Object-Oriented Programming (OOP) Project Report

Title: Inventory Management System

Abstract:

This project implements an Inventory Management System using C++ and object-oriented programming principles. The system allows users to add, delete, search, and display items in the inventory, which are categorized as Electronic, Frozen Food, or Beauty products.

Introduction:

The Inventory Management System is designed to help businesses keep track of their inventory efficiently. It provides functionalities to manage different types of products, their prices, and other relevant details.

Objective:

The main objective of this project is to demonstrate the use of object-oriented programming concepts in developing a practical application. Specifically, the project focuses on classes, inheritance, polymorphism, and file handling.

Tools and Technologies Used:

- C++ programming language and OOP principles
- Windows operating system (for console output formatting)
- Visual Studio Code (sometimes any other C++ IDE)

Design and Implementation:

Class Diagram:

The project consists of the following classes:

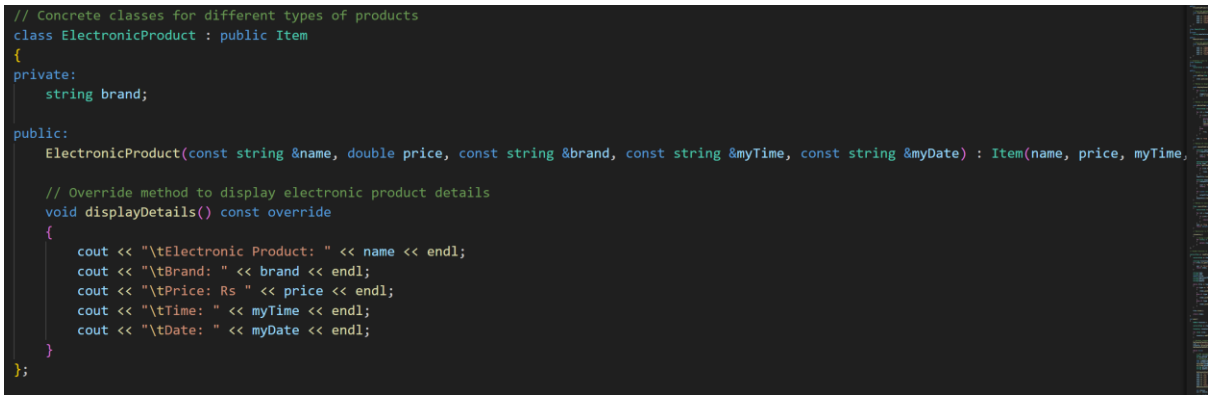
- **Item**: Abstract base class for all types of products, with attributes like name, price, time, and date.
- **ElectronicProduct**, **FrozenFoodProduct**, **BeautyProduct**: Concrete classes representing specific types of products, each with its own set of attributes.
- **Inventory**: Manages a collection of items, provides methods to add, delete, and display items.

Description of Classes:

Item: This class provides the basic structure for all types of products. It contains common attributes like name, price, time, and date. It also declares a pure virtual function `displayDetails()` which is overridden in the derived classes.

```
class Item
{
protected:
    string name;
    double price;
    string myTime;
    string myDate;

public:
    Item(const string &name, double price, const string &myTime, const string &myDate) : name(name), price(price), myTime(myTime), myDate(myDate) {}

    // Virtual method for displaying item details
    virtual void displayDetails() const = 0;

    // Getter for item name
    string getName() const
    {
        return name;
    }

    // Getter for item price
    double getPrice() const
    {
        return price;
    }

    string getTime() const
    {
        return myTime;
    }
```

ElectronicProduct, **FrozenFoodProduct**, **BeautyProduct**: These classes inherit from the Item class and represent specific types of products. Each class adds its own unique attributes and implements the `displayDetails()` function to display product details.



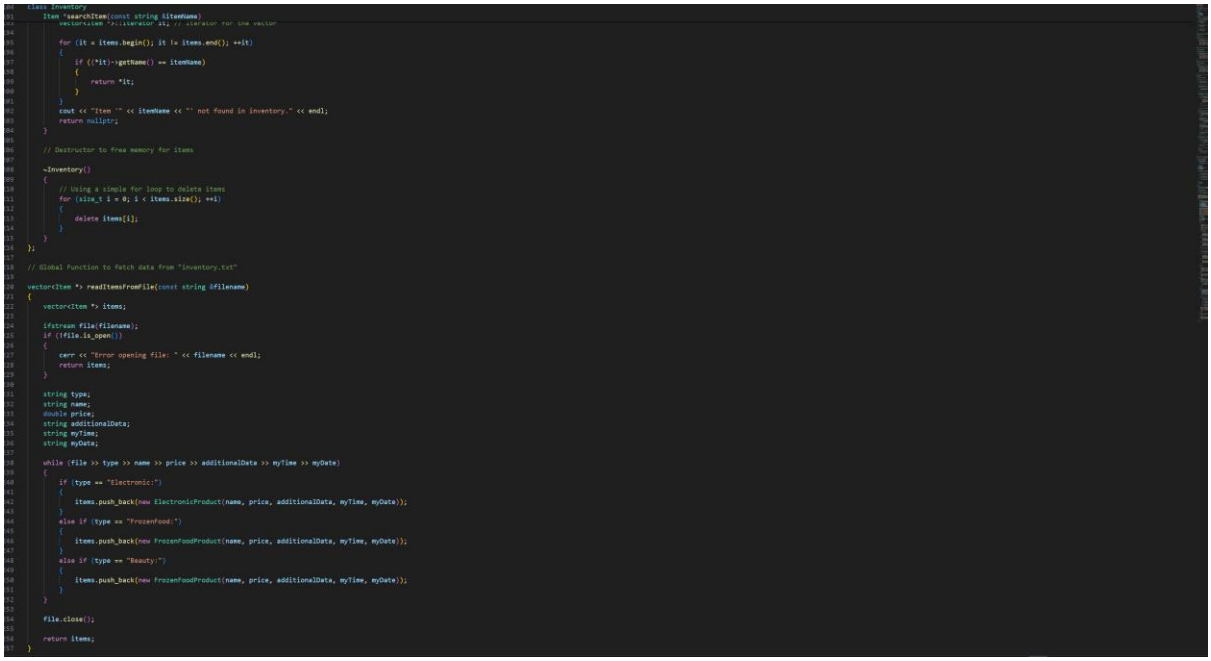
```
// Concrete classes for different types of products
class ElectronicProduct : public Item
{
private:
    string brand;

public:
    ElectronicProduct(const string &name, double price, const string &brand, const string &myTime, const string &myDate) : Item(name, price, myTime,

    // Override method to display electronic product details
    void displayDetails() const override
    {
        cout << "\tElectronic Product: " << name << endl;
        cout << "\tBrand: " << brand << endl;
        cout << "\tPrice: Rs " << price << endl;
        cout << "\tTime: " << myTime << endl;
        cout << "\tDate: " << myDate << endl;
    }
};
```

```cpp
class FrozenFoodProduct : public Item
{
private:
    string expiryDate;

public:
    FrozenFoodProduct(const string &name, double price, const string &expiryDate, const string &myTime, const string &myDate) : Item(name, price, my

    // Override method to display frozen food product details
    void displayDetails() const override
    {
        cout << "\tFrozen Food Product: " << name << endl;
        cout << "\tExpiry Date: " << expiryDate << endl;
        cout << "\tPrice: Rs " << price << endl;
        cout << "\tTime: " << myTime << endl;
        cout << "\tDate: " << myDate << endl;
    }
};
```

```cpp
class BeautyProduct : public Item
{
private:
    string manufacturer;

public:
    BeautyProduct(const string &name, double price, const string &manufacturer, const string &myTime, const string &myDate) : Item(name, price, myTi

    // Override method to display beauty product details
    void displayDetails() const override
    {
        cout << "\tBeauty Product: " << name << endl;
        cout << "\tManufacturer: " << manufacturer << endl;
        cout << "\tPrice: Rs " << price << endl;
        cout << "\tTime: " << myTime << endl;
        cout << "\tDate: " << myDate << endl;
    }
};
```

Inventory: This class manages a vector of Item pointers to store the inventory items. It provides methods to add, delete, display, and search items in the inventory. It also includes a method to remove items from the "inventory.txt" file.

```cpp
// Inventory class to manage items
class Inventory
{
private:
    vector<Item *> items;

public:
    // Method to add items to the inventory

    void addItem(Item *item)
    {
        items.push_back(item);
    }

    // Method to display all items in the inventory

    void displayInventory() const
    {
        for (size_t i = 0; i < items.size(); ++i)
        {
            items[i]->displayDetails();
            cout << endl;
        }
    }

    // Method to delete an item from the inventory by name

    void deleteItem(const string &itemName)
    {
        vector<Item *>::iterator it; // Iterator for the vector

        for (it = items.begin(); it != items.end();)
        {
            if ((*it)->getName() == itemName)
            {
                delete *it;
                it = items.erase(it); // Update iterator after erasing
                cout << "Item '" << itemName << "' deleted from inventory." << endl;
                removeItemFromFile(itemName);
                return;
            }
            else
            {
                ++it; // Move to the next element if not deleting
            }
        }

        cout << "Item '" << itemName << "' not found in inventory." << endl;
    }

    // Method to remove the item from "inventory.txt"

    void removeItemFromFile(const string &itemName)
    {
        ifstream inputFile(fileName);
        if (!inputFile.is_open())
        {
            cout << "Error: Unable to open file: inventory.txt" << endl;
            return;
        }

        vector<string> lines;
        string line;
        while (getline(inputFile, line))
        {
```



```
class Inventory
    Item *searchItem(const string &itemName)
        vector<Item *>::iterator it; // iterator for the vector

        for (it = items.begin(); it != items.end(); ++it)
        {
            if ((*it)->getName() == itemName)
            {
                return *it;
            }
        }
        cout << "Item '" << itemName << "' not found in inventory." << endl;
        return nullptr;
    }

    // Destructor to free memory for items

    ~Inventory()
    {
        // Using a simple for loop to delete items
        for (size_t i = 0; i < items.size(); ++i)
        {
            delete items[i];
        }
    }
};

// Global Function to fetch data from "inventory.txt"

vector<Item *> readItemsFromFile(const string &filename)
{
    vector<Item *> items;

    ifstream file(filename);
    if (!file.is_open())
    {
        cerr << "Error opening file: " << filename << endl;
        return items;
    }

    string type;
    string name;
    double price;
    string additionalData;
    string myTime;
    string myDate;

    while (file >> type >> name >> price >> additionalData >> myTime >> myDate)
    {
        if (type == "Electronic:")
        {
            items.push_back(new ElectronicProduct(name, price, additionalData, myTime, myDate));
        }
        else if (type == "FrozenFood:")
        {
            items.push_back(new FrozenFoodProduct(name, price, additionalData, myTime, myDate));
        }
        else if (type == "Beauty:")
        {
            items.push_back(new FrozenFoodProduct(name, price, additionalData, myTime, myDate));
        }
    }

    file.close();

    return items;
}
```

project.cpp ● inventory.txt ×

Obj project > inventory.txt

```
1   Electronic:     Laptop     1000     Lenovo     21:58:45     12/05/2024
2   Electronic:     MacBook     2000     Apple     21:59:00     12/05/2024
3   Beauty:     FaceWash     500     Garnier     21:59:34     12/05/2024
4   FrozenFood:     Spinach     10     05/06/2024     19:58:45     13/05/2024
5   Electronic:     iphone     10000     Apple     20:18:58     13/05/2024
6   FrozenFood:     Carrots     40     09/06/2024     20:19:21     13/05/2024
7   Beauty:     Lipstick     49     notourconcern     20:20:04     13/05/2024
8
```

Results and Testing:

The Inventory Management System was tested extensively to ensure that all functionalities work as expected. Various scenarios were tested, including adding new items, deleting existing items, searching for items, and displaying the inventory.

Project Conclusion:

The Inventory Management System project demonstrates the practical application of object-oriented programming concepts in developing a useful and efficient system. By using classes, inheritance, and polymorphism, the project provides a flexible and scalable solution for managing inventory.

Future Work:

Future enhancements to the project could include adding more functionalities such as updating item details, type of products(Hardware, Clothes, etc), generating reports, and implementing a graphical user interface (GUI) for easier interaction.

References:

- C++ documentation provided by Sir Talha Shahid and Miss Rafia Shaikh.

- YouTube Channels particularly “Code With Harry”.

Acknowledgements:

We would like to express our gratitude to Sir Talha Shahid for their guidance and support throughout the development of this project. Their valuable insights and feedback have been instrumental in shaping this Inventory Management System.

Conclusion:

In conclusion, the Inventory Management System project has been an enriching experience for Sheikh Naveed Azeemi ,Rao Ghulam Mohiuddin and Ghulam Murtaza allowing us to apply theoretical concepts of object-oriented programming to a practical scenario.