**B.M.S College of Engineering**

**P.O. Box No.: 1908 Bull Temple Road,**

**Bangalore-560 019**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING **

**Course –Object oriented programming using C++**

**Course Code –18IS3PCOOP**

**AY 2021-22**

**Title:**

**Store Management System**

Submitted to – Dr V Shubha Rao

Submitted by -

Khushi Agrawal-1BM20IS066

Vignesh JR- 1BM20IS181

**B.M.S College of Engineering**

**P.O. Box No.: 1908 Bull Temple Road,**

**Bangalore-560 019**

INDEX

|  |  |  |
| --- | --- | --- |
| **Sl. no.** | **Contents** | **Page**  **no.** |
| **1** | **Abstract** | **3** |
| **2** | **Introduction** | **4** |
| **3** | **Oops concept used and explained** | **5-7** |
| **4** | **Software used** | **7** |
| **5** | **Code** | **8-22** |
| **6** | **Snapshots** | **23-28** |
| **7** | **Reference** | **29** |

**ABSTRACT**

Managing a Store isn’t as easy as taking candy from a baby, it requires a proper functioning system with zero errors or bugs to hold a proper, well managed Database. With the help of concepts of object oriented programming in c++ it is possible to create a system that stores all the input given by the store staff about the items present in the store.

**INTRODUCTION**

A Store Management system basically allows a user to perform operations like

* Insertion of data(different category of items sold, their price, quality description and availability).
* Display the data(Everything that is present at the store category-wise).
* Search for an item(Among all the items present,searching for a particular product with the help of its code).

● Deletion of record(Updating the item count at the end of the day baked on sales).

Switch Case is used to ask the user to choose whatever he/she wants to do.

**OOPS CONCEPTS USED AND EXPLAINED**

**Classes and objects:**

A class in C++ is the building block that leads to Object-Oriented programming. 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 C++ class is like a blueprint for an object. 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.

The concepts of classes and objects are used in this project. The project contains a class called “Store”, which consists of various functions and variables necessary to implement the product registration. Various functions are defined and declared to perform operations like insertion ,traversal,deletion and search.

**Constructors:**

A constructor is a special type of member function of a class which initializes objects of a class. In C++, Constructor is automatically called when an object(instance of class) is created. It is a special member function of the class because it does not have any return type.

Default constructor is used to initialize the member variables to 0.

**Operator Overloading:**

In C++, we can make operators to work for user defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading. For example, we can overload an operator ‘+’ in a class like String so that we can concatenate two strings by just using +. Other example classes where arithmetic operators may be overloaded are Complex Number, Fractional Number, Big Integer, etc.

Here operator ‘+’ is overloaded to add price of different objects.

**Dynamic memory allocation:**

Often some situation arises in programming where data or input is dynamic in nature, i.e. the number of data items keeps changing during program execution. A live scenario where the program is developed to process lists of students of an organization. The list grows as the names are added and shrink as the names get deleted. With the increase in name the memory allocates space to the list to accommodate additional data items. Such situations in programming require dynamic memory management techniques.

**Encapsulation:**

This is a programming style where implementation details are hidden. It reduces software development complexity greatly. With Encapsulation, only methods are exposed. The programmer does not have to worry about implementation details but is only concerned with the operations. For example, if a developer wants to use a dynamic link library to display date and time, he does not have to worry about the codes in the date and time class; rather he would simply use the data and time class by using public variables to call it up. In essence encapsulation is achieved in Python and Java by creating Private variables to define hidden classes in and then using public variables to call them up for use. With this approach, a class can be updated or maintained without worrying about the methods using them. If you are calling up a class in ten methods and you need to make changes, you don’t have to update the entire methods, rather you update a single class. Once the class is changed, it automatically updates the methods accordingly. Encapsulation also ensures that your data is hidden from external modification. Encapsulation is also known as Data-Hidden.

**Inheritance:**

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class.

The idea of inheritance implements a relationship. For example, mammal is an animal, dog is a mammal hence dog is an animal as well and so on.

We have “Store” as our base class and “cloth”, “dusing”, “material” are classes derived from it.

**Exceptions:**

An exception is **a problem that arises during the execution of a program**. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. Exceptions provide a way to transfer control from one part of a program to another.

**Files:**

Files are **used to store data in a storage device permanently**. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it. A stream is an abstraction that represents a device on which operations of input and output are performed.

**Modular Programming:**

Modular programming is the process of subdividing a computer program into separate sub-programs. A module is a separate software component. It can often be used in a variety of applications and functions with other components of the system.

* Some programs might have thousands or millions of lines and to manage such programs it becomes quite difficult as there might be too many of syntax errors or logical errors present in the program, so to manage such type of programs concept of **modular** **programming** approached.
* Each sub-module contains something necessary to execute only one aspect of the desired functionality.
* Modular programming emphasis on breaking of large programs into small problems to increase the maintainability, readability of the code and to make the program handy to make any changes in future or to correct the errors.

**Points which should be taken care of prior to modular program development:**

1. Limitations of each and every module should be decided.
2. In which way a program is to be partitioned into different modules.
3. Communication among different modules of the code for proper execution of the entire program.

**SOFTWARES USED:**

Visual Studio Code - Code Editor;

Windows/Linux/iOS - Operating System;

**CODE:**

**//project : STORE MANAGEMENT//**

**#include<iostream>**

**#include<conio.h>**

**#include<string.h>**

**using namespace std;**

**class store**

**{**

**protected:**

**char name[20][20];**

**float price[20];**

**int sum,code[20];**

**public:**

**//Default constructor**

**store()**

**{**

**sum=0;**

**temp=0;**

**}**

**char category[20];**

**int temp;**

**void insert();**

**void add();**

**void display();**

**void search();**

**void remove();**

**void total\_amount();**

**void catego\_amount();**

**void error1();**

**void initial\_task();**

**int function1();**

**};**

**void store::insert()**

**{**

**int i,n,k;**

**cout<<"Enter the category:";**

**cin>>category;**

**cout<<"How many items do you wish to enter:";**

**cin>>n;**

**for(i=temp;i<n;i++)**

**{**

**cout<<"Enter the name of item=";**

**cin>>name[temp];**

**cout<<"Enter its code=";**

**cin>>code[temp];**

**cout<<"Enter the price=";**

**cin>>price[temp];**

**temp++;**

**sum=sum+price[i];**

**}**

**}**

**void store::add()**

**{**

**int i,n;**

**cout<<"Enter the name of item=";**

**cin>>category;**

**cout<<"How many items do you wish to enter:";**

**cin>>n;**

**for(i=temp;i<n;i++)**

**{**

**cout<<"Enter the name of item=";**

**cin>>name[temp];**

**cout<<"Enter its code=";**

**cin>>code[temp];**

**cout<<"Enter the price=";**

**cin>>price[temp];**

**temp++;**

**sum=sum+price[i];**

**}**

**}**

**void store::display()**

**{**

**int i;**

**cout<<"under the "<<category<<" catagory the items are:"<<'\n'<<'\n';**

**cout<<"Item name"<<'\t';**

**cout<<"Code"<<'\t';**

**cout<<"Price"<<'\t'<<'\n';**

**for(i=0;i<temp;i++)**

**{**

**cout<<name[i]<<'\t'<<'\t';**

**cout<<code[i]<<'\t';**

**cout<<price[i]<<'\n'<<'\n';**

**}**

**cout<<'\n'<<'\n';**

**}**

**void store::search()**

**{**

**int i,m;**

**cout<<"Enter the code:";**

**cin>>m;**

**for(i=0;i<temp;i++)**

**{**

**if(code[i]==m)**

**{**

**cout<<"Item name is =" <<name[i]<<'\n';**

**cout<<"Price is="<<price[i]<<'\n'<<'\n';**

**}**

**else**

**cout<<"Wrong Code entereed! TRY AGAIN:) "<<'\n'<<'\n';**

**}**

**cout<<'\n'<<'\n';**

**}**

**void store::remove()**

**{**

**int i,m;**

**cout<<"Enter the code:";**

**cin>>m;**

**for(i=0;i<temp;i++)**

**{**

**if(code[i]==m)**

**{**

**price[i]=0;**

**}**

**else**

**cout<<"Wrong Code entereed! TRY AGAIN:)"<<'\n'<<'\n';**

**}**

**cout<<'\n'<<'\n';**

**}**

**void store::total\_amount()**

**{**

**cout<<"The total price of all the items in all categories is="<<sum<<'\n'<<'\n';**

**}**

**void store:: catego\_amount()**

**{**

**cout<<"under the "<<category<<" catagory the total price of the items are="<<sum<<'\n'<<'\n';**

**}**

**void store::error1()**

**{**

**cout<<"Insert atleast one item.";**

**cout<<'\n'<<'\n';**

**}**

**void store::initial\_task()**

**{**

**cout<<"Choose one of the following options"<<'\n';**

**cout<<"1.To INSERT press 1"<<'\n';**

**cout<<"2.To ADD press 2"<<'\n';**

**cout<<"3.To DISPLAY press 3"<<'\n';**

**cout<<"4.To SEARCH press 4"<<'\n';**

**cout<<"5.To REMOVE press 5"<<'\n';**

**cout<<"6.To see the TOTAL PRICE press 6"<<'\n';**

**cout<<"7.To EXIT press 0"<<'\n';**

**}**

**int store::function1()**

**{**

**int a1;**

**cout<<"choose an option to enter into the following available options"<<'\n';**

**cout<<"1.Press 1 for CLOTHES."<<'\n'<<"2.Press 2 for DAILY usage stuff."<<'\n'**

**<<"3.Press 3 for RAW MATERIALS."<<'\n'<<"4.Press 0 for EXIT."<<'\n';**

**cin>>a1;**

**return a1;**

**}**

**class cloth:public store**

**{**

**private:**

**char clt[15],cr[15];**

**public:**

**cloth operator+(cloth);**

**void insert\_cl();**

**void display\_cl();**

**};**

**cloth cloth::operator+(cloth c)**

**{**

**cloth ob1;**

**ob1.sum=sum+c.sum;**

**return (ob1);**

**}**

**void cloth::insert\_cl()**

**{**

**cout<<"what type of cloth is this ?(cotton,polyester,rayon and so on)"<<'\n';**

**cin>>clt;**

**cout<<"What's the color of the cloth ?"<<'\n';**

**cin>>cr;**

**cout<<'\n'<<'\n'<<'\n';**

**}**

**void cloth::display\_cl()**

**{**

**cout<<"Type :"<<clt<<'\n';**

**cout<<"Color :"<<cr<<'\n'<<'\n'<<'\n';**

**}**

**class dusing:public store**

**{**

**private:**

**char dus[15];**

**public:**

**dusing operator+(dusing);**

**void insert\_du();**

**void display\_du();**

**};**

**dusing dusing::operator+(dusing c)**

**{**

**dusing ob2;**

**ob2.sum=sum+c.sum;**

**return (ob2);**

**}**

**void dusing::insert\_du()**

**{**

**cout<<"what type of daily using thing is this ?(Cosmetics,Grocery and so on)"<<'\n';**

**cin>>dus;**

**cout<<'\n'<<'\n'<<'\n';**

**}**

**void dusing::display\_du()**

**{**

**cout<<"Type:" <<dus<<'\n'<<'\n'<<'\n';**

**}**

**class material:public store**

**{**

**private:**

**int sp;**

**char mat[14],qu[15];**

**public:**

**material operator+(material c);**

**void insert\_ma();**

**void display\_ma();**

**};**

**material material::operator+(material c)**

**{**

**material ob3;**

**ob3.sum=sum+c.sum;**

**return (ob3);**

**}**

**void material::insert\_ma()**

**{**

**cout<<"What type of raw material is this ?(Vegetable,Pulses?)"<<'\n';**

**cin>>mat;**

**cout<<"Quality?(super fresh,fresh,old)"<<'\n';**

**cin>>qu;**

**cout<<"Best before?"<<'\n';**

**cin>>sp;**

**cout<<'\n'<<'\n'<<'\n';**

**}**

**void material::display\_ma()**

**{**

**cout<<"Type:" <<mat<<'\n';**

**cout<<"Quality:" <<qu<<'\n';**

**cout<<"Best Before:" <<sp<<'\n'<<'\n'<<'\n';**

**}**

#include "first.h"

#include<fstream>

int main()

 {

     fstream file;

     char ch;

    cloth t[20],f1,ob;

    dusing du[20],f2,ob1;

    material ma[20],f3,ob2;

    int i,x=0,n,a,j,k,l,s,m,b,c,q,g=0,a1,y=0,z=0;

    char inp[20];

    string name="welcome.txt";

    file.open(name.c\_str());

    while(file.eof()==0)

    {

      file.get(ch);

      cout<<ch;

    }

    file.close();

    //clrscr();

    for(i=0;;i++)

    {

    a1=ob.function1();

    switch(a1)

    {

    case 1:

    for(i=0;;i++)

     {

       ob.initial\_task();

       cin>>a;

       switch(a)

       {

     case 1:

        try

        {

            if(x<=0)

         {

           cout<<"How many Categories do you want to Insert=";

           cin>>n;

           for(i=0;i<n;i++)

            {

              t[x].insert();

              t[x].insert\_cl();

               x++;

             }

          }

          else

          throw (-1);

        }

        catch(int x)

        {

            cout<<"Can Not Insert,Choose option ADD.";

        }

        if(x<=0)

          {

        //    cout<<"YOU CAN NOT INSERT ITEM MORE THAN ONCE"<<'\n'<<'\n';

        //    break;

        //   }

        //  else

        //  {

           cout<<"How many Categories do you want to Insert=";

           cin>>n;

           for(i=0;i<n;i++)

            {

              t[x].insert();

              t[x].insert\_cl();

               x++;

             }

          }

         break;

      case 2:

        if(x<=0)

        {ob.error1();

        break;}

         else

         {

           cout<<"How many categories do you want to Add=";

           cin>>n;

           for(j=0;j<n;j++)

            {

             t[x].add();

             t[x].insert\_cl();

              x++;

             }

          break;

          }

       case 3:

        if(x<=0)

        {ob.error1();

        break;}

         else

         {

          for(s=0;s<x;s++)

           {

             t[s].display();

             t[s].display\_cl();

           }

           break;

         }

       case 4:

        if(x<=0)

        {ob.error1();

        break;}

        if(x>0)

        {

         cout<<"Enter your Desired Category=";

         cin>>inp;

         for(c=0;c<x;c++)

         {

           b=strcmp(inp,t[c].category);

           if(b==0)

           break;

          }

          if(b==0)

            {t[c].search();

            t[c].display\_cl();}

           break;

          }

        case 5:

        if(x<=0)

        {ob.error1();

          break;}

        else

        {

         cout<<"Enter your Desired Category=";

         cin>>inp;

         for(c=0;c<x;c++)

         {

           b=strcmp(t[c].category,inp);

           if(b==0)

           break;

          }

          if(b==0)

            t[c].remove();

           break;

          }

         case 6:

        if(x<=0)

         {ob.error1();

          break;}

        else

        {

          for(z=g;z<x;z++)

          {

            f1=f1+t[z];

            g++;

          }

        for(i=0;i<x;i++)

        t[i].catego\_amount();

        f1.total\_amount();

        break;

        }

     }

     if(a==0)

     break;

       }

       break;

       case 2:

    for(i=0;;i++)

     {

       ob1.initial\_task();

       cin>>a;

       switch(a)

       {

     case 1:

        if(y>0)

         {

           cout<<"YOU CAN NOT INSERT ITEM MORE THAN ONES"<<'\n'<<'\n';

           break;

          }

         else

         {

           cout<<"How many categories do you want to Insert=";

           cin>>n;

           for(i=0;i<n;i++)

            {

              du[y].insert();

              du[y].insert\_du();

               y++;

             }

          }

         break;

      case 2:

        if(y<=0)

        {ob1.error1();

        break;}

         else

         {

           cout<<"How many categories do you want to Add=";

           cin>>n;

           for(j=0;j<n;j++)

            {

             du[y].add();

             du[y].insert\_du();

              y++;

             }

          break;

          }

       case 3:

        if(y<=0)

        {ob1.error1();

        break;}

         else

         {

          for(s=0;s<y;s++)

           {

             du[s].display();

             du[s].display\_du();

           }

           break;

         }

       case 4:

        if(y<=0)

        {ob.error1();

        break;}

        if(y>0)

        {

         cout<<"Enter your Desired Category=";

         cin>>inp;

         for(c=0;c<y;c++)

         {

           b=strcmp(inp,du[c].category);

           if(b==0)

           break;

          }

          if(b==0)

           {

            du[c].search();

            du[c].display\_du();

           }

           break;

          }

        case 5:

        if(y<=0)

        {ob1.error1();

          break;}

        else

        {

         cout<<"Enter your Desired Category=";

         cin>>inp;

         for(c=0;c<y;c++)

         {

           b=strcmp(du[c].category,inp);

           if(b==0)

           break;

          }

          if(b==0)

            du[c].remove();

           break;

          }

         case 6:

        if(y<=0)

         {ob.error1();

          break;}

        else

        {

          for(z=g;z<x;z++)

          {

            f2=f2+du[z];

            g++;

          }

        for(i=0;i<y;i++)

        du[i].catego\_amount();

        f2.total\_amount();

        break;

        }

     }

     if(a==0)

     break;

       }

       break;

       case 3:

    for(i=0;;i++)

     {

       ob2.initial\_task();

       cin>>a;

       switch(a)

       {

     case 1:

        if(z>0)

         {

           cout<<"YOU CAN NOT INSERT ITEM MORE THAN ONCE"<<'\n'<<'\n';

           break;

          }

         else

         {

           cout<<"How many Category do you want to Insert=";

           cin>>n;

           for(i=0;i<n;i++)

            {

              ma[z].insert();

              ma[z].insert\_ma();

               z++;

             }

          }

         break;

      case 2:

        if(z<=0)

        {ob2.error1();

        break;}

         else

         {

           cout<<"How many categories do you want to Add=";

           cin>>n;

           for(j=0;j<n;j++)

            {

             ma[z].add();

             ma[z].insert\_ma();

              z++;

             }

          break;

          }

       case 3:

        if(z<=0)

        {ob2.error1();

        break;}

         else

         {

          for(s=0;s<z;s++)

           {

             ma[s].display();

             ma[s].display\_ma();

           }

           break;

         }

       case 4:

        if(z<=0)

        {ob2.error1();

        break;}

        if(z>0)

        {

         cout<<"Enter your Desired category=";

         cin>>inp;

         for(c=0;c<z;c++)

         {

           b=strcmp(inp,ma[c].category);

           if(b==0)

           break;

          }

          if(b==0)

            {ma[c].search();

            ma[s].display\_ma();}

           break;

          }

        case 5:

        if(z<=0)

        {ob2.error1();

          break;}

        else

        {

         cout<<"Enter your Desired category=";

         cin>>inp;

         for(c=0;c<z;c++)

         {

           b=strcmp(ma[c].category,inp);

           if(b==0)

           break;

          }

          if(b==0)

            ma[c].remove();

           break;

          }

         case 6:

        if(z<=0)

         {ob2.error1();

          break;}

        else

        {

          for(q=g;q<z;q++)

          {

            f3=f3+ma[q];

            g++;

          }

        for(i=0;i<z;i++)

        ma[i].catego\_amount();

        f3.total\_amount();

        break;

        }

     }

     if(a==0)

     break;

       }

       break;

      }

      if(a1==0)

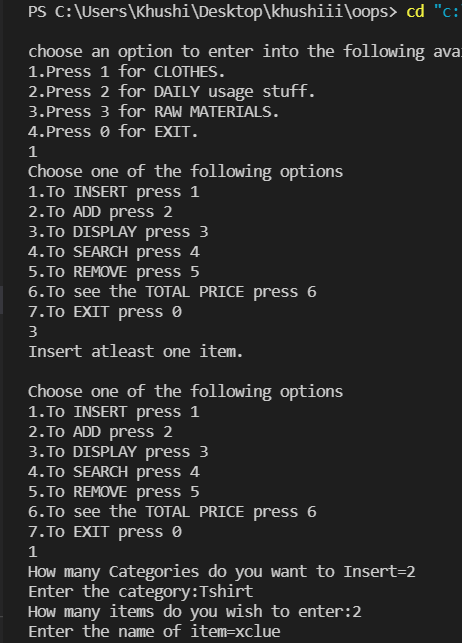
      break;

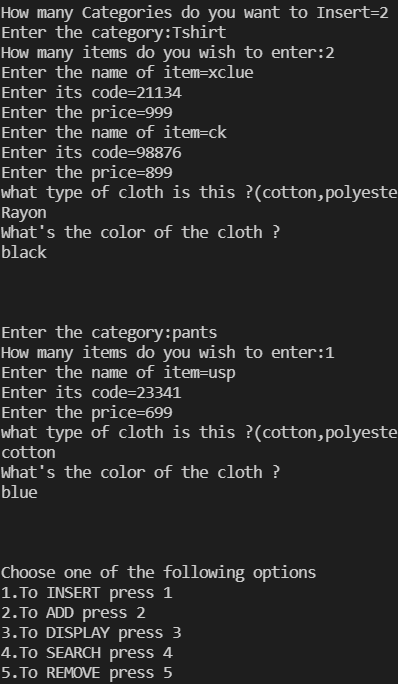
      }

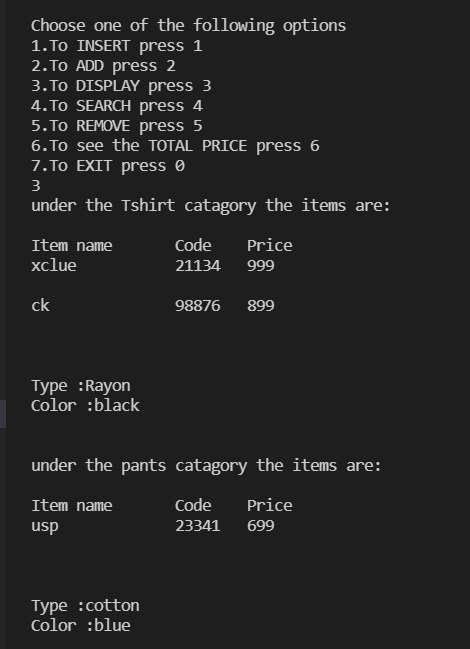
       return 0;

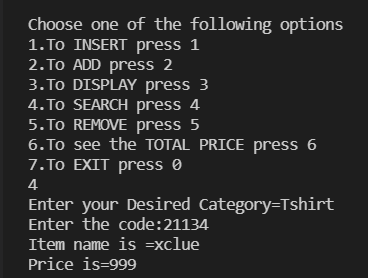
     }

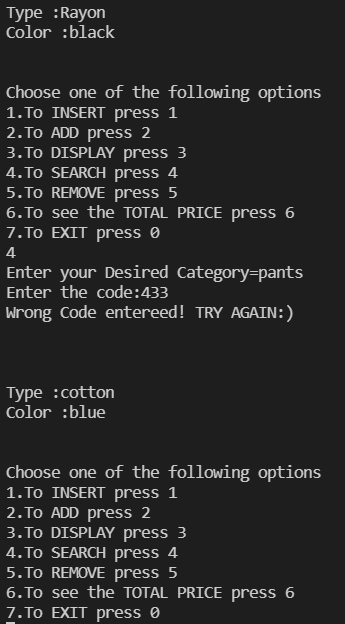
**Snapshots**

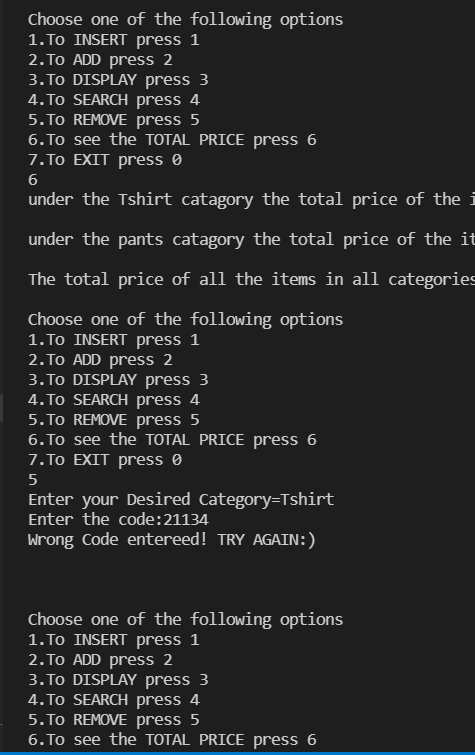
****

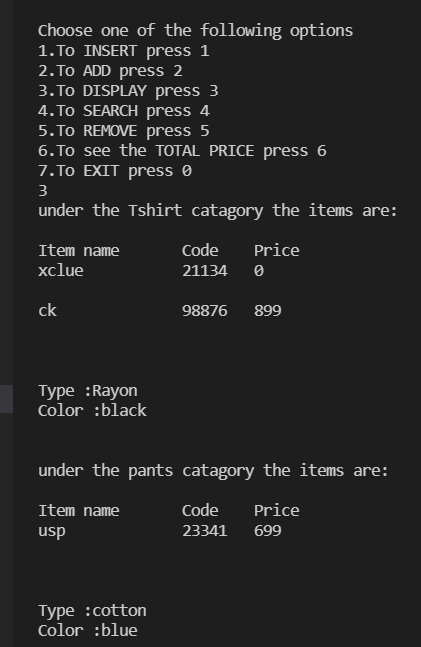












Reference

● Wikipedia

● Geeks for Geeks

● javapoint.com

**THANK YOU!**