Dokumentacja programisty – Shop Z03

1. Klasa Product

Zadaniem klasy jest obiektowa reprezentacja produktu odzieżowego znajdującego się w sklepie. Klasa przechowuje podstawowe informacje dotyczące produktu, jako swoje atrybuty:

- **string name** – nazwa produktu
- **int id** – numer referencyjny, jest to unikalny numer danego produktu
- **Money price** – cena produktu
- **int views** – liczba wyświetleń produktu przez użytkowników
- **Date adding_date** – data dodania produktu do sklepu
- **vector < string > categories** – wektor przechowujący nazwy kategorii produktów
- **vector < Size > available_models** – wektor przechowujący struktury **Size**, posiadającą atrybut **size** (nazwa rozmiaru) oraz **amount** (liczba sztuk danego rozmiaru w sklepie)
- **string description** – opis produktu

Klasa posiada jeden konstruktor, który tworzy obiekt produktu. Jego argumenty są przypisywane odpowiednio do atrybutów klasy.

Metody klasy dzielą się głownie na metody **get** i **set**. Metody **get** zwracają niemodyfikowalne atrybuty klasy, metody **set** umożliwiają zmianę wartości atrybutów klasy. Metody **set** zostały stworzone z myślą o rozwoju programu w przyszłości. Zostałby wtedy stworzony specjalny interfejs dla właściciela, do wprowadzania nowych produktów oraz modyfikacji już istniejących.

Na pozostałe metody składają się:

- **void view_plus**() - zwiększa **views** o jeden, przy każdorazowym użyciu,
- **bool operator** == (**Product other**) - porównuje produkty po **id,**
- **void add_category**(**string cat**) - do **categories** dodaje kolejną kategorie produktu,
- **void remove_category**(**string cat**) - usuwa kategorie z **categories** produktu,
- **bool is_in_categories**(**string cat**) - sprawdza czy kategoria znajduję się w atrybucie **categories,**
- **void remove_from_stock**(**string s**, **int val**) - modyfikuje ilość produktu w rozmiarze **s** o wartość **val**, modyfikacja opiera się na działaniu odejmowania,
- **void print_cout**() - wyprowadza, bezpośrednio na strumień, szczegółowe informacje dotyczące produktu,
- **friend ostrem**& **operator** << (**ostream**& **os**, **Product product**)- jest to funkcja zaprzyjaźniona, która za pomocą uchwytu **os** wyprowadza skrócone informacje o produkcie na strumień.

2. *Klasa Cart*

Zadaniem klasy jest obiektowa reprezentacja koszyka w sklepie internetowym. Jej istotną właściwością jest to, że jest to klasa szablonowa. Dzięki temu, nie ogranicza się do operowania jedynie na zdefiniowanej w programie klasie Product dostosowanej na potrzeby sklepu odzieżowego. Umożliwia to rozwój programu oraz rozszerzenie asortymentu sklepu o nowe kategorie produktów.

Klasa **Cart** posiada następujące atrybuty:

- **vector<Elem> cart –** wektor przechowujący elementy dodawane do koszyka,
- **delivery_methods delivery –** struktura zawierająca informacje o dodatkowych kosztach związanych z dostawą.

Element przechowywany w koszyku jest zdefiniowany przez strukturę **Elem**. Posiada ona następujące atrybuty:

- **int value** – liczba sztuk produktu dodanego do koszyka w danym rozmiarze
- **string product_size –** nazwa rozmiaru produktu dodanego do koszyka
- **shared_ptr<T> product** – inteligentny wskaźnik na dany produkt

Metody klasy:

- **bool check_availability(shared_ptr<T> prod, string prod_size, int value) –** ma na celu sprawdzić dostępność podanej ilości produktu w danym rozmiarze
- **void add_product(shared_ptr<T> prod, string prod_size, int value=1)** – dodaje nowy element do koszyka, bez podania value domyślnie przyjmuje wartość jeden
- **void delete_product(int id, int val=1)** – usuwa element z koszyka
- **int get_size()** – zwraca liczbę wszystkich elementów w koszyku, uwzględniając liczbę sztuk każdego obiektu **Elem**
- **int get_number() –** zwraca liczbę obiektów Elem znajdujących się w koszyku
- **void set_delivery_methods(delivery_methods dev) –** ustawia atrybut delivery
- **Elem& operator[](int id), Elem const& operator[](int id) const –** operator wyłuskania
- **Money get_amount() –** zwraca całkowitą kwotę zawartości koszyka
- **Money free_delivery() –** zwraca kwotę jakiej brakuje do otrzymania darmowej dostawy
- **bool is_free_delivery() –** sprawdza czy całkowita kwota zawartości koszyka pozwala na otrzymanie darmowej dostawy
- **void clean_cart() –** usuwa wszystkie obiekty z koszyka
- **delivery_methods get_delivery_methods() –** zwraca atrybut klasy **delivery**

3. Klasa Order

Klasa jest obiektową reprezentacją składania zamówienia. Jako atrybuty klasa przechowuje najważniejsze informację oraz elementy potrzebne do złożenia zamówienia.

Atrybuty klasy:

- **data_order data; –** struktura przechowująca podstawowe informacje o zamawiającym
- **Cart<Product> cart; –** koszyk, który przechowuje obiekty zamawianych elementów
- **Date date_order; –** datę dokonania zamówienia
- **delivery_method dev; –** wybrany przez klienta sposób dostawy zamówienia
- **bool payment; –** wartość boolowską określająca status płatności

Klasa w większości posiada metody typu get i set. Metody set (tj. **set_name, set_surname, set_street, set_city, set_post_code, set_telephone, set_email, set_delivery_method**) pobierają informacje od użytkownika o nim oraz o miejscu dostawy produktu. Metody typu get (tj. **get_data_to_delivery, get_cart, get_date_order, get_delivery_method**) zwracają bez możliwości modyfikacji atrybuty klasy.

Pozostałe metody klasy:

- **Money pay()** – wykonuje ponowne sprawdzenie dostępności zamawianych produktów, redukuje odpowiednio ilość asortymentu sklepu o dokonane zamówienie
- **void confirm_payment(bool condition)** – zmienia status płatności, w momencie niedokonanej płatności przywraca zamówione produkty do asortymentu sklepu
- **string get_order()** – zwraca całość sformatowanych informacji o zamówieniu
- **Money get_amount_with_delivery()** – zwraca kwotę zamówienia po uwzględnieniu dostawy

4. *Klasa Database*

Klasa jest obiektową reprezentacją bazy danych sklepu. Jej głównym zadaniem jest przechowywanie listy wszystkich produktów i danych do wysyłki. Umożliwia również wyszukiwanie i sortowanie wyników. Wypełnienie bazy może nastąpić poprzez odczyt danych z pliku, jak również ręczne ich dodawanie.

Klasa Database posiada atrybuty:

- **Vector<Product> products –** lista wszystkich produktów w bazie
- **Vector<Products> search_results –** lista produktów spełniających warunki wyszukiwania
- **Vector<Order> new_orders –** lista nowych zamówień
- **Delivery_methods delivery –** struktura przechowująca informacje o sposobach dostawy

Klasa posiada potrzebne settery i gettery oraz dodatkowe metody:

- **void add_product(const Product& new_prod) –** metoda dodająca obiekt klasy Product do listy produktów, rzuca wyjątek gdy produkt jest już w bazie
- **void remove_product(const int id) –** metoda usuwająca z listy produkt o określonym id, rzuca wyjątek, gdy nie ma produktu o danym id
- **void print_delivery_methods() const –** metoda wyświetlająca informacje o sposobach dostawy
- **void print() –** metoda wyświetlająca dane wszystkich produktów z bazy
- **void print_results() –** metoda wyświetlająca dane produktów spełniających warunki wyszukiwania
- **const Product& operator[](const int id) –** metoda zwracająca produkt z listy o podanym id, rzuca wyjątek gdy produktu nie ma w bazie
- **void add_new_order(const Order& new_ord) –** metoda dodająca obiekt klasy order do listy nowych zamówień
- **void search_by_name(string name) –** metoda dodająca do listy *search_results* wszystkie produkty zaczynające się od podanego ciągu znaków
- **void search_by_category(string category) –** metoda dodająca do listy *search_results* wszystkie produkty, które należą do kategorii zaczynającej się od podanego ciągu znaków
- **void sort_results_by_price(bool ascending) –** metoda sortująca produkty na liście *search_results* według ceny w porządku rosnącym, jeśli ascending ma wartość true, lub malejącym w przeciwnym przypadku
- **void sort_results_by_views(bool ascending) –** metoda sortująca produkty na liście *search_results* według liczby wyświetleń w porządku rosnącym, jeśli ascending ma wartość true, lub malejącym w przeciwnym przypadku
- **void sort_results_by_adding_date(bool ascending) –** metoda sortująca produkty na liście *search_results* według daty dodania od najstarszych, jeśli ascending ma wartość true, lub najmłodszych w przeciwnym przypadku
- **void read_products_from_txt(string path) –** metoda wczytująca dane z pliku o podanej ścieżce, tworząca z nich strukturę *delivery*, a następnie obiekty klasy Product, które są dodawane do listy wszystkich produktów *products.*

- **void write_products_to_txt(string path) –** metoda zapisująca do pliku o podanej ścieżce dane o sposobach dostawy i dane wszystkich produktów z listy *products* w formacie identycznym jak plik, którego są pobierane dane
- **void write_orders_to_txt(string path) –** metoda zapisująca do pliku o podanej ścieżce dane wszystkich nowych zamówień złożonych od ostatniego generowania pliku
- **void write_searching_to_txt(string path, string name) –** metoda dopisująca do pliku o podanej ścieżce informacje o czasie i wyszukiwanej nazwie

5. Klasa Money

Klasa jest obiektową reprezentacją pieniądza w złotówkach. Aby stworzyć obiekt Money należy wywołać konstruktor **Money(int z, int g)** i w jego argumentach odpowiednio wpisać ilość złotówek i groszy w cenie. W klasie są zdefiniowane operatory takie jak: **operator+**, **operator-**, **operator==**, **operator<**, **operator>**, które umożliwiają podstawowe operacje na pieniądzach.

6. Klasa Date

Klasa jest obiektową reprezentacją daty. Aby stworzyć obiekt Date należy wywołać konstruktor **Date(int day, int month, int year)**, którego argumenty odpowiednio oznaczają dzień, miesiąc, rok.

Kacper Bober

Jakub Cholewiński

Paulina Dąbrowska