**Nom et prénom : EL MOUTAOUADI Ahmed Ali**

**Groupe : 7**

**Intitulé du mini projet : Panier Virtuel**

<Veuillez coller le code source de votre projet ici>

**Fichier .h:**

#include <iostream>

#include <deque>

#include <vector>

#ifndef Panier\_virtuel\_h

#define Panier\_virtuel\_h

using namespace std;

int menu();

class Article {

public:

int quantite;

string reference;

double prix;

Article();

Article(string ref, double price, int qte);

Article& operator=(const Article& other);

friend ostream& operator<<(ostream& os, const Article& obj);

bool operator<(const Article& other) const;

bool articlesIdentiques(Article A1, string ref);

void commande(int qt);

void remplirArticle(string ref, double pr, int qt);

void modifierQte(int qte);

};

class Stock {

public:

vector<Article> V;

Stock();

Stock(vector<Article> vec);

void remplirStock(Article A);

void afficherStock();

void diminuerQte(int qte, Stock& S, string ref);

};

class Panier {

public:

deque<Article> P;

Panier();

Panier(deque<Article> D);

Panier& operator=(const Panier& other);

bool ajouterArticle(string ref, Stock S, int qte);

void afficherPanier();

bool remise();

void lePlusCher();

void articlesEntreMinMax(double min, double max);

void infoArticles(string ref);

};

class Client {

protected:

string email;

string nom;

public:

Client();

Client(string mail, string name);

Client& operator=(const Client& other);

};

class Commande : public Client, Panier {

int numero;

public:

Commande();

Commande(int num);

void totalCmd(Panier& Pa);

};

#endif

**Fichier .cpp:**

#include <iostream>

#include <deque>

#include <string>

#include <algorithm>

#include "Panier\_Virtuel.h"

using namespace std;

#define RESET "\033[0m"

#define RED "\033[31m"

#define GREEN "\033[32m"

#define YELLOW "\033[33m"

#define MAGENTA "\033[35m"

#define CYAN "\033[36m"

// Fonction menu:

int menu() {

string choix;

cout << endl << "\033[1m" << "Bienvenu cher(e) client(e)" << RESET << endl << endl;

cout << "\t 1 - Ajouter un article a votre panier." << endl << endl;

cout << "\t 2 - Modifier la quantite commandee d\'un article deja commande." << endl << endl;

cout << "\t 3 - Supprimer un article du panier." << endl << endl;

cout << "\t 4 - Vider votre panier." << endl << endl;

cout << "\t 5 - Afficher le montant total de ma commande." << endl << endl;

cout << "\t 6 - Afficher l\'article le plus cher." << endl << endl;

cout << "\t 7 - Afficher les articles dont le prix est compris entre deux valeurs." << endl << endl;

cout << "\t 8 - Afficher les informations d\'un article." << endl << endl;

cout << "\t 9 - Quitter." << endl << endl;

cout << "\t Veuillez entrer votre choix: ";

while (true) {

getline(cin, choix);

// Check if the input consists only of digits

if (choix.find\_first\_not\_of("0123456789") == string::npos) {

// If input consists only of digits, convert it to an integer

int choice = stoi(choix);

// Check if the integer falls within the range 1-9

if (choice >= 1 && choice <= 9)

// If valid choice, return it

return choice;

else

// If not in the range, display error message and prompt again

cout << endl << RED << "--> Veuillez saisir un choix valide: " << RESET;

}

else

// If input contains non-digit characters, display error message and prompt again

cout << endl << RED << "--> Veuillez saisir un choix valide: " << RESET;

}

}

// Classe Article:

Article::Article() : reference(""), prix(0), quantite(0) {} // Constructeur par défaut

Article::Article(string ref, double price, int qte) { // Constructeur avec paramètres

reference = ref;

prix = price;

quantite = qte;

}

Article& Article::operator=(const Article& other) { // Surcharge de l'opérateur =

if (this != &other) {

reference = other.reference;

prix = other.prix;

quantite = other.quantite;

}

return \*this;

}

ostream& operator<<(ostream& os, const Article& obj) { // Surcharge de l'opérateur <<

os << "Reference: " << obj.reference << endl;

os << "Prix: " << obj.prix << "dhs" << endl;

os << "Quantite: " << obj.quantite << endl;

return os;

}

bool Article::operator<(const Article& other) const {

return prix < other.prix;

}

bool Article::articlesIdentiques(Article A1, string ref) {

if (A1.reference == ref)

return true;

return false;

}

void Article::remplirArticle(string ref, double pr, int qt) {

reference = ref;

prix = pr;

quantite = qt;

}

void Article::modifierQte(int qte) {

quantite -= qte;

}

// Classe Stock:

Stock::Stock() {} // Constructeur par défaut

Stock::Stock(vector<Article> vec) { // Constructeur avec paramètres

for (auto& i : V)

V.push\_back(i);

}

void Stock::remplirStock(Article A) {

string reference = "001";

double pr = 100;

int qt = 10;

for (int i = 0; i < 5; i++) {

A.remplirArticle(reference, pr, qt);

int ctr = 0; // Compteur pour parcourir la liste

for (char c : reference) {

if ((ctr == reference.length() - 1))

reference.erase(ctr, 1);

ctr++;

}

reference += to\_string(i + 2); // Ajouter à la fin de la liste le prochain chiffre (2-3-4-5)

pr -= 10;

qt += 10;

this->V.push\_back(A);

}

}

void Stock::afficherStock() {

cout << endl << "-- Stock --" << endl << endl;

for (auto& i : V)

cout << i << endl;

}

void Stock::diminuerQte(int qte, Stock& S, string ref) {

for (auto& i : S.V) {

if (i.reference == ref) {

i.modifierQte(qte);

break;

}

}

}

// Classe Panier:

Panier::Panier() {} // Constructeur par défaut

Panier::Panier(deque<Article> D) { // Constructeur avec paramètres

for (auto& i : P)

P.push\_back(i);

}

Panier& Panier::operator=(const Panier& other) { // Surcharge de l'opérateur =

if (this != &other) {

for (auto& i : other.P)

P.push\_back(i);

}

return \*this;

}

bool Panier::ajouterArticle(string ref, Stock S, int qte) {

bool articleTrouve = false;

for (auto& article : S.V) {

if (article.articlesIdentiques(article, ref)) {

article.quantite = qte;

this->P.push\_back(article);

articleTrouve = true;

break;

}

}

if (articleTrouve) {

cout << endl << CYAN << "--> Article ajoute avec succes." << RESET << endl;

return true;

}

else {

cout << endl << "--> Article non disponible!" << endl;

return false;

}

}

void Panier::afficherPanier() {

string choix; bool temp = false; int convertirInt{ 0 };

sort(P.begin(), P.end());

do {

if (!temp) {

cout << endl << "Voulez-vous afficher votre panier? (1 pour oui et 0 pour non): ";

getline(cin, choix);

}

if (choix.length() == 1) {

try {

system("cls");

convertirInt = stoi(choix);

if (convertirInt == 1) {

cout << endl << YELLOW << "--- Mon panier: ---" << endl;

if (!(P.empty())) {

for (auto& i : P)

cout << endl << i;

cout << RESET;

}

else

cout << endl << "Votre panier est vide." << endl;

temp = true;

break;

}

else if (convertirInt == 0)

break;

}

catch (const exception& e) {

cout << endl << RED << "--> Choix invalide, veuillez saisir 1 ou 0: " << RESET;

getline(cin, choix);

temp = true;

}

}

else {

cout << endl << RED << "--> Choix invalide, veuillez saisir 1 ou 0: " << RESET;

getline(cin, choix);

}

} while (convertirInt != 1 || convertirInt != 0);

}

void Panier::lePlusCher() {

double max = P[0].prix;

for (auto& article : P) {

if (article.prix > max)

max = article.prix;

}

if (!(P.empty())) {

for (auto& article : P) {

if (max == article.prix) {

cout << endl << "L\'article le plus cher est: " << endl;

cout << endl << article << endl;

break;

}

}

}

else

cout << endl << "Votre panier est vide." << endl;

}

void Panier::articlesEntreMinMax(double min, double max) {

for (auto& article : P) {

if (article.prix >= min && article.prix <= max)

cout << endl << article << endl;

}

}

void Panier::infoArticles(string ref) {

bool temp = false;

for (auto& article : P) {

if (article.reference == ref) {

cout << endl << article << endl;

temp = true;

break;

}

}

if (!temp)

cout << endl << "L\'article que vous souhaiter afficher n\'existe pas dans votre panier." << endl;

}

// Classe Client:

Client::Client() : email(""), nom("") {} // Constructeur par défaut

Client::Client(string mail, string name) { // Constructeur avec paramètres

email = mail;

nom = name;

}

Client& Client::operator=(const Client& other) { // Surcharge de l'opérateur =

if (this != &other) {

email = other.email;

nom = other.nom;

}

return \*this;

}

// Classe Commande:

Commande::Commande() : numero(0) {} // Constructeur par défaut

Commande::Commande(int num) { // Constructeur avec paramètres

numero = num;

}

void Commande::totalCmd(Panier& Pa) {

double somme = 0; bool temp = false;

for (auto& article : Pa.P) {

if (article.quantite > 10) {

somme += article.prix \* 0.2;

temp = true;

}

else

somme += article.prix;

}

if (temp)

cout << endl << "Le montant total de votre commande avec remise est: " << somme << " dhs." << endl;

else

cout << endl << "Le montant total de votre commande sans remise est: " << somme << " dhs." << endl;

}

**Fichier main:**

#include <iostream>

#include <sstream>

#include "Panier\_virtuel.h"

using namespace std;

#define RESET "\033[0m"

#define RED "\033[31m"

#define GREEN "\033[32m"

#define YELLOW "\033[33m"

#define BLUE "\033[34m"

#define MAGENTA "\033[35m"

#define CYAN "\033[36m"

int main() {

int choix, qteStock = 100, quantiteSaisie = 0, newQt, l = 0, llll;

bool test, temp = true, z;

string ref, yaw, r;

int sup{ 0 }, c;

string qtee;

Stock S;

Article A;

Panier monPanier;

Commande C;

S.remplirStock(A);

cout << R"( \_\_\_\_\_ \_ \_\_ \_\_\_ \_ \_

| \_\_ \ (\_) \ \ / (\_) | | | |

| |\_\_) |\_ \_ \_ \_\_ \_ \_\_\_ \_ \_\ \ / / \_ \_ \_\_| |\_ \_ \_ \_\_\_| |

| \_\_\_/ \_` | '\_ \| |/ \_ \ '\_\_\ \/ / | | '\_\_| \_\_| | | |/ \_ \ |

| | | (\_| | | | | | \_\_/ | \ / | | | | |\_| |\_| | \_\_/ |

|\_| \\_\_,\_|\_| |\_|\_|\\_\_\_|\_| \/ |\_|\_| \\_\_|\\_\_,\_|\\_\_\_|\_|)" << endl;

while (true) {

c = menu();

switch (c) {

case 1:

z = false;

do {

if (!z) {

system("cls");

cout << endl << "Veuillez saisir la reference de l\'article a ajouter dans votre panier: ";

getline(cin, ref);

}

if (ref != "001" && ref != "002" && ref != "003" && ref != "004" && ref != "005") {

cout << endl << RED << "--> Choix invalide, veuillez ressaisir: " << RESET;

getline(cin, ref);

z = true;

}

} while (ref != "001" && ref != "002" && ref != "003" && ref != "004" && ref != "005");

for (const auto& elem : S.V) {

if (ref == elem.reference) {

qteStock = elem.quantite; // La quantité du produit commandé dans le stock

break;

}

}

cout << endl << "Veuillez saisir la quantite desiree: ";

while (true) {

getline(cin, qtee);

// Si l'utilisateur n'a rien saisi

if (qtee.empty()) {

cout << endl << RED << "Veuillez saisir un nombre entier: " << RESET;

continue;

}

// Vérifier que chaque caractère est int

if (qtee.find\_first\_not\_of("0123456789") != string::npos) {

cout << endl << RED << "--> Veuillez saisir un nombre entier: " << RESET;

continue;

}

// Convertir en int

stringstream ss(qtee);

ss >> quantiteSaisie;

if (quantiteSaisie >= 1 && quantiteSaisie <= qteStock)

break;

else

cout << endl << RED << "--> La quantite saisie doit etre comprise entre 1 et " << qteStock << ": " << RESET;

}

test = monPanier.ajouterArticle(ref, S, quantiteSaisie);

if (test)

S.diminuerQte(quantiteSaisie, S, ref); // Actualiser la quantité du produit dans le stock

monPanier.afficherPanier();

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 2:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

cout << endl << "Veuillez saisir la reference de l\'article: ";

cin >> yaw;

cout << endl << "Veuillez saisir la nouvelle quantite: ";

cin >> newQt;

for (auto& i : monPanier.P) {

if (i.reference == yaw) {

i.quantite = newQt;

break;

}

}

monPanier.afficherPanier();

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 3:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

cout << endl << "Veuillez saisir la reference de l\'article a supprimer: ";

cin >> yaw;

for (auto& i : monPanier.P) {

if (i.reference == yaw) {

monPanier.P.erase(monPanier.P.begin() + l);

cout << endl << MAGENTA << "--> Article supprime avec succes." << RESET << endl;

sup = 1;

}

l++;

}

if(sup == 0)

cout << endl << "--> Article introuvable." << endl;

monPanier.afficherPanier();

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 4:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

do {

if (temp) {

cout << endl << "Voulez-vous vraiment vider votre panier? (1 pour oui et 0 pour non): ";

cin >> choix;

}

if (choix == 1) {

monPanier.P.clear();

cout << endl << "--> Votre panier est vide.";

break;

}

else if (choix == 0) {

break;

}

else {

cout << endl << "Veuillez saisir un choix valide: ";

cin >> choix;

temp = false;

}

if (choix == 0 || choix == 1) {

if (choix == 1) {

monPanier.P.clear();

cout << endl << "--> Votre panier est vide." << endl;

}

monPanier.afficherPanier();

}

} while (choix != 0 && choix != 1);

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 5:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

system("cls");

C.totalCmd(monPanier);

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 6:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

system("cls");

monPanier.lePlusCher();

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 7:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

double min, max;

cout << endl << "Veuillez saisir le min puis le max: ";

cin >> min;

cin >> max;

system("cls");

monPanier.articlesEntreMinMax(min, max);

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 8:

if (monPanier.P.empty()) {

system("cls");

cout << endl << "Votre panier est vide, veuillez le remplir avant d\'effectuer cette operation." << endl;

break;

}

cout << endl << "Veuillez saisir la reference de l\'article: ";

cin >> r;

system("cls");

monPanier.infoArticles(r);

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 9: exit(0);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

default:

cout << endl << "Veuillez saisir un choix valable!" << endl;

}

}

//S.afficherStock(); // si vou voulez afficher le stock

}