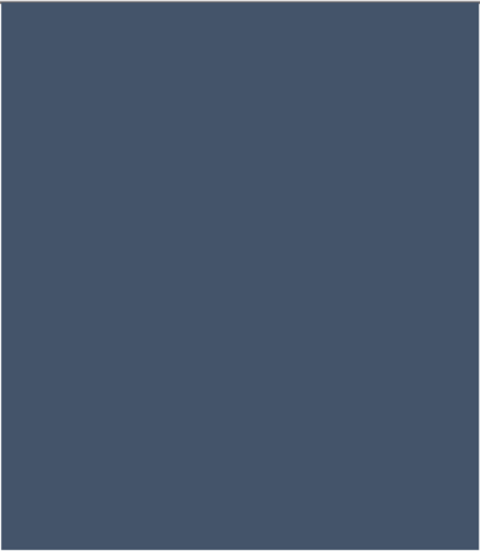


Assignment R Lucas

by Robert James Lucas

FILE	2935_ROBERT_JAMES_LUCAS_ASSIGNMENT_R_LUCAS_108537_788138608.DOCX		
TIME SUBMITTED	21-MAR-2018 06:33PM (UTC+0000)	WORD COUNT	881
SUBMISSION ID	85246047	CHARACTER COUNT	6587



Object Oriented Programming Assignment

Robert Lucas (P151516)

Contents

Table of Figures.....	2
Brief	3
Features Included	3
Classes	3
Files	4
Inheritance	5
Overloading	5
Behavioural Diagrams	6

Table of Figures

Figure 1: #include.....	3
Figure 2: Code Which Displays the Price and Quantity of the Resistors	4
Figure 3: Displays the Buying of the Component.....	4
Figure 4: Component.h	5
Figure 5: Capacitor.h.....	5
Figure 6: Asic.h	6
Figure 7: Class Diagram of Assignment.....	7
Figure 8: Use Case Diagram 1: Buying	8
Figure 9: Use Case Diagram 2: Log in/Registering	9

Brief

The purpose of this assignment is to create a theoretical electronics website via C++. This website will include a registering and login function complete with a username, password and credit check. When the 'customer' purchases an item or multiple items, the reduction of said items should be shown, for example if someone buys 5 resistors and there are 100 in stock, the stock number should reduce to 95.

It also deals with the raw materials that create said components, ordering them in order to manufacture said components. The program must be able to order more products when the product number goes below a certain number, for example if there is less than five hundred resistors the system must order one thousand more. The system should also provide discounts whenever a certain number of components are bought.

Features Included

The program contains the following coding principles

Classes

The most basic of the principles used was the class. Classes are objects that live in header files. The code within these header files is then called upon in a process called instantiation. In order to instantiate a class one must use the following code.

```
#include "Capacitor.h"
#include "Resistor.h"
#include "Photoresistor.h"
#include "Inductor.h"
#include "Transistor.h"
#include "Asic.h"
#include "Diode.h"
```

Figure 1: #include

Robert Lucas

By using the above code the header files are a part of the source file, and can be used when called upon for whatever purpose, in this case the header files contain the prices of the components, which are displayed in the 'main menu' of the program, and are used once more in the procurement of said components, multiplied by the amount of components which the customer wishes to buy.

```
Resistor*resistor = new Resistor();
cout << "Resistor:" << endl;
// << "Quantity: " << resistor->GetResistorQuantity() << endl;
ifstream resistoramount;
resistoramount.open("resistor.txt", ios::in);
resistoramount >> resistamount;
cout << "Quantity: " << resistamount << endl;
cout << "Cost: " << resistor->GetPrice() << "\n" << endl;
resistoramount.close();
```

Figure 2: Code Which Displays the Price and Quantity of the Resistors

Above is the code which shows the resistor statistics. It displays the following manner:

Quantity: 3000

Cost: 0.10

```
ifstream resistoramount;
resistoramount.open("resistor.txt");
resistoramount >> resistamount;
resistoramount.close();

ofstream resistorlamount;
Resistor*resistor = new Resistor();
cout << "A resistor, and you want how many?" << endl;
cin >> newvalue;
resistorlamount.open("resistor.txt");
changedvalue = resistamount - newvalue;
resistorlamount << changedvalue << endl;
cout << "You want " << newvalue << "? That will cost " << newvalue*resistor->GetPrice() << " pounds" << endl;
resistorlamount.close();
```

Figure 3: Displays the Buying of the Component

The code above is also for the resistor, and would ask how many resistors the customer wishes to buy. In the same section of code as section 3 are files.

Files

Files are alternate methods of storage compared to headers. To use files one must use the following codes: fstream, ifstream and ofstream. Ifstream allows users to read from files, while ofstream allows users to alter their contents, in this case the contents are the quantities of the components. Fstream allows the use of the prior two functions in the first place in the #include section. In figure 3 the code instructs for changed value to equal the original value minus the amount of components requested. That value then replaces the main value. For example someone orders one thousand resistors from a store that possesses three thousand resistors. The math is done and afterwards there are now two thousand resistors left.

Inheritance

Inheritance is a coding principle that allows one class to quite literally inherit the code from another class and use it. In this case each of the components: capacitors, resistors, transistors, diodes, photoresistors, inductors, and ASIC's all inherited from the Component header file.

```

1  #pragma once
2  class Component
3  {
4
5      float Price;
6  public:
7      Component();
8      ~Component();
9      Component(float price);
10
11
12      void SetPrice(float price);
13      float GetPrice();
14  };
15

```

Figure 4: Component.h

Above is the code of the main header file: Component.h. All of the other header files will inherit the price object. Each of the components will have their own price but will inherit the float itself from Figure 4's header file.

```

#pragma once
#include "Component.h"
class Capacitor: public Component
{
private:
public:
    Capacitor() : Component(0.15) {}; //example of overload
    Capacitor(float price) : Component(price) {};
    ~Capacitor();
}

```

Figure 5: Capacitor.h

Figure 5 shows the header file for the capacitor, which inherits the price float from the component header file, but has a value of its own: 0.15. The other components also possess separate values of their own as well.

Also as displayed above as a comment, this header possesses an example of overloading.

Overloading

Overloading is a method that allows the programmer to use a single definition multiple times, making the program much more flexible. Overloading allows the use of one definition multiple times, in this case it is the headers shown in Figures 5 and 6.

```
#pragma once
#include "Component.h"
class Asic: public Component
{
private:

public:
    Asic() : Component(1.00) {}; //example of overload
    Asic(float price) : Component(price) {};
    ~Asic();
}
```

Figure 6: Asic.h

Overloading is mainly used to allow programmers to make attach different data types onto the name as well, like int or char, but in this case only float was used.

Behavioural Diagrams

The system's workings will be shown in the following diagrams below:

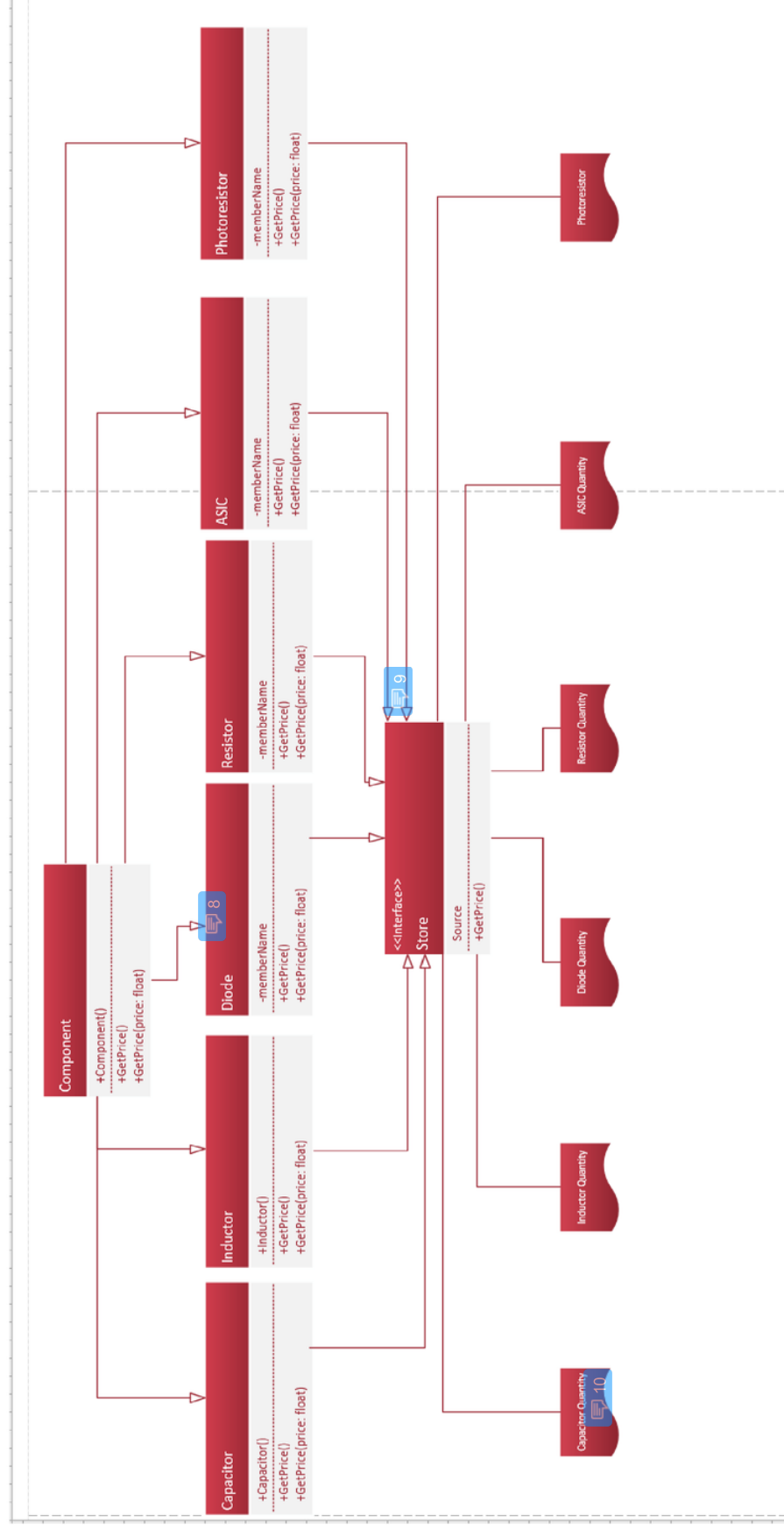


Figure 7: Class Diagram of Assignment

The diagram above details the headers, files and the main source file, labelled Store. The header Store is shown to inherit the header files, while the document files represent the file.txt's which are used for the quantities of the components.

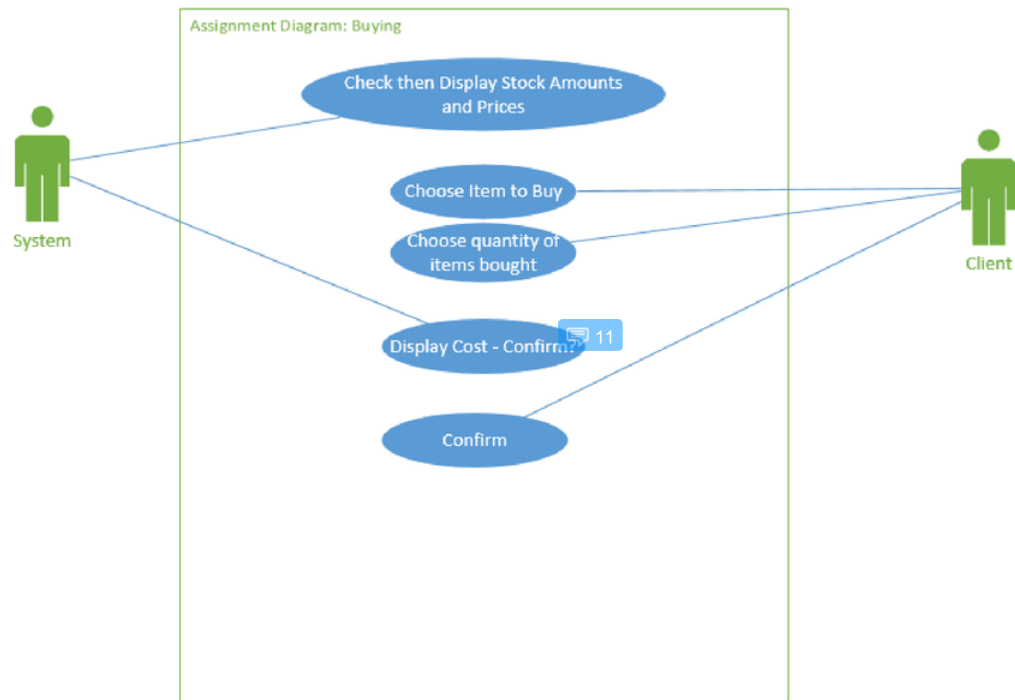


Figure 8: Use Case Diagram 1: Buying

Figure 8 represents the process of buying components whereupon the user will choose a component and then decide how many components they desire.

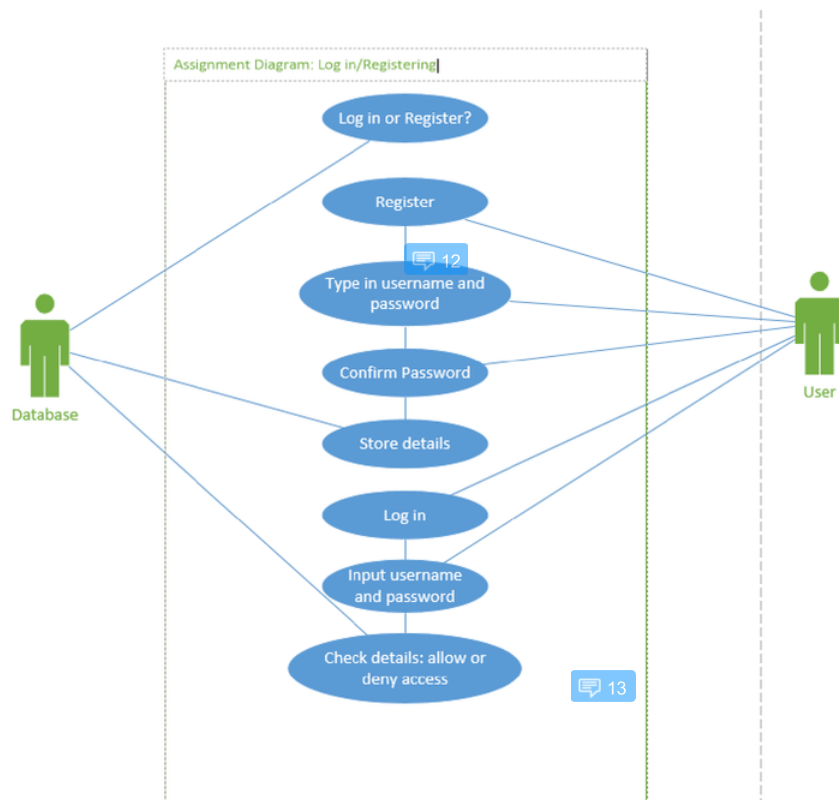


Figure 9: Use Case Diagram 2: Log in/Registering

Figure 9 displays the process of logging in or registering. The log in and registering paths have their lines of association between them to represent the different options available.

Assignment R Lucas

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

30/50

PAGE 1

PAGE 2

PAGE 3



Comment 1

This definition is a little unclear but makes way toward explaining the process of instantiation and the use of classes well done

PAGE 4



Comment 2

Good example of instantiation utilised here

PAGE 5



Comment 3

This is a very good explanation but make use of references to support your assertions



Comment 4

no comment



Comment 5

Good understanding shown of how this process works



Comment 6

The commented line could be an example of overloading, however, the line below is the better example in the depicted extract



Comment 7

Slightly clearer definition could have been given, would have made it easier with references



Comment 8

Assuming this is meant to denote inheritance, the direction of the arrow is a problem



Comment 9

Not sure what you are trying to represent here. If aggregation, this is an empty diamond. If composition this is a filled diamond



Comment 10

Not part of the syntax for a class diagram



Comment 11

Good attempt at a use-case diagram



Comment 12

What is this vertical line for ?



Comment 13

Where are the extends and or includes



Comment 14

Where are the references for this work ?