# EOOP - preliminary project

### Date:29.04.2022 Semester: Spring 2022

### Author and Group:!!!!!!!!!!!!!!

### Subject (Keyword): Tobacco factory

## **Description of the project**

### Overview of the project

### Tobacco factory has products which can be produced, materials which are needed to produce products, client who buy products and Supplier from whom materials are bought. The app buys materials to produce products ordered by the client automatically.

### Class and data structures overview

### There are following classes: Tobacco\_Factory, Product, Material, Client and Supplier.

### Tobacco\_Factory contains factory info and unlimited number of Products, Materials, Clients and Suppliers and method Menu in which manager can call method for add product, client, material and supplier, update product and material price, make order and delivery and print products, clients, materials, suppliers and factory info. This class has methods to choose product, material, client and supplier.

### Product contains data about product and unlimited number of pointers to materials and number of them which is need to produce one product and methods to add and remove material, update price and quantity and print product and materials.

### Material contains data about material and supplier which sell this material and methods to order material, update price and quantity and print material.

### Client contains data about client and unlimited number of pointers to product and number of them and methods to add and remove product, update, delete and print cart and print client.

### Supplier contains data about supplier and unlimited number of pointers to product and methods to add and remove material, update, delete and print order\_list and print supplier.

## **Case study (a memory map)**

**At the end of Final Project.**

## **Declaration of the classes**

class TobaccoFactory

{

private:

    string name;

// Name of the factory

    string address;

// Address of the factory

    string phone;

// Phone number of the factory

    string email;

// Email of the factory

    string website;

// Website of the factory

    vector<Product \*> products;

// Vector of products

    vector<Client \*> clients;

// Vector of clients

    vector<Material \*> materials;

// Vector of materials

    vector<Supplier \*> suppliers;

// Vector of suppliers

    void add\_product();

// Function to add a product with parameters from the manager

    void add\_client();

// Function to add a client with parameters from the manager

    void add\_material();

// Function to add a material with parameters from the manager

    void add\_supplier();

// Function to add a supplier with parameters from the manager

    void update\_product\_price();

// Function call Product\* chooseProduct() and call Product::updatePrice()

    void update\_material\_price();

// Function call Material\* chooseMaterial() and call Material::updatePrice()

    void make\_order();

// Function call Client\* chooseClient(), Product\* chooseProduct() and ask for the quantity of the product and call Client::addProduct\_to\_Cart()

    void add\_to\_cart(Client \*client);

// Function call Product\* chooseProduct() and call Client::addProduct\_to\_Cart()

    void make\_delivery(Client \*client);

// Function call Client\* chooseClient() and call Client::deleteCart()

    void print\_products();

// Function print all products in the vector

    void print\_clients();

// Function print all clients in the vector

    void print\_materials();

// Function print all materials in the vector

    void print\_suppliers();

// Function print all suppliers in the vector

    void print\_factory\_info();

// Function print information about the factory

    Product \*choose\_product();

// Function choose product from the vector using function Product::showProduct() and ask for the number of the product from the manager

    Material \*choose\_material();

// Function choose material from the vector using function Material::showMaterial() and ask for the number of the material from the manager

    Supplier \*choose\_supplier();

// Function choose supplier from the vector using function Supplier::showSupplier() and ask for the number of the supplier from the manager

    Client \*choose\_client();

// Function choose client from the vector using function Client::showClient() and ask for the number of the client from the manager

    void save\_to\_file(const string foldName = "data/");

    void load\_from\_file(const string foldName = "data/");

public:

    TobaccoFactory();

// Constructor

    void enter\_menu();

// Function to enter the menu and call the function according to the number

};

class Product

{

private:

    string name;

// Name of the product

    string description;

// Description of the product

    float price;

// Price of the product

    float price\_materials;

// Price of the materials used in the product

    map<Material \*, int> materials;

// Map of materials and their quantity which are used to produce the product

public:

    int quantity;

// Quantity of the product

    Product();

// Constructor

    Product(const string name, const string description, const float price, const int quantity);

// Constructor with parameters

    void add\_material(Material \*material, int quantity);

// Function to add material to the map

    void remove\_material();

// Function to remove material from the map

    void update\_price();

// Function to update the price of the product which ask for the new price from the manager and this price higher than sum of the materials prices

    void calculate\_price\_materials();

// Function to calculate the price of the materials used in the product

    void print\_product();

// Function to print the product

    void print\_materials();

// Function to print the materials used to produce the product

    void make(int quantity);

// Function to make the product

    string get\_name();

// Function to return the name of the product

    float get\_price();

// Function to return the price of the product

    float get\_price\_materials();

// Function to return the price of the materials used in the product

    void save\_to\_file(ofstream &file, int productID, map<Material \*, int> \*materialID, ofstream &materialFile);

// Function to save the product to the file

    void load\_from\_file(istringstream &iss, map<int, Product \*> \*productID);

// Function to load the product from the file

};

class Client

{

private:

    string name;

// Name of the client

    string address;

// Address of the client

    string phone;

// Phone number of the client

    string email;

// Email of the client

    map<Product \*, int> cart;

// Map of products and their quantity which are in the cart

public:

    Client();

// Constructor

    Client(const string name, const string address, const string phone, const string email);

// Constructor with parameters

    void add\_product\_to\_cart(Product \*product, int quantity);

// Function to add product to the cart

    void remove\_product\_from\_cart();

// Function to remove product from the cart

    void delete\_cart();

// Function to delete the cart

    void print\_cart();

// Function to print the cart

    void print\_client();

// Function to print the client

    void make\_products();

// Function to make products

    void deliver\_order();

// Function to delete the order

    float calculate\_total\_price();

// Function to calculate the total price of the cart

    float calculate\_total\_price\_materials();

// Function to calculate the total price of the materials used in the cart

    void save\_to\_file(ofstream &file, int clientID);

    void load\_from\_file(istringstream &iss, map<int, Client \*> \*clientID);

};

class Material

{

private:

    string name;

// Name of the material

    string description;

// Description of the material

    float price;

// Price of the material

    Supplier \*supplier;

// Supplier of the material

    vector<Product \*> products;

// Vector of products which are made with the material

public:

    int quantity;

// Quantity of the material

    Material();

// Constructor

    Material(const string name, const string description, const float price, const int quantity, Supplier \*supplier);

// Constructor with parameters

    void add\_product(Product \*product);

// Function to add product to the vector

    void delete\_product(Product \*product);

// Function to add product to the vector

    void update\_price();

// Function to update the price of the material which ask for the new price from the manager and this price higher than sum of the products prices

    void print\_material();

// Function to print the material

    void print\_products();

// Function to print the products which are made with the material

    float get\_price();

// Function to return the price of the material

    string get\_name();

// Function to return the name of the material

    void save\_to\_file(ofstream &file, int MaterialID, map<Supplier \*, int> \*supplierID);

// Function to save the material to the file

    void load\_from\_file(istringstream &iss, map<int, Material \*> \*materialID, map<int, Supplier \*> \*supplierID);

};

class Supplier

{

private:

    string name;

// Name of the supplier

    string address;

// Address of the supplier

    string phone;

// Phone number of the supplier

    string email;

// Email of the supplier

    vector<Material \*> materials;

// Vector of materials which are ordered

public:

    Supplier();

// Constructor

    Supplier(const string name, const string address, const string phone, const string email);

// Constructor with parameters

    void add\_material(Material \*material);

// Function to add material to the vector

    void delete\_material(Material \*material);

// Function to remove material from the vector

    void print\_materials();

// Function to print the order list

    void print\_supplier();

// Function to print the supplier

    string get\_name();

// Function to return the name of the supplier

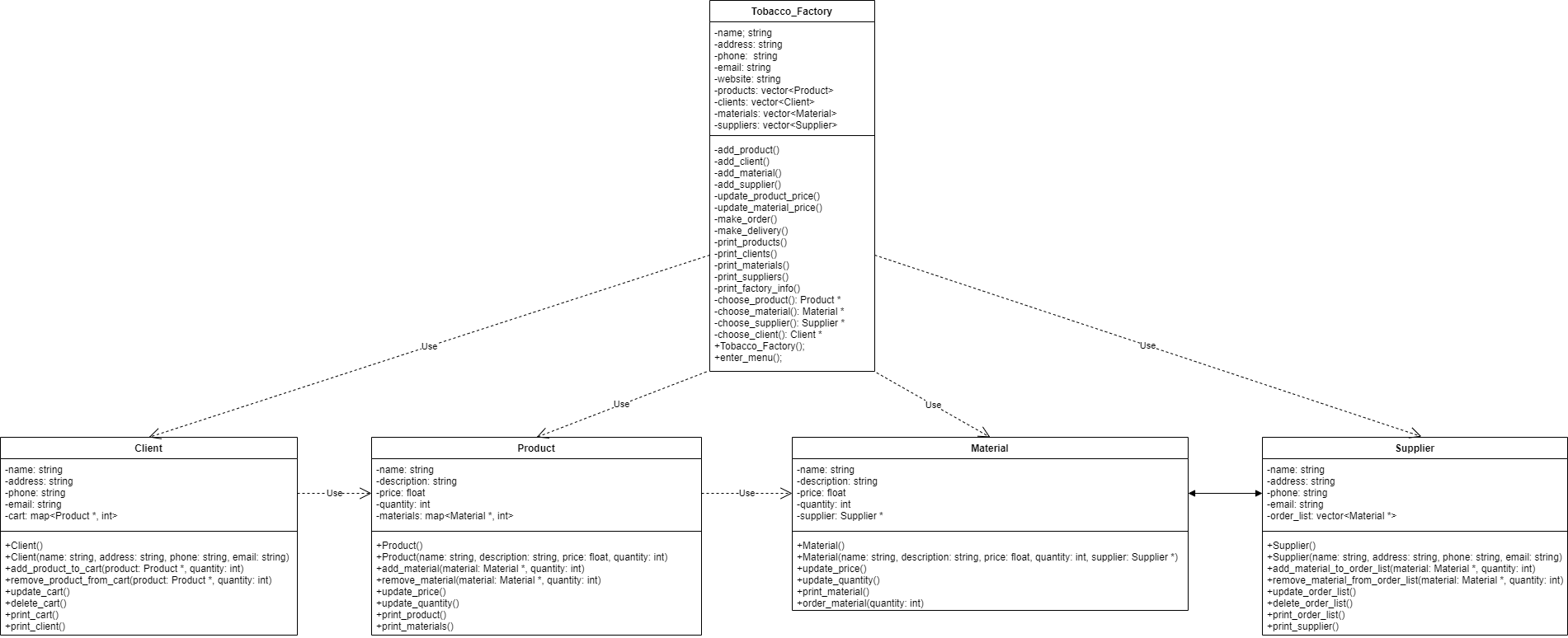
    void save\_to\_file(ofstream &file, int supplierID);

    void load\_from\_file(istringstream &iss, map<int, Supplier \*> \*supplierID);

};

## **Functional test cases**

* + - 1. **Try to enter negative number or higher than numbers in menu in Tobacco\_Factory::enterMenu(), Product \* Tobacco\_Factory::choose\_product(), Material \* Tobacco\_Factory::choose\_material(), Supplier \* Tobacco\_Factory::choose\_supplier(), Client \* Tobacco\_Factory::choose\_client()**
      2. **Try to enter negative number, when enter quantity or price in void Tobacco\_Factory::make\_order(), void Product::update\_price(), void Product::update\_quantity(), void Material::update\_price(), void Material::update\_quantity()**
      3. **Try to enter price of product which is lower than sum of price all materials needs to produce this product Product::update\_price().**

****