# 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 have products which can produce, materials which needs to produce products, client which buy products and Supplier from which buy materials. App automatically buy materials to produce products which order client.

### Class and data structures overview

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

### Tobacco\_Factory contain 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 contain data about product and unlimited number of pointers to materials and number of them which needs to produce one product and methods to add and remove material, update price and quantity and print product and materials.

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

### Client contain 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 contain 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 Preliminary Project.**

## **Declaration of the classes**

class Tobacco\_Factory

{

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 make\_delivery();

// 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

public:

    Tobacco\_Factory(); // 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

    int quantity;                   // Quantity of the product

    map<Material \*, int> materials;

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

public:

    Product(); // Constructor

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

// Constructor with parameters

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

// Function to add material to the map

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

// Function to remove material from the map

    void update\_price();

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

    void update\_quantity();    // Function to update the quantity of the product

    void print\_product();      // Function to print the product

    void print\_materials();

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

};

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(string name, string address, string phone, 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(Product \*product, int quantity);

// Function to remove product from the cart

    void update\_cart(); // Function to update 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

};

class Material

{

private:

    string name;        // Name of the material

    string description; // Description of the material

    float price;        // Price of the material

    int quantity;       // Quantity of the material

    Supplier \*supplier; // Supplier of the material

public:

    Material();   // Constructor

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

// Constructor with parameters

    void update\_price();

 // Function to update the price of the material with the new price from the manager

    void update\_quantity();                                                                   // Function to update the quantity of the material

    void print\_material();       // Function to print the material

    void order\_material(int quantity);

 // Function to order the material with the quantity

};

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 \*> order\_list; // Vector of materials which are ordered

public:

    Supplier();   // Constructor

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

// Constructor with parameters

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

// Function to add material to the order list

    void remove\_material\_from\_order\_list(Material \*material, int quantity);

// Function to remove material from the order list

    void update\_order\_list();       // Function to update the order list

    void delete\_order\_list();         // Function to delete the order list

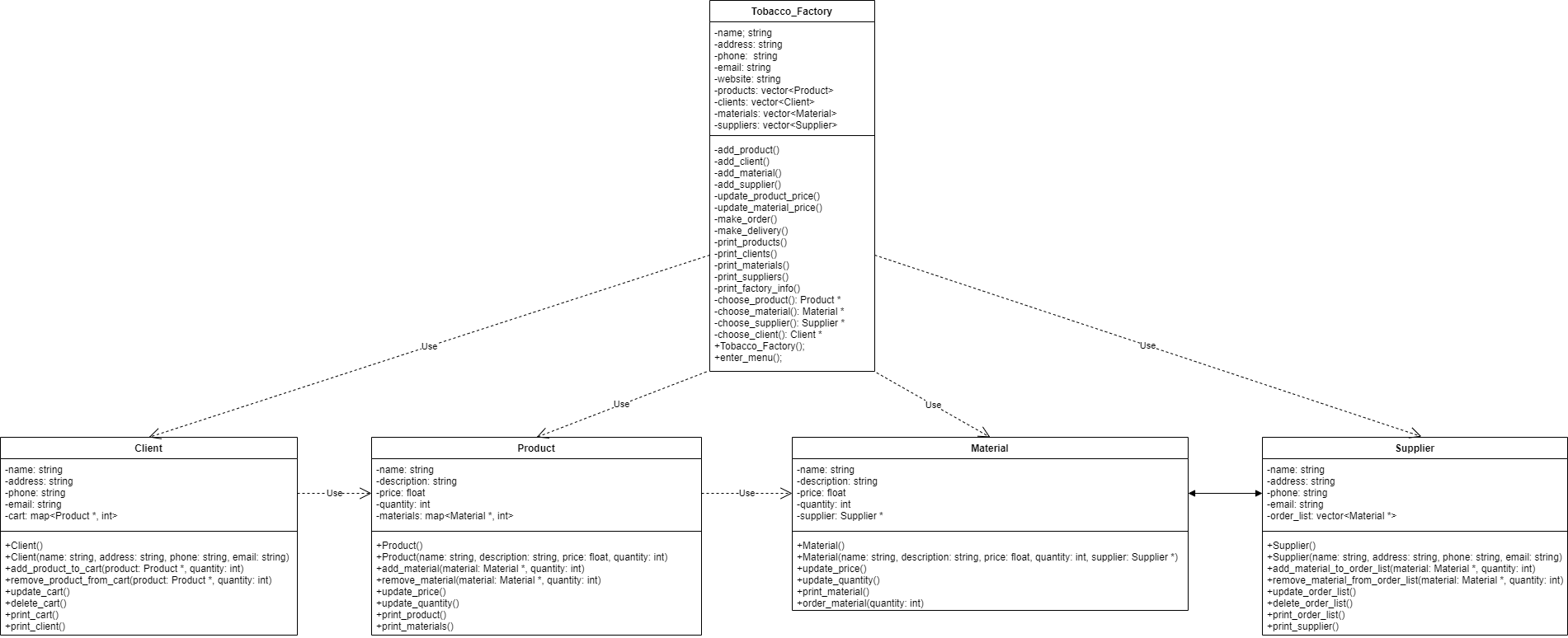
    void print\_order\_list();          // Function to print the order list

    void print\_supplier();            // Function to print the supplier

};

## **Functional test cases**

* + - 1. **Try to enter number negative or higher thin 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 lower then sum of price all materials needs to produce this product Product::update\_price().**

****