular check, regular credit card, contractor cash, contractor check, or contractor credit card,

You must define the following inheritance hierarchy with the following classes: `Sale`, `RegularSale`, `ContractorSale`, `RegularCash`, `RegularCheck`, `RegularCreditcard`, `ContractCash`,

ContractCheck, and ContractCreditcard. These classes must contain member functions and variables as described below. The following is the inheritance hierarchy diagram.



First, define a base class named Sale that contains member variables which store the *list of item description and price, sale types information (regular sale or contractor sale), sales tax,* the *total amount, and payment information*. Also, define the appropriate accessor and mutator functions for Sale. In the Sale class, also create three ***virtual*** member functions named process_sale, process_payment and print_sale that process the appropriate sale type (regular sale or contractor sale), payment type (cash, check or credit card) and print the sales records. These virtual functions are overridden in the derived classes for different sale types and different payment types. The process_sale function will process the appropriate sales and print the descriptions of the *list of item description and price*, sale types (e.g. discount of 15% is contractor sales), *sales tax and the total amount.* The process_payment function will process either the cash, check or credit card payments and print *the payment information*. The print_sale function will print previously recorded sales depending on the sales and payment types correctly. The process_sale, process_payment and print_sale functions are **virtual functions** and when called from the appropriate object will process the sale and payment correctly and print the appropriate sale and payment information based on whether the sale object is of the type RegularCash, RegularCheck, RegularCreditcard, ContractCash, ContractCheck, or ContractCreditcard classes.

Next, define a class named RegularSale and ContractorSale that is derived from the Sale class. These classes must define the overriding ***virtual function*** process_sale to process the appropriate sale type. In regular sales, there is no special discount and process_sale function calculates the sub-total, sales tax (8%) and the total amount. However, in contractor sales, the process_sale first calculates the sub-total, then it prompts the customer for the Contractor ID. After that is entered, it calculates the new sub-total with 15% discounted and print the sales tax (8%) and the total amount.

For cash processing, define two classes named RegularCash and ContractCash that are derived from the RegularSale and ContractorSale classes respectively. These classes must define the overriding ***virtual function*** process_payment to enter and store the amount of cash received and calculate and store the change. It then prints the amount of cash received and the change. Include appropriate constructor(s).

For check processing, define two classes named RegularCheck and ContractCheck that are derived from the RegularSale and ContractorSale classes respectively. This class must contain member variables for the name on the check, and driver's license number. Include appropriate constructor(s). This class must define the overriding ***virtual function*** processs_payment to enter and store the name on the check and the driver's license number. It then prints these check payment information.

Finally, for credit card processing, define two classes named RegularCreditcard and ContractCreditcard that that are derived from the RegularSale and ContractorSale classes respectively. This class must contain member variables for the name on the card, expiration date, and credit card number. Include appropriate constructor(s). This class must define the overriding ***virtual function***

`processs_payment` to enter and store the name on the card, expiration date, and credit card number. It then prints the credit card payment information (only the last four digits are printed and the other numbers are replaced with '*'.

Your program must keep ***a single list*** of different types of sales in a vector of pointers to `Sale` objects. The program will repetitively do one of seven options: process one of the six appropriate sales and payment types or print all sales from the sales list. It will continuously prompt the user for the following options: (1) RegularCash, (2) RegularCheck, (3) RegularCreditcard, (4) ContractCash, (5) ContractCheck, and (6) ContractCreditcard, (7) Print all sales, (8) Quit.

If the option is (1) to (6), the program will continuous prompt to itemize all the items bought by the customer with their prices and calculate the subtotal. When all the items have been entered, the cashier can end the itemization using the special option '*'. It will then perform `process_sale` to process one of the two types of sales: regular sale or contractor sale. This includes adding a 8% tax and calculates the total amount payable. It then performs `process_payment` to process different types of payments, such as cash, check and credit card payments.

If the option is (7), your program must print all previous sales objects in the vector. Although the vector contains pointers to the base class `Sale`, they actually points to the derived class `RegularCash`, `RegularCheck`, `RegularCreditcard`, `ContractCash`, `ContractCheck`, and `ContractCreditcard` objects. So when the function `print_sale` is called, it will call the appropriate `print_sale` virtual function that is defined in the derived class.

The user interface

Write a menu-based and text-based user interface for the simple cash register application where the cashier can process one of the seven options. Each sale and payment will be stored in the system in a vector called `saleslist`. At any point the cashier can print out all the sales stored in the cash register.

The following is a sample of a program that uses the API provided by the `Sales` class.

Sample Usage of the API:

```
int main()
{
/*
Create a vector or "basket of sales"
Note: The data type is a pointer to the BASE CLASS Sale
*/
vector<Sale *> saleslist;

//create some objects of the various derived classes

RegularCash* rc = new RegularCash;
/*
Omitted here are codes that follows that will itemize the sale items. The next
step is to process regular sale.
*/
rc->process_sale();

/*
The next virtual function will process the appropriate cash payment amount,
change, etc. entered by the cashier.
*/
rc->process_payment();

/*
Then, add the derived class objects to the vector.
This is legal, since saleslist stores pointers to Sale objects.
```

```
*/
saleslist.push_back(rc);

ContractCheck* cc = new ContractCheck;

/*
Omitted here are codes that follows that will itemize the sale items. The next
step is to process the sales for Contractors.
*/
cc->process_sale();

/* The next virtual function will process the appropriate check payment
amount, name, driver's license number, etc. entered by the cashier.
*/
cc->process_payment();

/*
Then, add the pointer to the derived class objects to the vector.
*/
saleslist.push_back(cc);

ContractCreditcard* ccc = new ContractCreditcard;

/*
Omitted here are codes that follows that will itemize the sale items. The next
step is to process contractor sale.
*/
ccc->process_sale();

/*
The next virtual function will process the appropriate credit card
information, e.g. names, credit card number, and expiration date entered by
the cashier.
*/
ccc->process_payment();

/*
Then, add the pointer to the derived class objects to the vector.
*/
saleslist.push_back(ccc);

/*
Call the function print_sale() in the base class Sale which will call the
virtual function print_sale() as defined in the RegularCash class to output
the complete payment information on the regular sale and cash payment.
*/
saleslist[0]->print_sale();

/*
Call the function print_sale() in the base class Sale which will call the
virtual function print_sale() as defined in the ContractCheck class to output
the complete payment information on the contractor sale and check payment.
*/
saleslist[1]->print_sale();

/*
```

```
Call the function print_sale() in the base class Sale which will call the
virtual function print_sale() as defined in the ContractCreditCard class to
output the complete payment information on the contractor sale and credit card
payment.
*/
saleslist[2]->print_sale();

}
```

The above sample code does not show the menu and how the appropriate payment values and information are entered by the user into the payment objects. In your program, you must integrate all these operations.

The user interface must check for correct input value from the users. If there is any error, e.g. selecting an invalid payment value, then the program must display the appropriate error message and continue to prompt for the correct input. Your program must not exit or terminate when there is an incorrect input value.

The name of your program must be called <username>_4.cpp (for example, mine would read "lim_4.cpp"

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. You will lose points if you do not use the specific program file name, or do not have a comment block on **EVERY** program you hand in.

Your program's output need not exactly match the style of the sample output (see the end of this file for one example of sample output).

Important Notes:
You must use an object-oriented programming strategy in order to design and implement this cash register system (in other words, you will need write class definitions and use those classes, you can't just throw everything in main() or an independent function). A well-done implementation will produce a number of robust classes, many of which may be useful for future programs in this course and beyond. Some of the classes in your previous lab project may also be re-used here, possibly with some modifications. Remember good design practices discussed in class:

a) A class should do one thing, and do it well
b) Classes should NOT be highly coupled
c) Classes should be highly cohesive

- You should follow standard commenting guidelines.

- You DO NOT need any graphical user interface for this simple, text-based application. If you want to implement a visualization of some sort, then that is extra credit.

Error-Checking:

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing. Your prompts should still inform the user of what is expected of them, even if you have error-checking in place.

Submit your program through the Canvas online system. If for some disastrous reason Canvas goes down, instead e-mail your submission to TA – Tianhang Lan – at tzl0033@tigermail.auburn.edu. Canvas going down is not an excuse for turning in your work late.

You should submit the two files in digital format. No hardcopy is required for the final submission:

<username>_4.cpp
<username>_4p.txt (script of sample normal execution and a script of the results of testing)

A sample execution is shown below, where the bold fonts indicate input by the user.

```
>  lim_4

        ============================================================
        |                  Welcome to Tiger Store!                 |
        ============================================================

Select an option: (1) Regular Cash, (2) Regular Check, (3) Regular Credit Card, (4)
Contract Cash, (5) Contract Check, (6) Contract Credit Card, (7) Print all sales, or (8)
Quit:  1

Enter item:  Calculator

Enter amount:  $ 35.70

Enter item:  Battery

Enter amount:  $ 5.20

Enter item:  *

Sub-Total:  $ 40.90

Regular Sale

Tax:  $ 3.27

Total amount:  $ 44.17

Amount received:  $ 50.00

Change:  $ 5.83



        ============================================================
        |                  Welcome to Tiger Store!                 |
        ============================================================

Select an option: (1) Regular Cash, (2) Regular Check, (3) Regular Credit Card, (4)
Contract Cash, (5) Contract Check, (6) Contract Credit Card, (7) Print all sales, or (8)
Quit:  6

Enter item:  Printer

Enter amount:  $ 148.30

Enter item:  Ink Catridge

Enter amount:  $ 43.70

Enter item:  *

Sub-Total:  $ 192.00
```

Contractor Sale

Contractor ID: **24680**

Discount: $ 29.85

Sub-Total: $ 162.15

Tax: $ 12.97

Total amount: $ 175.12

Enter name on the credit card: **John Doe**

Enter credit card number: **123456782468**

Enter expiration date: **07/14**

```
============================================================
|                  Welcome to Tiger Store!                 |
============================================================
```

Select an option: (1) Regular Cash, (2) Regular Check, (3) Regular Credit Card, (4) Contract Cash, (5) Contract Check, (6) Contract Credit Card, (7) Print all sales, or (8) Quit: **5**

Enter item: **Scanner**

Enter amount: $ **105.70**

Enter item: **Printer paper**

Enter amount: $ **21.40**

Enter item: *****

Sub-Total: $ 127.10

Contractor Sale

Contractor ID: **13579**

Discount: $19.07

Sub-Total: $ 108.04

Tax: $ 8.64

Total amount: $ 116.68

Enter name on the check: **Jane Smith**

Enter driver’s license number: **24680135**

```
============================================================
|                  Welcome to Tiger Store!                 |
============================================================
```

```
Select an option: (1) Regular Cash, (2) Regular Check, (3) Regular Credit Card, (4)
Contract Cash, (5) Contract Check, (6) Contract Credit Card, (7) Print all sales, or (8)
Quit:  7

Sale #1:
   1. Calculator  $ 35.70
   2. Battery  $ 5.20
       Sub-Total:  $ 40.90
      Regular Sale
       Tax:  $ 3.27
       Total amount:  $ 44.17
       CASH
       Amount received:  $ 50.00
       Change:  $ 5.83

Sale #2:
   1. Printer  $ 148.30
   2. Ink Catridge  $ 43.70
       Sub-Total:  $ 192.00
      Contractor Sale
      Contractor ID: 24680
       Discount: $ 29.85
       Sub-Total:  $ 162.15
       Tax:  $ 12.97
       Total amount:  $ 175.12
       CREDIT CARD John Doe 123456782468 07/14

Sale #3:
   1. Scanner     $ 105.70
   2. Printer paper   $ 21.40
       Sub-Total:  $ 127.10
      Contractor Sale
      Contractor ID: 13579
       Discount: $19.07
       Sub-Total:  $ 108.04
       Tax:  $ 8.64
       Total amount:  $ 116.68
       CHECK   Jane Smith 24680135

End

        ============================================================
        |                  Welcome to Tiger Store!                 |
        ============================================================

Select an option: (1) Regular Cash, (2) Regular Check, (3) Regular Credit Card, (4)
Contract Cash, (5) Contract Check, (6) Contract Credit Card, (7) Print all sales, or (8)
Quit:  8
```