VNR VIGNANA JYOTI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Affiliated to J.N.T.U, Hyderabad) Bachupally(v), Hyderabad, Telangana, India

CSBS

II nd Year I Sem

COURSE BASED PROJECT REPORT

OBJECT ORIENTED PROGRAMMING LABORATORY

Under the Guidance of

Mrs. K. Jhansi Laxmi Bai

Professor, CSE & CSBS

CERTIFICATE

This is to certify that

21071A3224 – G PRANAV

21071A3245 – M SRAVANTH

21071A3247 – MD ABDUL SAMI

21071A3253 – MANHITH SAI

22075A3205 – KUNDAN KUMAR

have completed their course project work at CSE Department of VNR VJIET, Hyderabad entitled " " in complete fulfilment of the requirements for the award of B.Tech degree during the academic year 2022-2023.
The performance of the students was

____

. and He/she worked well as a part of a Team.

Signature of the HOD

Signature of the Staff Member

Date of the exam: ______________________

Signature of the examiners

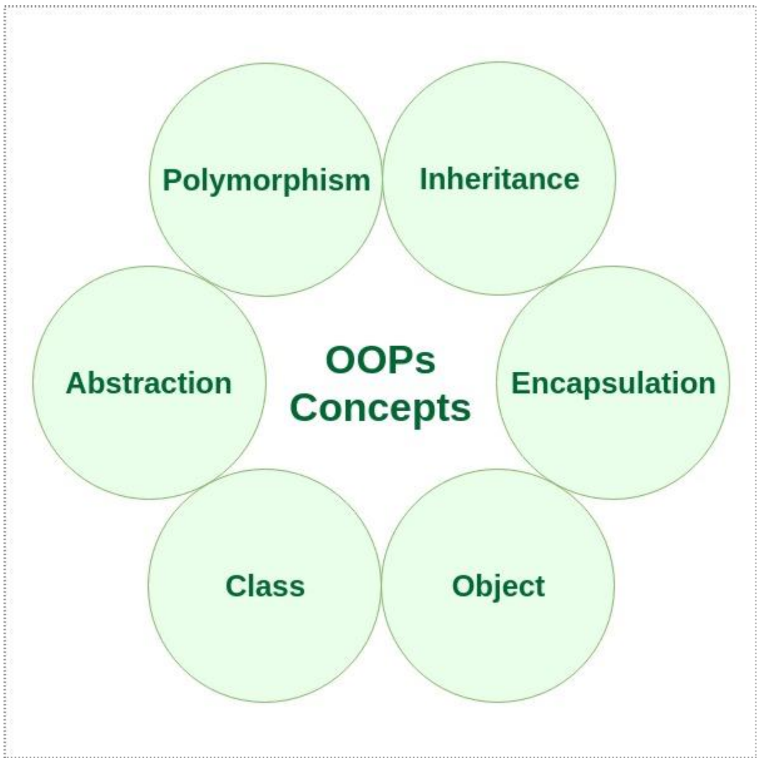
Internal Examiner

External Examiner

OBJECT ORIENTED PROGRAMMING IN C++

As the name suggests uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

CHARACTERISTICS OF OBJECT ORIENTED PROGRAMMING LANGUAGE



CLASS:

The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

A Class is a user-defined data-type which has data members and member functions.
Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behavior of the objects in a Class.

OBJECT:

An Object is an identifiable entity with some characteristics and behavior. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

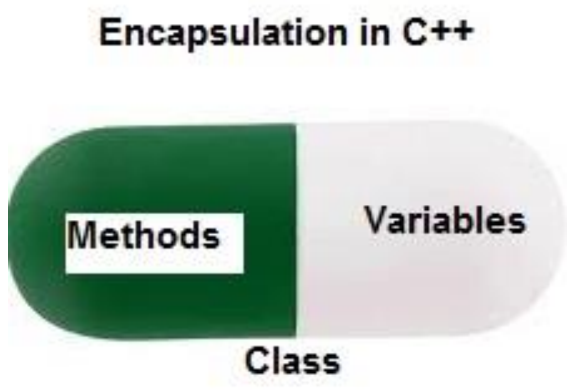
Object take up space in memory and have an associated address like a record in pascal or structure or union in C.

When a program is executed the objects interact by sending messages to one another.

Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

ENCAPSULATION:

In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.



Encapsulation also leads to *data abstraction or hiding*. As using encapsulation also hides the data.

ABSTRACTION:

Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

***Abstraction using Classes*:** We can implement Abstraction in C++ using classes. The class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to the outside world and which is not.
***Abstraction in Header files*:** One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header file. Whenever we need to calculate the power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers.

POLYMORPHISM:

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation.
C++ supports operator overloading and function overloading.
Operator Overloading: The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.
Function Overloading: Function overloading is using a single function name to perform different types of tasks.
Polymorphism is extensively used in implementing inheritance.

INHERITANCE:

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

ABSTRACT

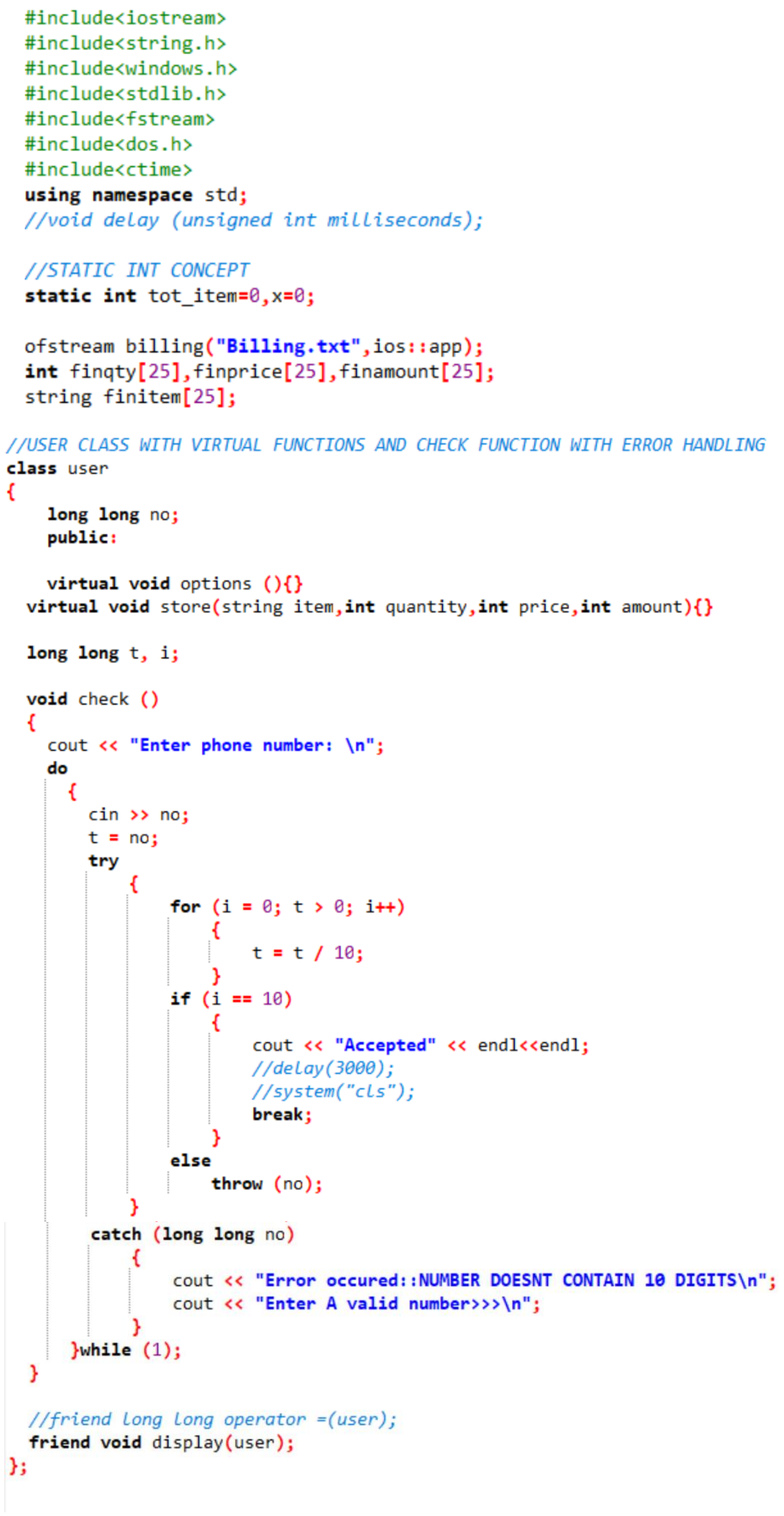
BILLING SYSTEM

The billing system is the most important function of the sales process in an organisation. A lot of thought and effort goes into setting up a streamlined billing system. In recent times, billing system software has been prominent in helping businesses to improve efficiency in their business process.

A billing system includes procedures and processes that help create bills and invoices for customers. Nowadays, billing systems include software that allows transmitting bills and invoices to the customers offline and online. Businesses need to have billing system software for the following reasons:

- To keep track of sales and payments received
- To manage cash effectively
- To prevent errors in the compilation of bills and invoice
- Optimisation of business processes

CODE

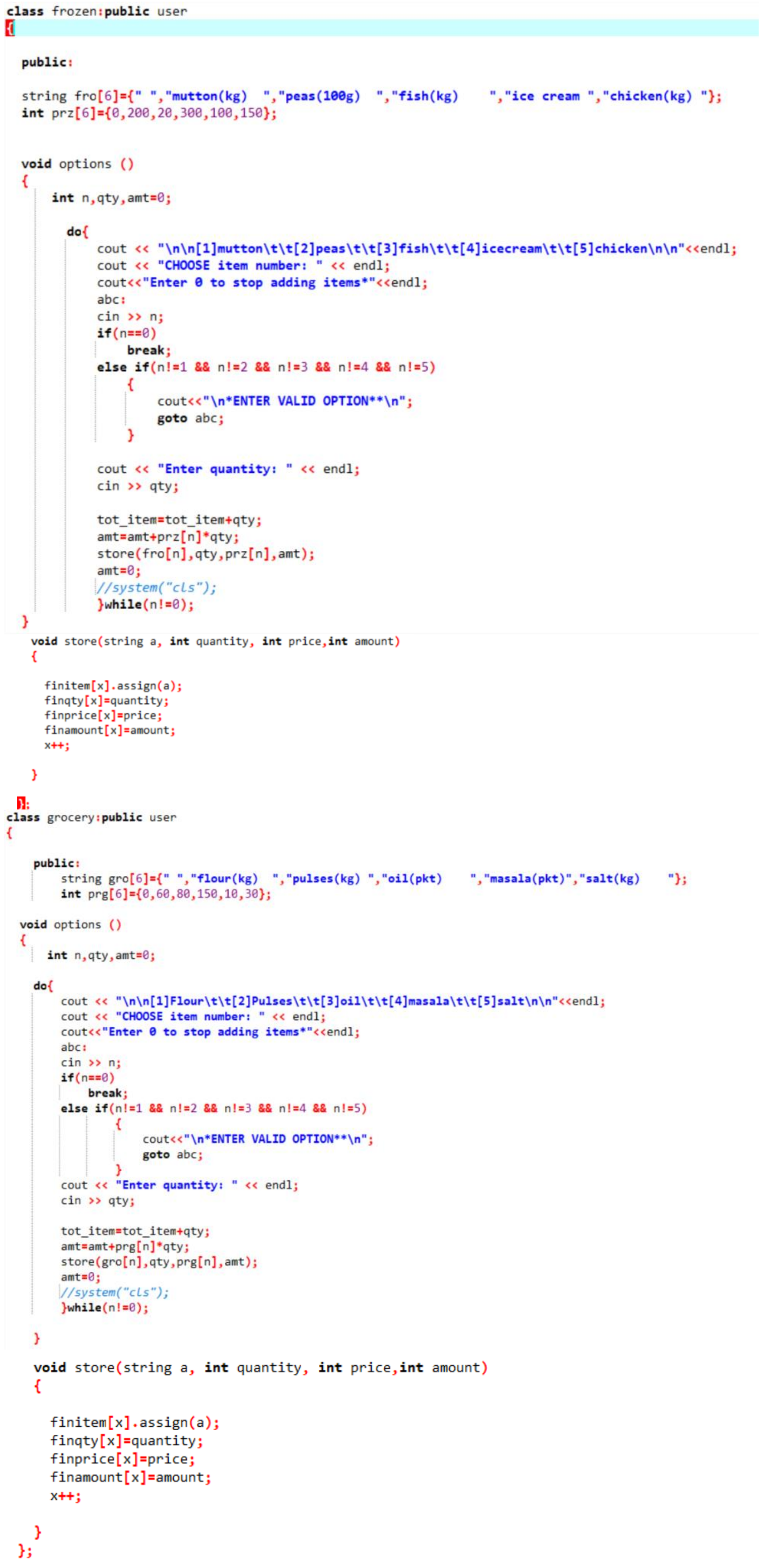


```
 #include<iostream>
 #include<string.h>
 #include<windows.h>
 #include<stdlib.h>
 #include<fstream>
 #include<dos.h>
 #include<ctime>
 using namespace std;
 //void delay (unsigned int milliseconds);

 //STATIC INT CONCEPT
 static int tot_item=0,x=0;

 ofstream billing("Billing.txt",ios::app);
 int finqty[25],finprice[25],finamount[25];
 string finitem[25];

//USER CLASS WITH VIRTUAL FUNCTIONS AND CHECK FUNCTION WITH ERROR HANDLING
class user
{
    long long no;
    public:

    virtual void options (){}
  virtual void store(string item,int quantity,int price,int amount){}

  long long t, i;

  void check ()
  {
    cout << "Enter phone number: \n";
    do
      {
        cin >> no;
        t = no;
        try
          {
            for (i = 0; t > 0; i++)
              {
                t = t / 10;
              }
            if (i == 10)
              {
                cout << "Accepted" << endl<<endl;
                //delay(3000);
                //system("cls");
                break;
              }
            else
              throw (no);
          }
        catch (long long no)
          {
            cout << "Error occured::NUMBER DOESNT CONTAIN 10 DIGITS\n";
            cout << "Enter A valid number>>>\n";
          }
      }while (1);
  }

  //friend long long operator =(user);
  friend void display(user);
};
```

Files stream

Virtual functions

Exception handling



```
class frozen:public user
{

  public:

  string fro[6]={" ","mutton(kg)  ","peas(100g)  ","fish(kg)    ","ice cream ","chicken(kg) "};
  int prz[6]={0,200,20,300,100,150};


  void options ()
  {
      int n,qty,amt=0;

        do{
          cout << "\n\n[1]mutton\t\t[2]peas\t\t[3]fish\t\t[4]icecream\t\t[5]chicken\n\n"<<endl;
          cout << "CHOOSE item number: " << endl;
          cout<<"Enter 0 to stop adding items*"<<endl;
          abc:
          cin >> n;
          if(n==0)
              break;
          else if(n!=1 && n!=2 && n!=3 && n!=4 && n!=5)
              {
                  cout<<"\n*ENTER VALID OPTION**\n";
                  goto abc;
              }

          cout << "Enter quantity: " << endl;
          cin >> qty;

          tot_item=tot_item+qty;
          amt=amt+prz[n]*qty;
          store(fro[n],qty,prz[n],amt);
          amt=0;
          //system("cls");
          }while(n!=0);
  }
    void store(string a, int quantity, int price,int amount)
    {

      finitem[x].assign(a);
      finqty[x]=quantity;
      finprice[x]=price;
      finamount[x]=amount;
      x++;

    }

};
class grocery:public user
{

    public:
      string gro[6]={" ","flour(kg)  ","pulses(kg) ","oil(pkt)    ","masala(pkt)","salt(kg)    "};
      int prg[6]={0,60,80,150,10,30};

  void options ()
  {
     int n,qty,amt=0;

    do{
      cout << "\n\n[1]Flour\t\t[2]Pulses\t\t[3]oil\t\t[4]masala\t\t[5]salt\n\n"<<endl;
      cout << "CHOOSE item number: " << endl;
      cout<<"Enter 0 to stop adding items*"<<endl;
      abc:
      cin >> n;
      if(n==0)
          break;
      else if(n!=1 && n!=2 && n!=3 && n!=4 && n!=5)
          {
              cout<<"\n*ENTER VALID OPTION**\n";
              goto abc;
          }
      cout << "Enter quantity: " << endl;
      cin >> qty;

      tot_item=tot_item+qty;
      amt=amt+prg[n]*qty;
      store(gro[n],qty,prg[n],amt);
      amt=0;
      //system("cls");
      }while(n!=0);

    }

    void store(string a, int quantity, int price,int amount)
    {

      finitem[x].assign(a);
      finqty[x]=quantity;
      finprice[x]=price;
      finamount[x]=amount;
      x++;

    }
  };
```

Inheritance

Base class

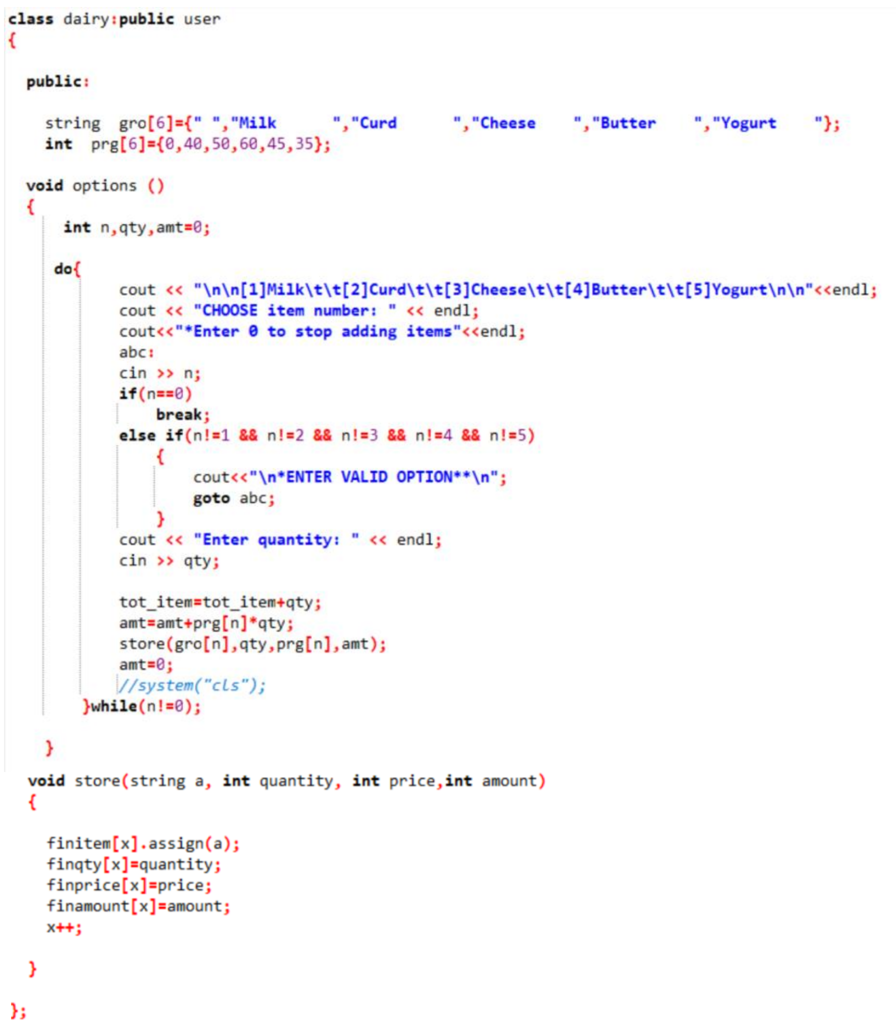
- user

Derived classes

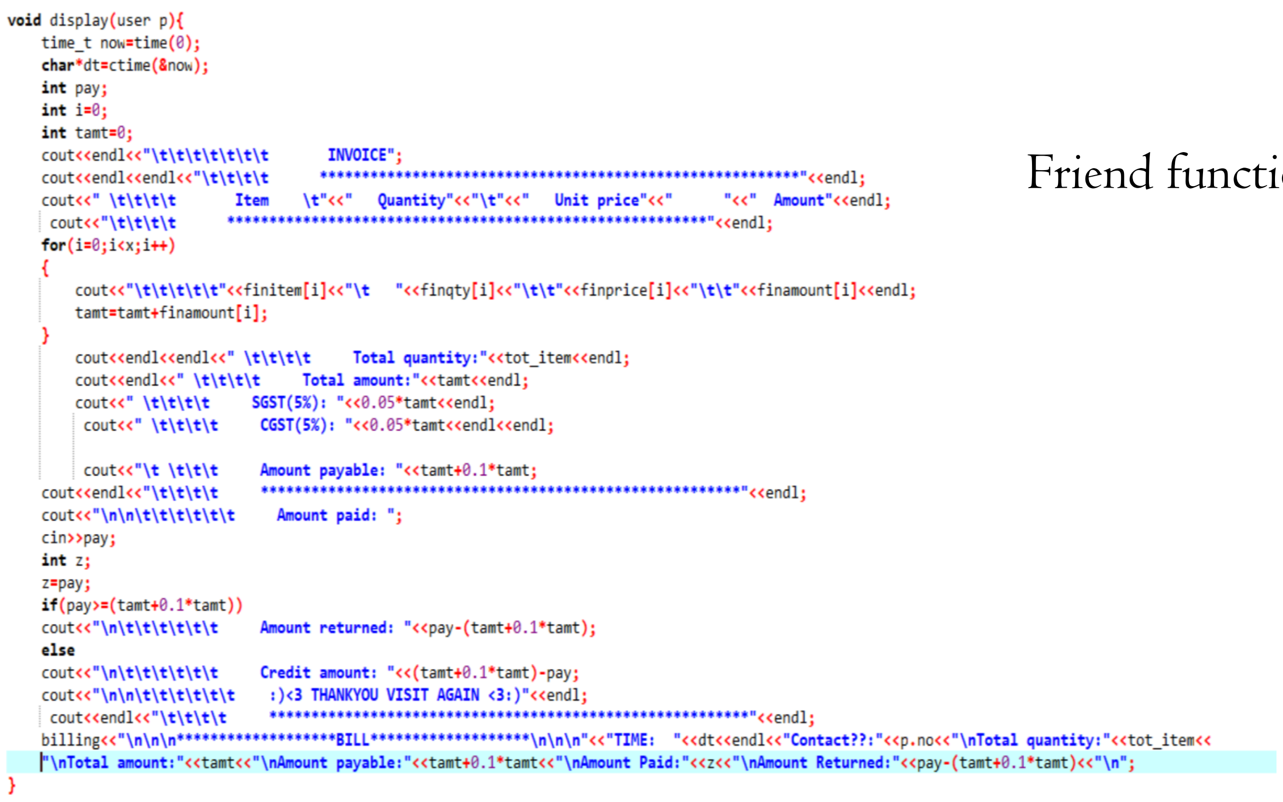
- Frozen
- Grocery
- Fresh
- Snacks
- Dairy

```
class snacks:public user
{

    public:

    string gro[6]={" ","Lays      ","Doritos    ","DairyMilk ","Kitkat    ","coke(1L)   "};
    int prg[6]={0,10,20,100,50,55};

  void options ()
  {
      int n,qty,amt=0;

    do{
        cout << "\n\n[1]Lays\t\t[2]Doritos\t\t[3]DairyMilk\t\t[4]kitkat\t\t[5]coke\n\n"<<endl;
        cout << "CHOOSE item number: " << endl;
        cout<<"Enter 0 to stop adding items*"<<endl;
        abc:
        cin >> n;
        if(n==0)
            break;
        else if(n!=1 && n!=2 && n!=3 && n!=4 && n!=5)
                {
                    cout<<"\n*ENTER VALID OPTION**\n";
                    goto abc;
                }
        cout << "Enter quantity: " << endl;
        cin >> qty;

        tot_item=tot_item+qty;
        amt=amt+prg[n]*qty;
        store(gro[n],qty,prg[n],amt);
        amt=0;
        //system("cls");
        }while(n!=0);

}
void store(string a, int quantity, int price,int amount)
  {

    finitem[x].assign(a);
    finqty[x]=quantity;
    finprice[x]=price;
    finamount[x]=amount;
    x++;

  }

};
```

```
class fresh:public user
{

  public:

  string  gro[6]={" ","Eggs(12)    ","onions(kg)  ","tomatoes(kg)","Apples(pc) ","Bananas(6)  "};
  int  prg[6]={0,60,30,40,25,35};

  void options ()
  {
      int n,qty,amt=0;

     do{
          cout << "\n\n[1]Eggs\t\t[2]Onions\t\t[3]Tomatoes\t\t[4]Apples\t\t[5]Bananas\n\n"<<endl;
          cout << "CHOOSE item number: " << endl;
          cout<<"Enter 0 to stop adding items*"<<endl;
          abc:
          cin >> n;
          if(n==0)
              break;
          else if(n!=1 && n!=2 && n!=3 && n!=4 && n!=5)
              {
                  cout<<"\n*ENTER VALID OPTION**\n";
                  goto abc;
              }
          cout << "Enter quantity: " << endl;
          cin >> qty;

          tot_item=tot_item+qty;
          amt=amt+prg[n]*qty;
          store(gro[n],qty,prg[n],amt);
          amt=0;
          //system("cls");
        }while(n!=0);

    }

  void store(string a, int quantity, int price,int amount)
  {

    finitem[x].assign(a);
    finqty[x]=quantity;
    finprice[x]=price;
    finamount[x]=amount;
    x++;

  }
};
```

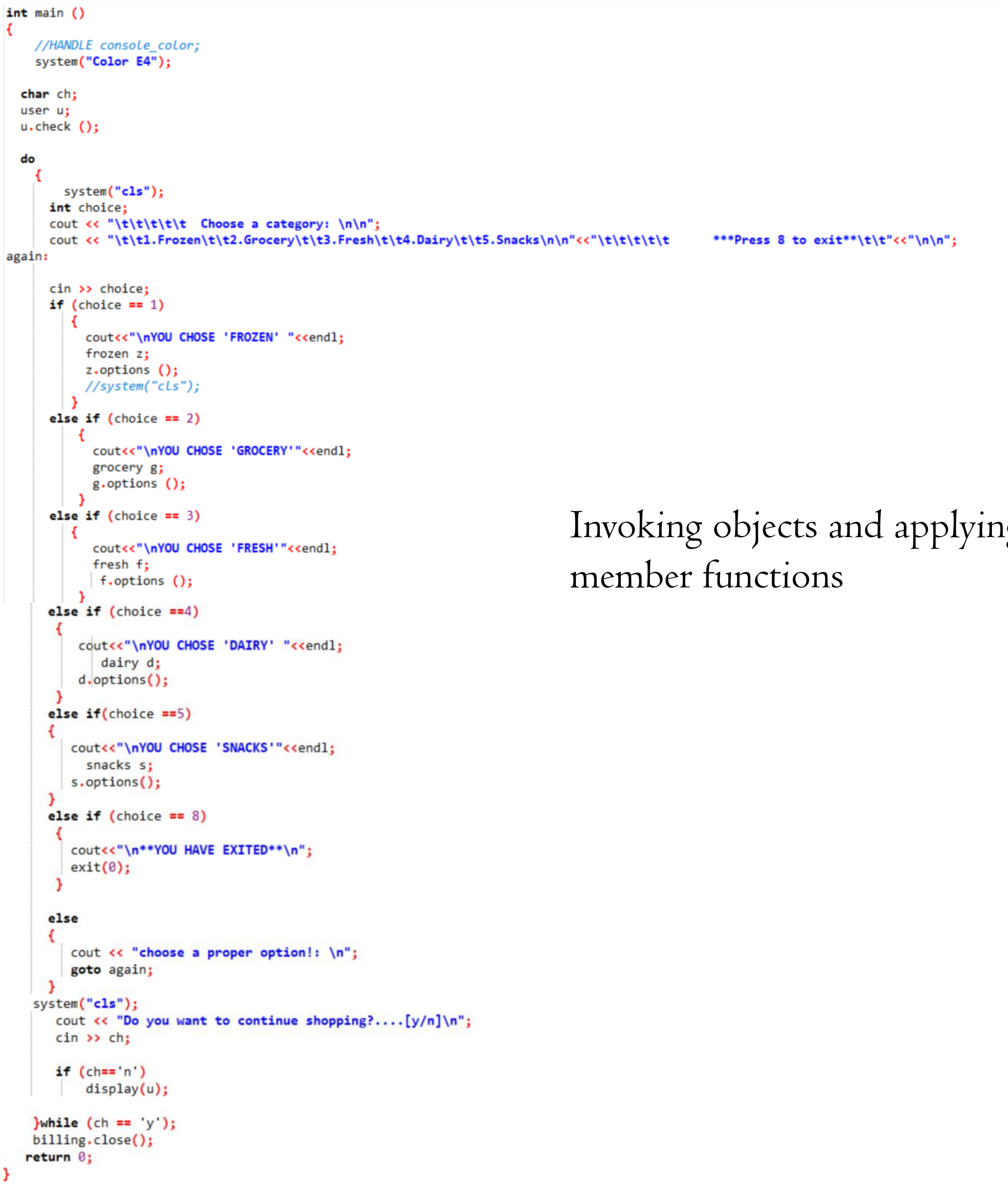


```
class dairy:public user
{

  public:

    string  gro[6]={" ","Milk      ","Curd      ","Cheese    ","Butter    ","Yogurt    "};
    int  prg[6]={0,40,50,60,45,35};

  void options ()
  {
      int n,qty,amt=0;

     do{
            cout << "\n\n[1]Milk\t\t[2]Curd\t\t[3]Cheese\t\t[4]Butter\t\t[5]Yogurt\n\n"<<endl;
            cout << "CHOOSE item number: " << endl;
            cout<<"*Enter 0 to stop adding items"<<endl;
            abc:
            cin >> n;
            if(n==0)
                break;
            else if(n!=1 && n!=2 && n!=3 && n!=4 && n!=5)
                {
                    cout<<"\n*ENTER VALID OPTION**\n";
                    goto abc;
                }
            cout << "Enter quantity: " << endl;
            cin >> qty;

            tot_item=tot_item+qty;
            amt=amt+prg[n]*qty;
            store(gro[n],qty,prg[n],amt);
            amt=0;
            //system("cls");
        }while(n!=0);

    }
  void store(string a, int quantity, int price,int amount)
  {

    finitem[x].assign(a);
    finqty[x]=quantity;
    finprice[x]=price;
    finamount[x]=amount;
    x++;

  }

};
```



```
void display(user p){
    time_t now=time(0);
    char*dt=ctime(&now);
    int pay;
    int i=0;
    int tamt=0;
    cout<<endl<<"\t\t\t\t\t\t\t       INVOICE";
    cout<<endl<<endl<<"\t\t\t\t      ************************************************************"<<endl;
    cout<<" \t\t\t\t       Item    \t"<<"   Quantity"<<"\t"<<"   Unit price"<<"      "<<"  Amount"<<endl;
     cout<<"\t\t\t\t      ************************************************************"<<endl;
    for(i=0;i<x;i++)
    {
        cout<<"\t\t\t\t\t\t"<<finitem[i]<<"\t   "<<finqty[i]<<"\t\t"<<finprice[i]<<"\t\t"<<finamount[i]<<endl;
        tamt=tamt+finamount[i];
    }
        cout<<endl<<endl<<" \t\t\t\t     Total quantity:"<<tot_item<<endl;
        cout<<endl<<" \t\t\t\t     Total amount:"<<tamt<<endl;
        cout<<" \t\t\t\t     SGST(5%): "<<0.05*tamt<<endl;
         cout<<" \t\t\t\t     CGST(5%): "<<0.05*tamt<<endl<<endl;

         cout<<"\t \t\t\t     Amount payable: "<<tamt+0.1*tamt;
    cout<<endl<<"\t\t\t\t     ************************************************************"<<endl;
    cout<<"\n\n\t\t\t\t\t\t     Amount paid: ";
    cin>>pay;
    int z;
    z=pay;
    if(pay>=(tamt+0.1*tamt))
    cout<<"\n\t\t\t\t\t\t     Amount returned: "<<pay-(tamt+0.1*tamt);
    else
    cout<<"\n\t\t\t\t\t\t     Credit amount: "<<(tamt+0.1*tamt)-pay;
    cout<<"\n\n\t\t\t\t\t\t    :)<3 THANKYOU VISIT AGAIN <3:)"<<endl;
     cout<<endl<<"\t\t\t\t     ************************************************************"<<endl;
    billing<<"\n\n\n******************BILL******************\n\n\n"<<"TIME:  "<<dt<<endl<<"Contact??:"<<p.no<<"\nTotal quantity:"<<tot_item<<
    "\nTotal amount:"<<tamt<<"\nAmount payable:"<<tamt+0.1*tamt<<"\nAmount Paid:"<<z<<"\nAmount Returned:"<<pay-(tamt+0.1*tamt)<<"\n";
}
```

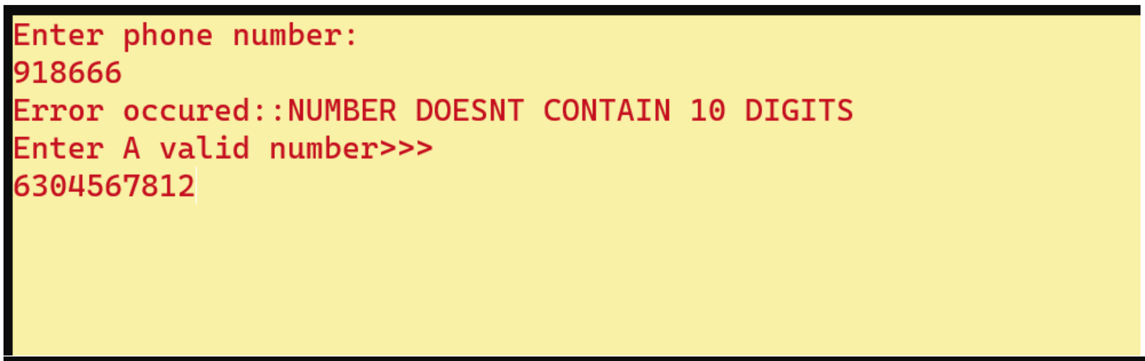
Friend function



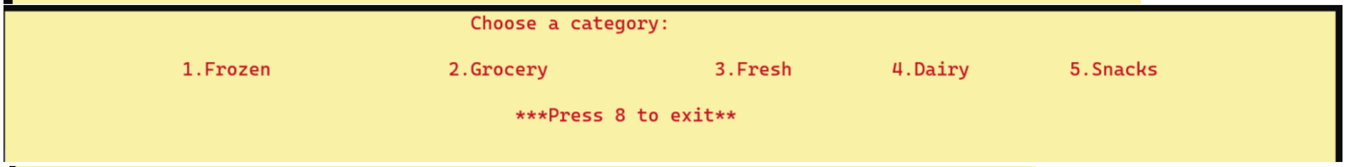
```
int main ()
{
    //HANDLE console_color;
    system("Color E4");

  char ch;
  user u;
  u.check ();

  do
    {
       system("cls");
     int choice;
     cout << "\t\t\t\t\t  Choose a category: \n\n";
     cout << "\t\t1.Frozen\t\t2.Grocery\t\t3.Fresh\t\t4.Dairy\t\t5.Snacks\n\n"<<"\t\t\t\t\t      ***Press 8 to exit**\t\t"<<"\n\n";
again:

     cin >> choice;
     if (choice == 1)
       {
         cout<<"\nYOU CHOSE 'FROZEN' "<<endl;
         frozen z;
         z.options ();
         //system("cls");
       }
     else if (choice == 2)
        {
          cout<<"\nYOU CHOSE 'GROCERY'"<<endl;
          grocery g;
          g.options ();
        }
     else if (choice == 3)
       {
          cout<<"\nYOU CHOSE 'FRESH'"<<endl;
          fresh f;
           f.options ();
        }
     else if (choice ==4)
      {
        cout<<"\nYOU CHOSE 'DAIRY' "<<endl;
           dairy d;
        d.options();
      }
     else if(choice ==5)
     {
       cout<<"\nYOU CHOSE 'SNACKS'"<<endl;
         snacks s;
       s.options();
     }
     else if (choice == 8)
      {
       cout<<"\n**YOU HAVE EXITED**\n";
       exit(0);
      }

     else
     {
       cout << "choose a proper option!: \n";
       goto again;
     }
    system("cls");
       cout << "Do you want to continue shopping?....[y/n]\n";
       cin >> ch;

       if (ch=='n')
           display(u);

    }while (ch == 'y');
    billing.close();
   return 0;
}
```

Invoking objects and applying member functions

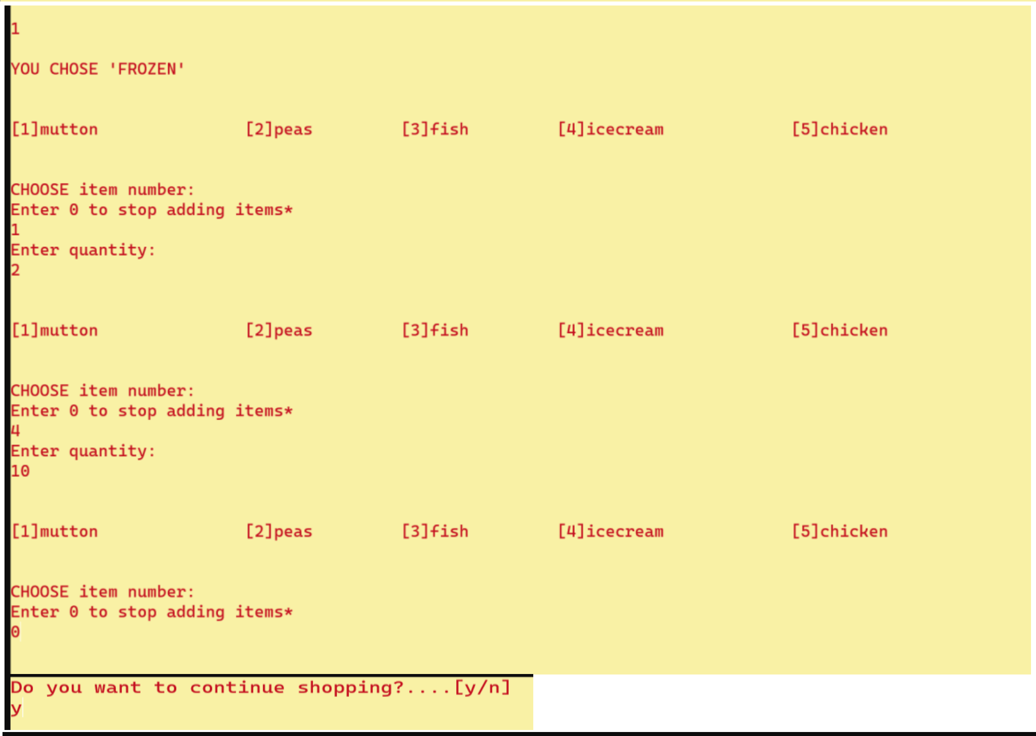
Output



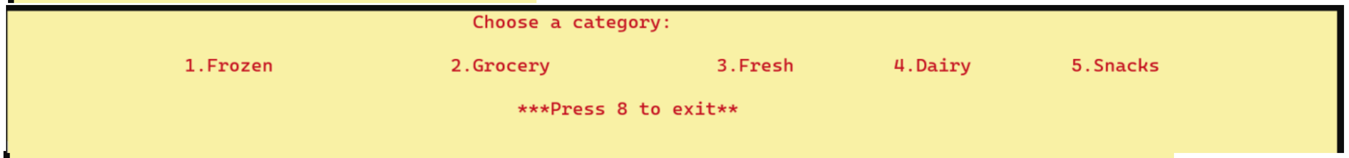
```
Enter phone number:
918666
Error occured::NUMBER DOESNT CONTAIN 10 DIGITS
Enter A valid number>>>
6304567812
```



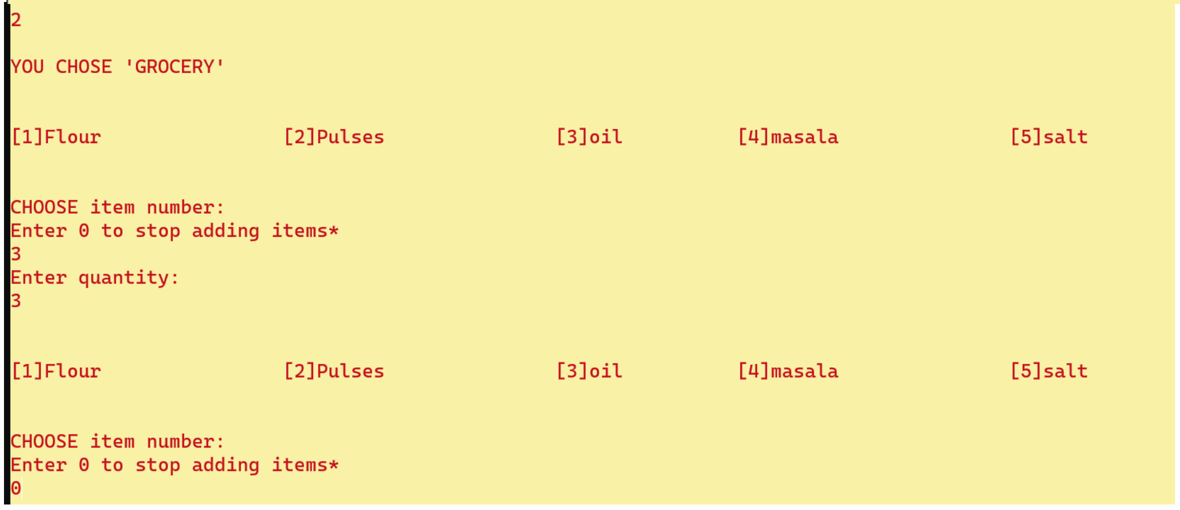
```
                                   Choose a category:

           1.Frozen                2.Grocery              3.Fresh        4.Dairy        5.Snacks

                                   ***Press 8 to exit**
```



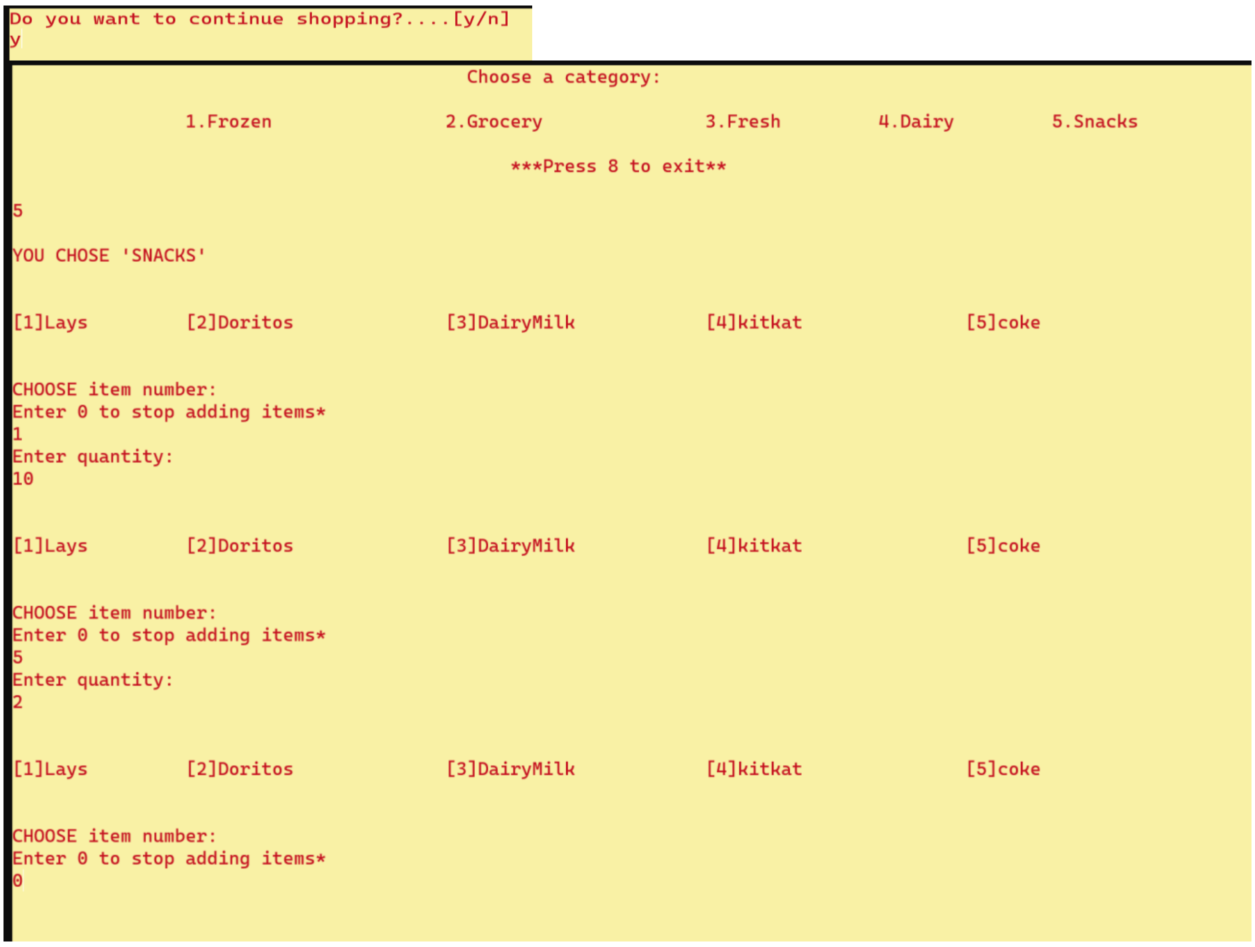
```
1

YOU CHOSE 'FROZEN'

[1]mutton               [2]peas         [3]fish         [4]icecream             [5]chicken


CHOOSE item number:
Enter 0 to stop adding items*
1
Enter quantity:
2

[1]mutton               [2]peas         [3]fish         [4]icecream             [5]chicken


CHOOSE item number:
Enter 0 to stop adding items*
4
Enter quantity:
10

[1]mutton               [2]peas         [3]fish         [4]icecream             [5]chicken


CHOOSE item number:
Enter 0 to stop adding items*
0

Do you want to continue shopping?....[y/n]
y
```



```
                                   Choose a category:

           1.Frozen                2.Grocery              3.Fresh        4.Dairy        5.Snacks

                                   ***Press 8 to exit**
```

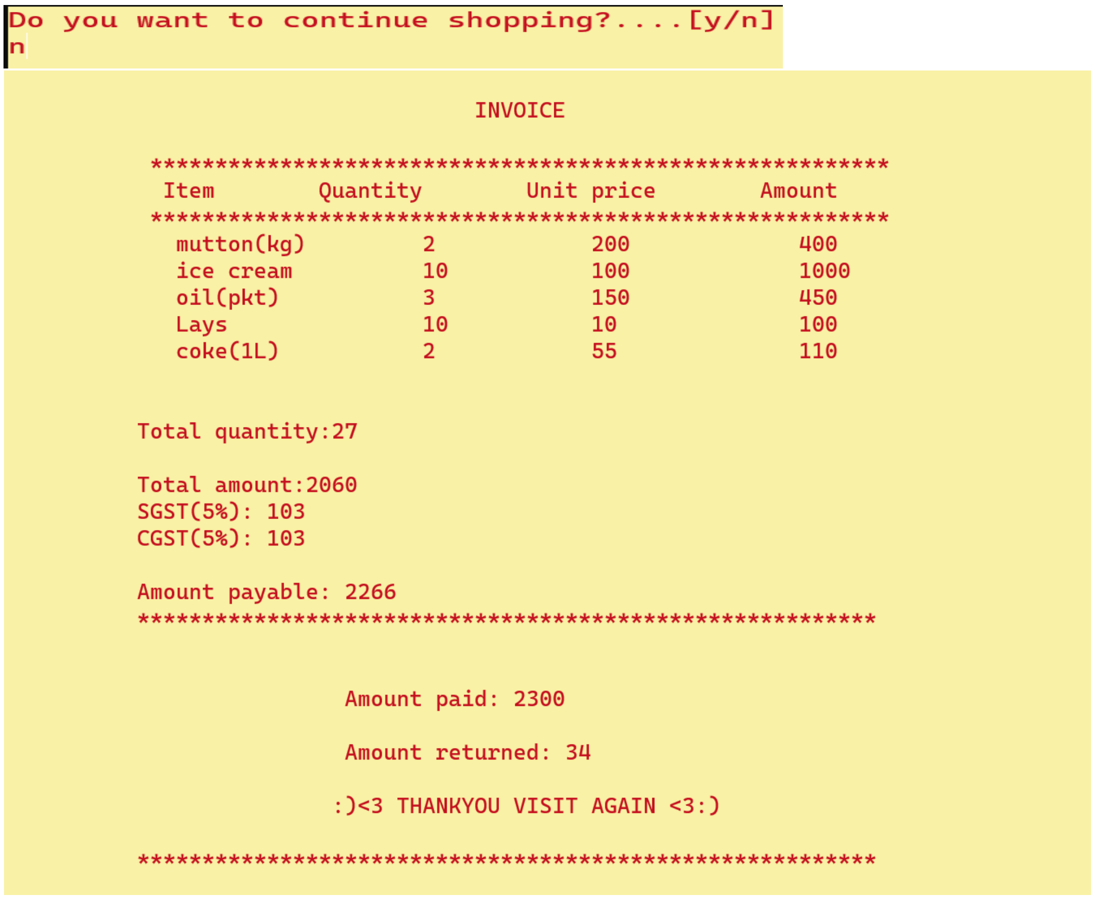


```
2

YOU CHOSE 'GROCERY'

[1]Flour                [2]Pulses               [3]oil          [4]masala               [5]salt


CHOOSE item number:
Enter 0 to stop adding items*
3
Enter quantity:
3

[1]Flour                [2]Pulses               [3]oil          [4]masala               [5]salt


CHOOSE item number:
Enter 0 to stop adding items*
0
```



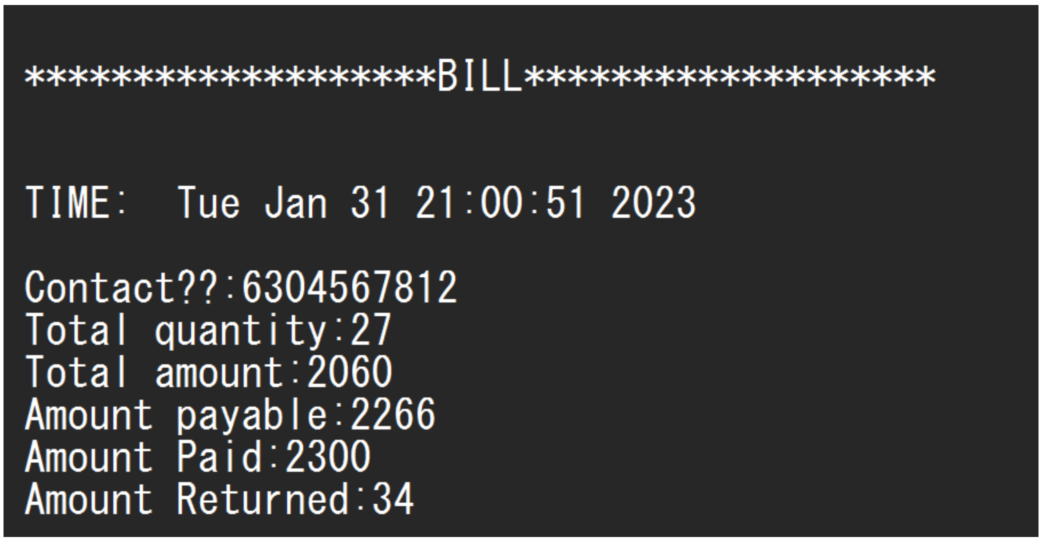
```
Do you want to continue shopping?....[y/n]
y
                                          Choose a category:

                1.Frozen                2.Grocery               3.Fresh         4.Dairy         5.Snacks

                                              ***Press 8 to exit**

5

YOU CHOSE 'SNACKS'


[1]Lays         [2]Doritos              [3]DairyMilk            [4]kitkat               [5]coke


CHOOSE item number:
Enter 0 to stop adding items*
1
Enter quantity:
10


[1]Lays         [2]Doritos              [3]DairyMilk            [4]kitkat               [5]coke


CHOOSE item number:
Enter 0 to stop adding items*
5
Enter quantity:
2


[1]Lays         [2]Doritos              [3]DairyMilk            [4]kitkat               [5]coke


CHOOSE item number:
Enter 0 to stop adding items*
0
```

Invoice generation



```
Do you want to continue shopping?....[y/n]
n

                                    INVOICE

          *********************************************************
           Item          Quantity        Unit price         Amount
          *********************************************************
            mutton(kg)          2            200               400
            ice cream           10           100               1000
            oil(pkt)            3            150               450
            Lays                10           10                100
            coke(1L)            2            55                110


         Total quantity:27

         Total amount:2060
         SGST(5%): 103
         CGST(5%): 103

         Amount payable: 2266
         *********************************************************


                         Amount paid: 2300

                         Amount returned: 34

                        :)<3 THANKYOU VISIT AGAIN <3:)

         *********************************************************
```

Bill stored in the file



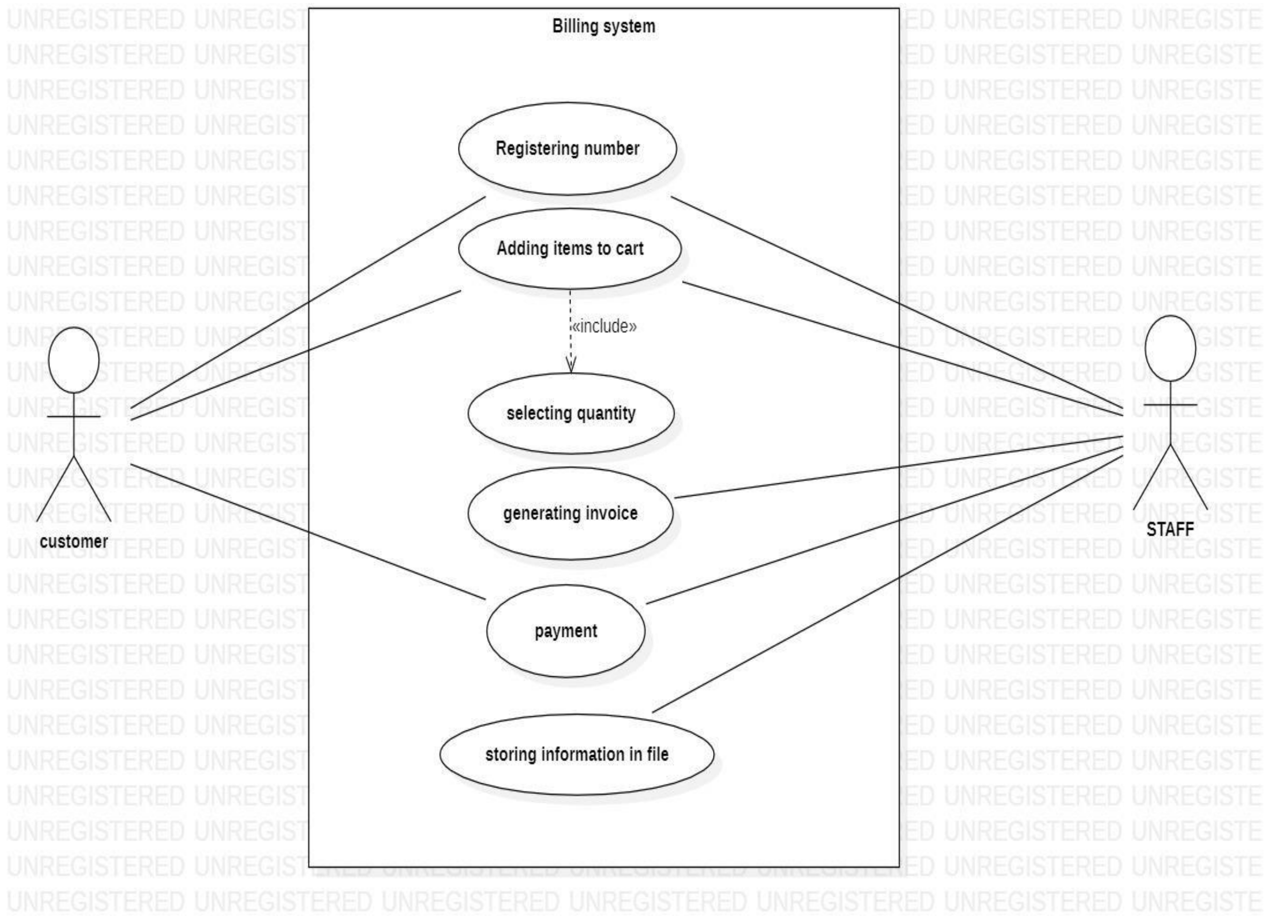
```
******************BILL******************


TIME:  Tue Jan 31 21:00:51 2023

Contact??:6304567812
Total quantity:27
Total amount:2060
Amount payable:2266
Amount Paid:2300
Amount Returned:34
```

CONCLUSION:

Object-oriented programming revolves around the concepts of objects and classes. In C++, the classes are referred as templates for the objects while the objects are instances of a class so, the objects can inherit all the characteristics, variables, and functions of the class. Object-Oriented Programming (OOP) uses "objects" to model real world objects. Object-Oriented Programming (OOP) consist of some important concepts namely Encapsulation, Polymorphism, Inheritance and Abstraction. These features are generally used to create multipurpose projects.

BILLING MANAGEMENT SOFTWARE

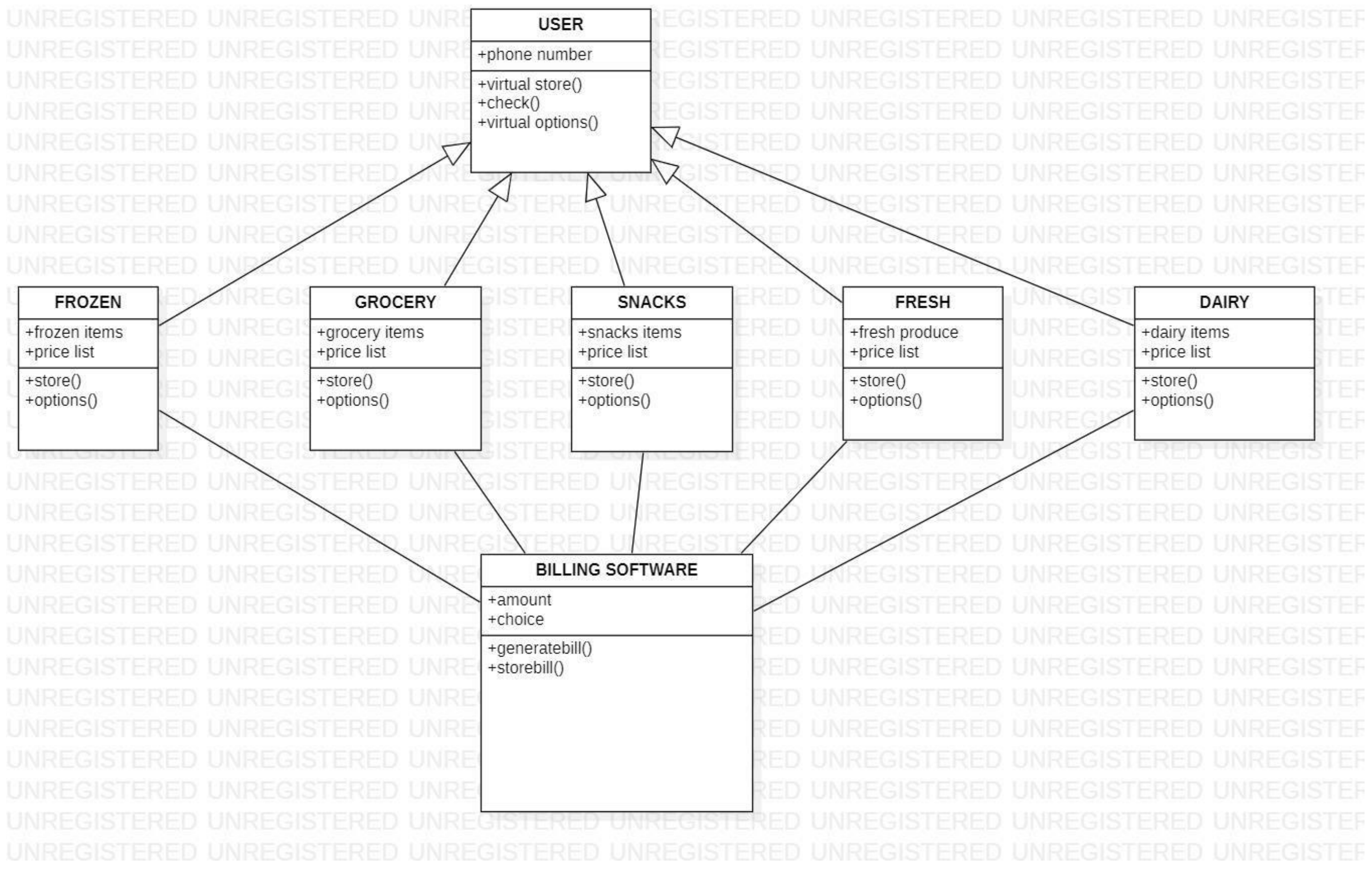
USE CASE DIAGRAM



Description:

Here we have taken a use case subject named billing system, and 2 actors: customer and staff. The oval shapes in this diagram are the use cases. The use case subject here is billing system. Here, the selecting quantity use case is an included use case of adding items to cart. Because without selecting quantity ,we can't add items to cart. These thick lines in the use case diagram represent association relationship between the actor and the use case.

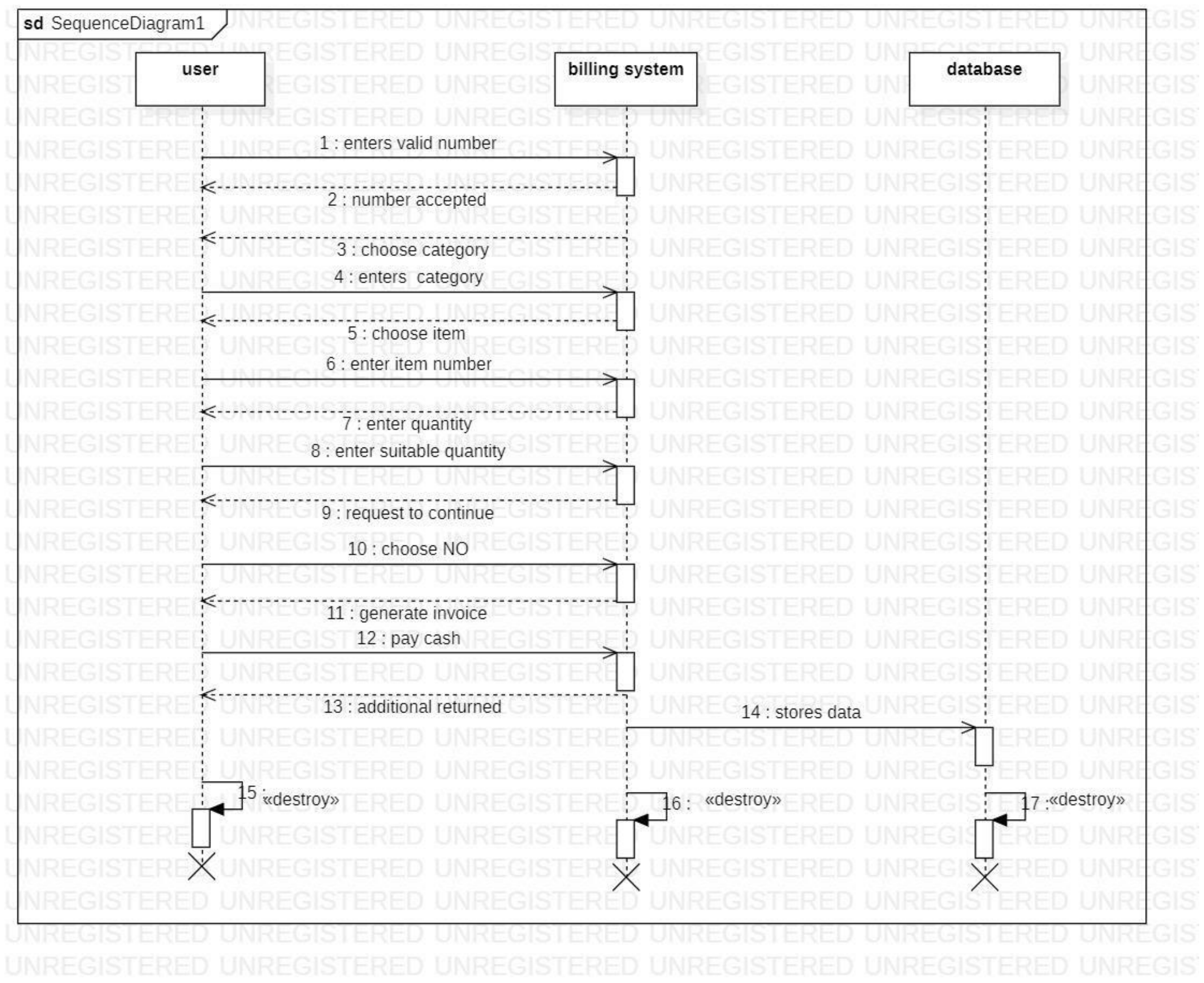
CLASS DIAGRAM



Description:

Here we have used generalization concept in above the class diagram. The 5 classes here in the diagram, frozen,grocery,snacks,fresh,dairy,are all inherited from the user class.To represent the generalization property, we use a thick line and triangular tip pointing towards the base class(user).After selecting all the items,billing software gathers all this data and generates the bill and stores the bill for future needs. Association(thick line representation) used as a relationship between 2 classes.

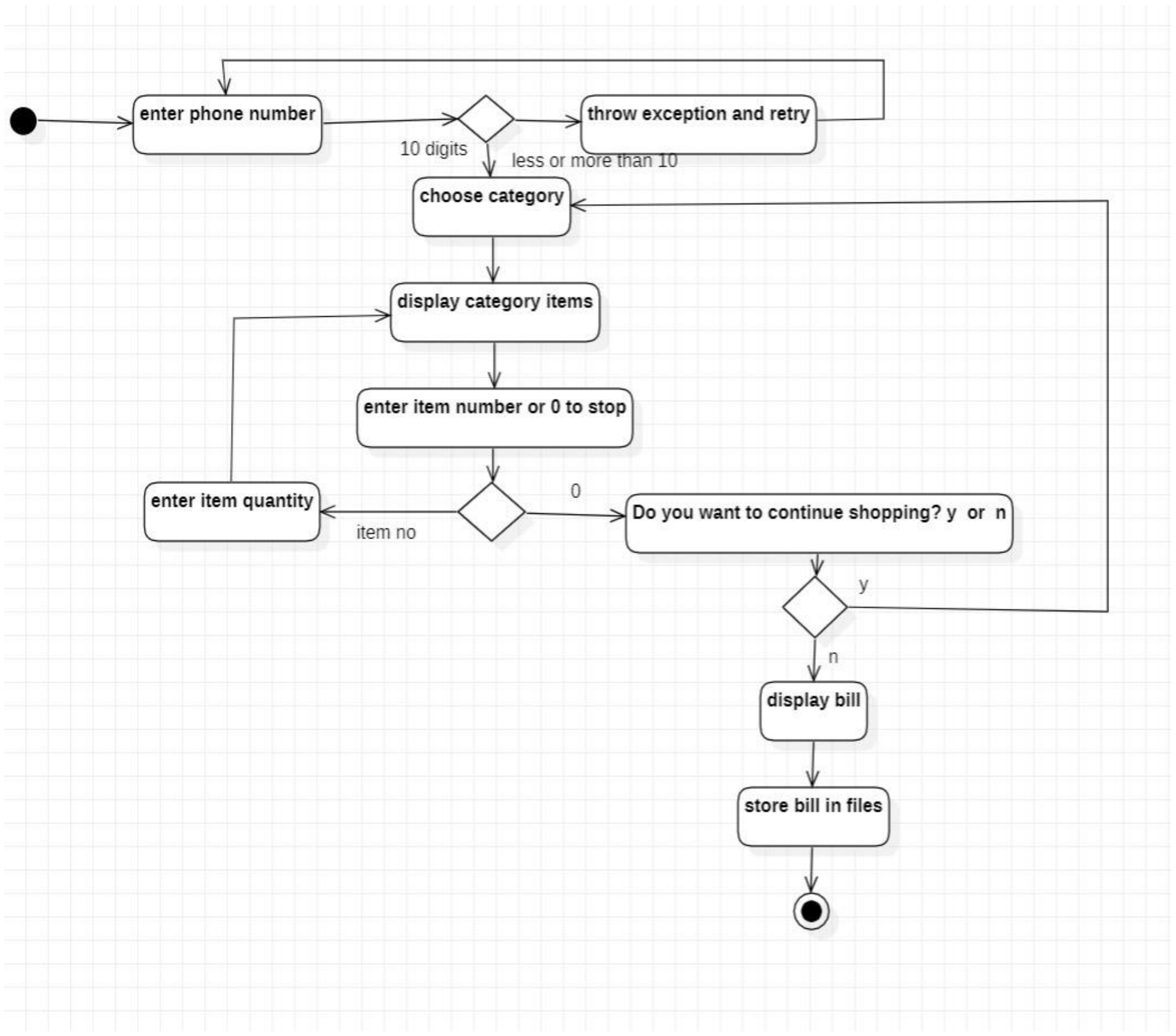
SEQUENCE DIAGRAM



Description:

The above sequence diagram contains 3 objects(also called as lifelines-user, billing system,database).And the arrowed lines represent messages between 2 objects. The vertical dotted line represents the time
axis,the sequence of messages that happen over a period of time.

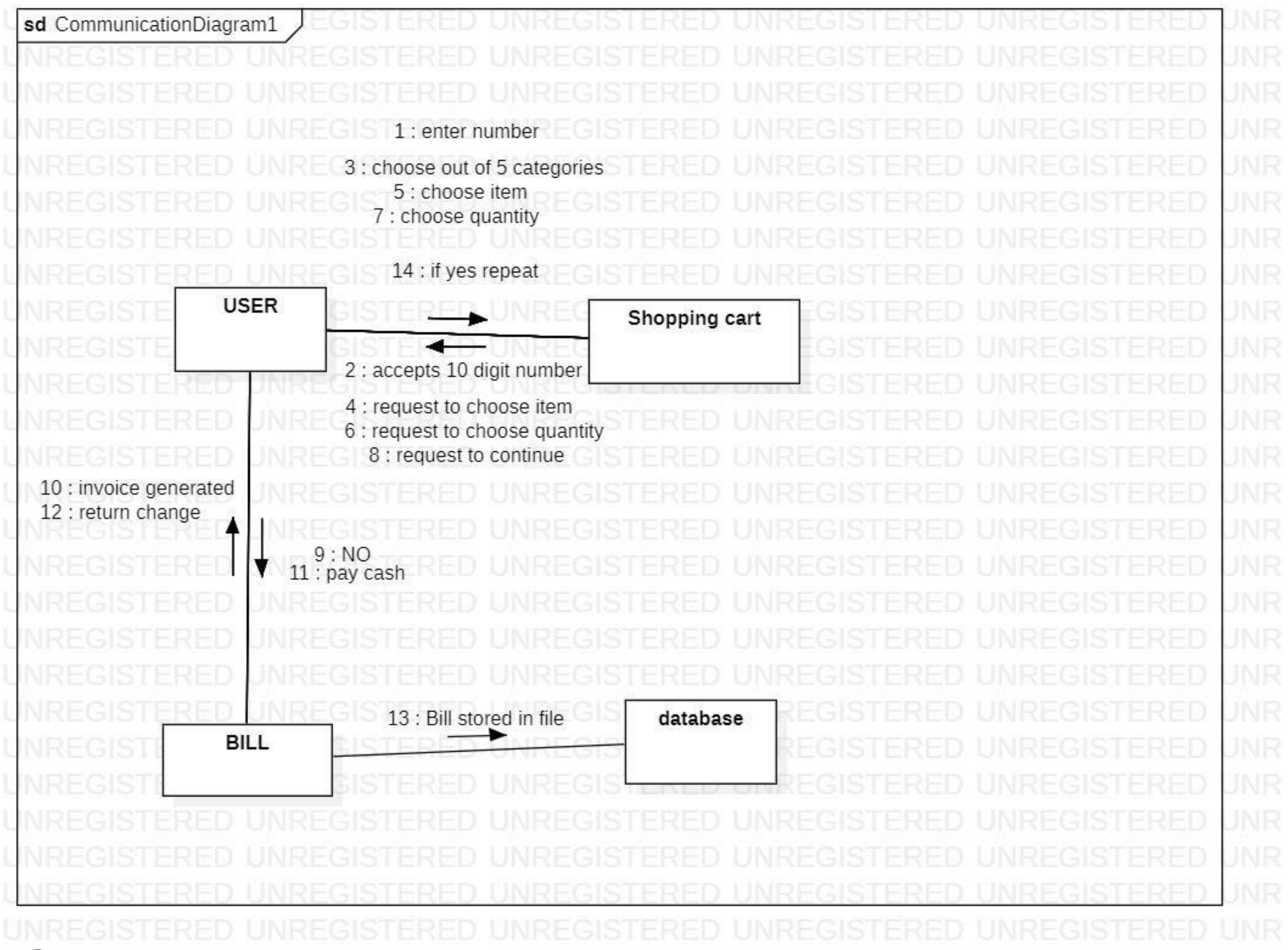
ACTIVITY DIAGRAM



Description:

Above is an activity diagram representing the flow of actions in the billing management software systems. The initialization symbol(initial) is represented by a filled circle. And the final symbol(final) is represented by a filled circle inside a circle. The actions are represented by the rectangular boxes and decision statements are represented by a rhombus shape,(it has two lines evolving from it-true or false)and arrowed lines represented the action flow .Here in activity diagram , the advantage is that we can handle the exception handling statements which come across in our use case, which will give us more insights .

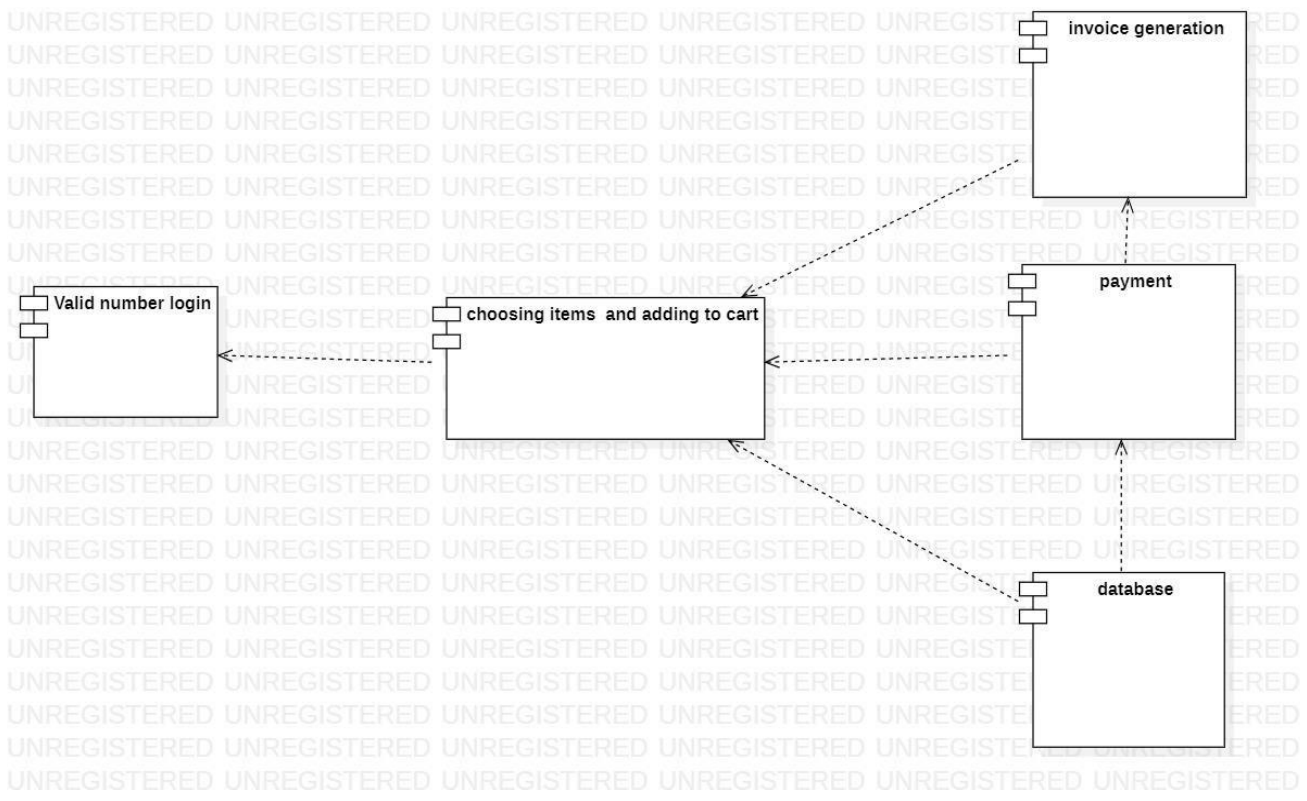
COMMUNICATION/COLLABORATION DIAGRAM



Description:

In communication diagrams, we have lifelines represent the objects that participate in an interaction. And we have a message pathway called as a connector that identifies objects and passes messages in interaction. As we can see there are connectors between 2 objects and messages are transferred only between those 2 objects, communication is on the whole, but it is important to understand inter-communication between objects ,especially in this use case..(understanding the relation b/w user-bill, user- shopping cart,bill-database)

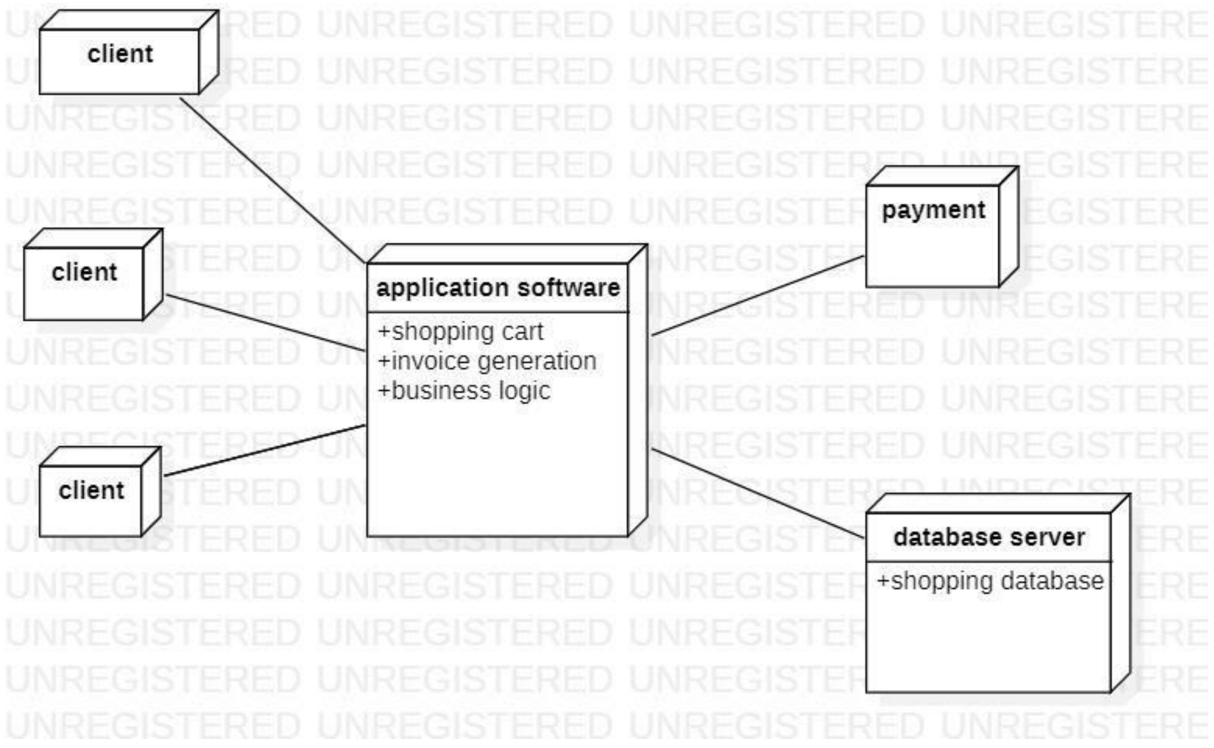
COMPONENT DIAGRAM



Description:

We have divided our-use case into components (rectangular boxes) and used dependency relationship here.

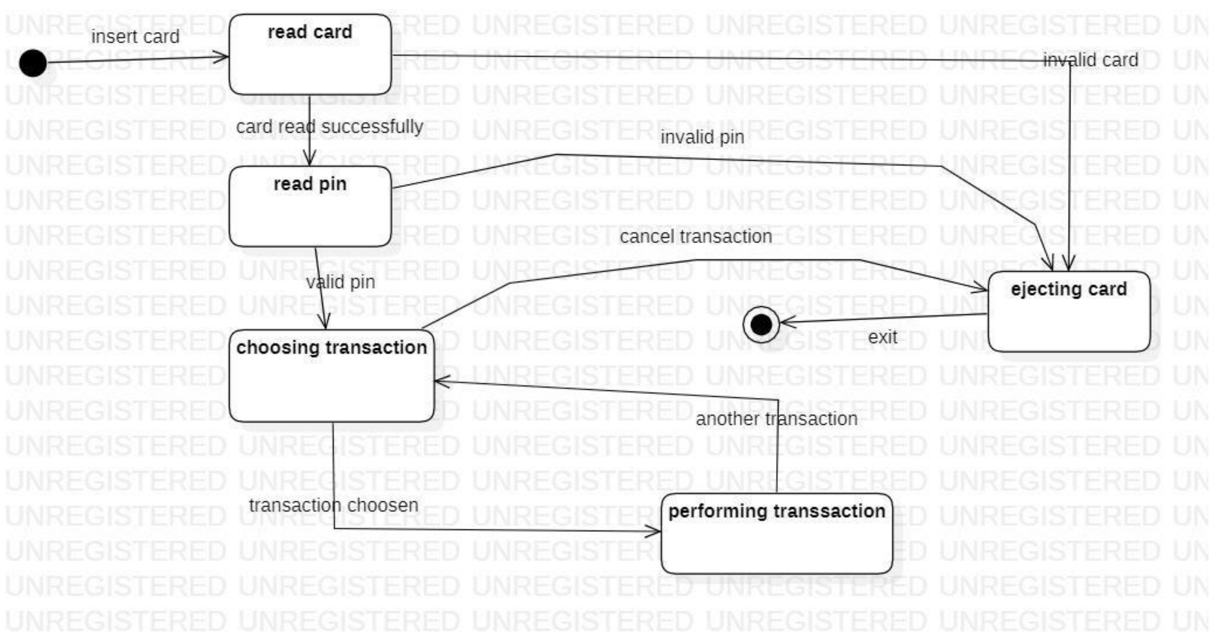
DEPLOYMENT DIAGRAM



Description:

The deployment diagram of our use case consists of nodes and communication between the nodes using relationships(basically to explain the client-database connections)

STATECHART DIAGRAM



Description:

Here's the statechart diagram of our use case.We have represented the actions and the transitions between 2 objects.

References:

www.google.com
www.geeksforgeeks.com
www.javatpoint.com

★ ★